По договору между издательством «Символ-Плюс» и Интернет-магазином «Books.Ru – Книги России» единственный легальный способ получения данного файла с книгой ISBN 5-93286-035-9, название «DNS и BIND, 4-е издание» – покупка в Интернет-магазине «Books.Ru – Книги России». Если Вы получили данный файл каким-либо другим образом, Вы нарушили международное законодательство и законодательство Российской Федерации об охране авторского права. Вам необходимо удалить данный файл, а также сообщить издательству «Символ-Плюс» (piracy@symbol.ru), где именно Вы получили данный файл.

DNS and BIND

Fourth Edition

Paul Albitz and Cricket Liu

O'REILLY®

DNS и BIND

Четвертое издание

Пол Альбитц и Крикет Ли

Санкт-Петербург
2002

Пол Альбитц, Крикет Ли

DNS и BIND, 4-е издание

Перевод М. Зислиса

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Научный редактор	*А. Маврин*
Редактор	*А. Лосев*
Корректура	*О. Маршкова*
Верстка	*Н. Гриценко*

Альбитц П., Ли К.
DNS и BIND. – Пер. с англ. – СПб: Символ-Плюс, 2002. – 696 с., ил.
ISBN 5-93286-035-9

Книга «DNS и BIND» стала библией для системных администраторов. Она уникальна по полноте изложения материала, что в сочетании с прекрасным авторским стилем делает ее незаменимой и актуальной для каждого, кто хочет наладить эффективную работу DNS. Четвертое издание включает информацию о версии 9 пакета BIND, в которой реализованы новые и очень важные механизмы, а также о версии 8, входящей в состав большинства действующих коммерческих разработок. Пакеты BIND 8 и 9 позволяют значительно повысить безопасность служб DNS.

Рассмотрены следующие темы: функциональность и принципы работы DNS; структура пространства доменных имен; установка и настройка серверов имен; применение MX-записей для маршрутизации почты; настройка узлов на работу с DNS; разделение доменов на поддомены; обеспечение безопасности DNS-сервера; новые возможности BIND 9; расширения системы безопасности DNS (DNSSEC) и подписи транзакций (TSIG); распределение нагрузки между DNS-серверами; динамические обновления, асинхронные уведомления об изменениях зон, пошаговая передача зон; устранение неполадок (применение nslookup и dig, чтение отладочного вывода); DNS-программирование с применением библиотеки DNS-клиента и модуля Perl Net::DNS.

ISBN 5-93286-035-9
ISBN 0-596-00158-4 (англ)

© Издательство Символ-Плюс, 2002
Authorized translation of the English edition © 2001 O'Reilly & Associates Inc. This translation is published and sold by permission of O'Reilly & Associates Inc., the owner of all rights to publish and sell the same.

Все права на данное издание защищены Законодательством РФ, включая право на полное или частичное воспроизведение в любой форме. Все товарные знаки или зарегистрированные товарные знаки, упоминаемые в настоящем издании, являются собственностью соответствующих фирм.

Издательство «Символ-Плюс». 193148, Санкт-Петербург, ул. Пинегина, 4,
тел. (812) 324-5353, edit@symbol.ru. Лицензия ЛП N 000054 от 25.12.98.
Налоговая льгота – общероссийский классификатор продукции
ОК 005-93, том 2; 953000 – книги и брошюры.
Подписано в печать 26.02.2002. Формат 70x100 1/16 . Печать офсетная.
Объем 43,5 печ. л. Тираж 3000 экз. Заказ N
Отпечатано с диапозитивов в Академической типографии «Наука» РАН
199034, Санкт-Петербург, 9 линия, 12.

Оглавление

Предисловие 9

1. Основы 20

(Очень) краткая история сети Интернет 20
Интернет и интернет-сети 21
Система доменных имен в двух словах 24
История пакета BIND 29
Надо ли мне использовать DNS? 30

2. Как работает DNS 32

Пространство доменных имен 32
Пространство доменных имен сети Интернет 39
Делегирование 42
DNS-серверы и зоны 44
Клиенты DNS 49
Разрешение имен 49
Кэширование 58

3. С чего начать? 61

Приобретение пакета BIND 61
Выбор доменного имени 66

4. Установка BIND 83

Наша зона 84
Создание данных для зоны 85
Создание файла настройки BIND 98
Сокращения 101
Проверка имени узла (BIND 4.9.4 и более поздние версии) 105
Инструменты 108
Запуск первичного мастер-сервера DNS 109
Запуск вторичного DNS-сервера 115
Добавление зон 123
Что дальше? 124

5. DNS и электронная почта 125

MX-записи 126
И все-таки, что такое почтовый ретранслятор? 129
MX-алгоритм 131

6. Конфигурирование узлов 135

DNS-клиент 135
Примеры настройки DNS-клиента 149
Как упростить себе жизнь 151
Специфика настройки различных систем 157

7. Работа с BIND 180

Управление DNS-сервером 180
Обновление файлов данных зон 190
Организация файлов 200
Перемещение системных файлов в BIND 8 и 9 205
Ведение log-файла в BIND 8 и 9 206
Основы благополучия 218

8. Развитие домена 240

Сколько DNS-серверов? 240
Добавление DNS-серверов 249
Регистрация DNS-серверов 255
Изменение значений TTL 259
Подготовка к бедствиям 263
Борьба с бедствиями 267

9. Материнство 272

Когда заводить детей 273
Сколько детей? 273
Какие имена давать детям 274
Заводим детей: создание поддоменов 276
Поддомены доменов in-addr.arpa 287
Заботливые родители 293
Как справиться с переходом к поддоменам 298
Жизнь родителя 301

10. Дополнительные возможности 302

Списки отбора адресов и управления доступом 303
DNS: динамические обновления 304
DNS NOTIFY (уведомления об изменениях зоны) 312

Инкрементальная передача зоны (IXFR) . . . 317
Ретрансляция . . . 321
Виды . . . 326
Round Robin: распределение нагрузки . . . 328
Сортировка адресов DNS-сервером . . . 332
DNS-серверы: предпочтения . . . 338
Нерекурсивный DNS-сервер . . . 339
Борьба с фальшивыми DNS-серверами . . . 340
Настройка системы . . . 342
Совместимость . . . 353
Основы адресации в IPv6 . . . 354
Адреса и порты . . . 356
IPv6: прямое и обратное отображение . . . 360

11. Безопасность . . . 367

TSIG . . . 368
Обеспечение безопасности DNS-сервера . . . 373
DNS и брандмауэры сети Интернет . . . 389
Расширения системы безопасности DNS . . . 414

12. nslookup и dig . . . 442

Насколько хорош nslookup? . . . 443
Пакетный или диалоговый? . . . 445
Настройка . . . 445
Как отключить список поиска . . . 449
Основные задачи . . . 449
Прочие задачи . . . 453
Разрешение проблем с nslookup . . . 461
Лучшие в сети . . . 467
Работа с dig . . . 467

13. Чтение отладочного вывода BIND . . . 473

Уровни отладки . . . 473
Включение отладки . . . 477
Чтение отладочной диагностики . . . 478
Алгоритм работы DNS-клиента
и отрицательное кэширование (BIND 8) . . . 491
Алгоритм работы DNS-клиента
и отрицательное кэширование (BIND 9) . . . 492
Инструменты . . . 493

14. Разрешение проблем DNS и BIND 494

Виновата ли служба NIS? 495
Инструменты и методы 496
Перечень возможных проблем 504
Проблемы перехода на новую версию 524
Проблемы сосуществования и версий 525
Ошибки TSIG 530
Симптомы проблем 531

15. Программирование при помощи функций библиотеки DNS-клиента 538

Написание сценариев командного интерпретатора с помощью программы nslookup 539
Программирование на языке C при помощи функций библиотеки DNS-клиента 545
Программирование на языке Perl при помощи модуля Net::DNS 573

16. Обо всем понемногу 577

Использование CNAME-записей 577
Маски 582
Ограничение MX-записей 583
Коммутируемые соединения 584
Имена и номера сетей 590
Дополнительные RR-записи 592
DNS и WINS 600
DNS и Windows 2000 602

A. Формат сообщений DNS и RR-записей 609

B. Таблица совместимости BIND 630

C. Сборка и установка BIND на Linux-системах 632

D. Домены высшего уровня 637

E. Настройка DNS-сервера и клиента BIND 643

Алфавитный указатель 664

Предисловие

Возможно, вам не так уж много известно о системе доменных имен (Domain Name System), но работая в Интернете, вы неизбежно ее используете. Всякий раз, отправляя сообщения электронной почты или исследуя просторы World Wide Web, вы полагаетесь на DNS – систему доменных имен.

Дело в том, что люди предпочитают запоминать *имена* компьютеров, а компьютерам больше нравится обращаться друг к другу по числовым адресам. В Интернете этот адрес имеет разрядность 32, то есть может быть числом от нуля до четырех с хвостиком миллиардов.[1] Компьютеры с легкостью запоминают такие вещи, потому что обладают большими объемами памяти, идеально подходящей для хранения чисел, но для людей эта задача не в пример сложнее. Попробуйте случайным образом выбрать из телефонной книги десять номеров и запомнить их. Непросто? Теперь вернитесь к началу телефонной книги и сопоставьте каждому номеру случайный код района. Примерно настолько же сложно будет запомнить 10 произвольных интернет-адресов.

Отчасти именно по этой причине необходима система доменных имен. DNS занимается двунаправленным отображением имен хостов, подходящих для запоминания людьми, и интернет-адресов, с которыми работают компьютеры. По сути дела, DNS в сети Интернет является не только средством работы с адресами, но и стандартным механизмом для предоставления и получения разнообразной информации об узлах сети. DNS нужен практически для каждой программы, обеспечивающей сетевое взаимодействие, в том числе программам для работы с электронной почтой, терминальным клиентам (например, telnet), средствам передачи файлов, таким как FTP, и, разумеется, веб-броузерам, таким как Netscape Navigator и Microsoft Internet Explorer.

Другой важной особенностью DNS является способность системы распространять информацию о хосте по всей сети Интернет. Хранение доступной информации о хосте на единственном компьютере полезно лишь для тех, кто пользуется этим компьютером. Система доменных имен обеспечивает получение информации из любой точки сети.

Более того, DNS позволяет распределять управление информацией о хостах между многочисленными серверами и организациями. Нет необ-

---

[1] А в системе IP-адресации версии 6 адреса имеют колоссальную длину – 128 бит, что позволяет охватить десятичные числа от нуля до 39-значных.

ходимости передавать данные на какой-то центральный сервер или регулярно синхронизировать свою базу данных с «основной». Достаточно убедиться, что ваш раздел, называемый *зоной*, соответствует действительности на ваших *DNS-серверах*. А они, в свою очередь, сделают информацию о зоне доступной всем остальным DNS-серверам сети.

Поскольку база данных DNS является распределенной, в системе должна быть предусмотрена возможность поиска нужной информации путем опроса множества возможных источников ее получения. Система доменных имен наделяет DNS-серверы способностью находить нужные источники информации и получать сведения по любой зоне.

Разумеется, система DNS не лишена недостатков. К примеру, в целях избыточности базы данных, система позволяет хранение зональной информации на более чем одном сервере. При этом возникает опасность десинхронизации копий зональной информации.

Но *самая большая* проблема, связанная с DNS, несмотря на широкое распространение в сети Интернет, – это реальное отсутствие хорошей документации по работе с системой. Большинство администраторов сети Интернет вынуждены обходиться лишь той документацией, которую считают достаточной поставщики используемых программ, а также тем, что им удается выудить из соответствующих списков интернет-рассылок и конференций Usenet.

Такой дефицит документации означает, что понимание предельно важной интернет-службы, одной из монументальных основ сегодняшней сети Интернет, либо передается от администратора к администратору как ревностно хранимая семейная тайна, либо постоянно изучается повторно отдельными программистами и разработчиками. Новые администраторы зон повторяют ошибки, уже бесчисленное число раз сделанные другими.

Цель этой книги – изменить сложившуюся ситуацию. Мы осознаем, что не у каждого читателя есть время и желание становиться специалистом по DNS. У большинства из вас есть достаточно других занятий, помимо управления зонами и DNS-серверами: системное администрирование, разработка сетевых инфраструктур, или разработка программного обеспечения. Заниматься исключительно DNS может только сотрудник безумно большой организации. Мы постарались представить информацию, достаточную для решения основных рабочих задач, будь то управление небольшой зоной или целой международной системой, работа с единственным сервером имен или наблюдение за сотней серверов. Извлеките из книги нужный вам минимум, и возвращайтесь к ней по мере необходимости.

DNS – это сложная тема, настолько сложная, что взяться за нее пришлось не одному, а двум авторам; и мы постарались представить систему настолько прозрачно и доступно, насколько это возможно. В первых двух главах содержится теоретический обзор и достаточный для применения объем практической информации, а в последующих гла-

вах использование системы доменных имен рассмотрено более подробно. С самого начала мы предлагаем читателям нечто вроде дорожной карты, чтобы каждый мог выбрать свой собственный путь изучения книги, соответствующий рабочим задачам или интересам.

Когда речь пойдет о программах, обеспечивающих работу DNS, мы практически целиком сконцентрируемся на инструменте под названием BIND, Berkeley Internet Name Domain, который является наиболее популярной (и наиболее нами изученной) реализацией спецификаций DNS. Мы старались представить в этой книге выжимку из нашего опыта управления и поддержки зон с помощью BIND. (Так получилось, что некоторое время одна из наших зон являлась самой большой зоной сети Интернет; правда, это было очень давно). Где это было возможно, мы включали реальные программы, используемые нами в администрировании; многие из них переписаны на языке Perl с целью достижения большей скорости работы и повышения эффективности.

Надеемся, эта книга поможет вам познакомиться с системой DNS и инструментом BIND, если вы еще новичок, лучше понять их работу, если вы уже знакомы, и приобрести ценное понимание и опыт, даже если вы уже знаете DNS и BIND, как свои пять пальцев.

Версии

Четвертое издание этой книги затрагивает новые версии BIND – 9.1.0 и 8.2.3, а также более старую версию 4.9. Несмотря на то, что на момент написания этой книги версии 9.1.0 и 8.2.3 являются наиболее свежими, они пока не получили широкого распространения в составе Unix-систем, отчасти потому, что обе версии были выпущены недавно, а многие поставщики настороженно относятся к использованию новых программ. Мы время от времени упоминаем и другие версии BIND, в частности 4.8.3, поскольку многие поставщики продолжают распространять программы, содержащие код, основанный на более старых версиях, в составе своих Unix-разработок. Если определенная возможность доступна только в версии 4.9, 8.2.3 или 9.1.0, либо если существуют различия в поведении версий, мы постараемся четко определить, что именно работает и для какой версии BIND.

В наших примерах мы очень часто прибегаем к служебной программе DNS – *nslookup*. Мы пользуемся *nslookup* из комплекта поставки BIND версии 8.2.3. Более старые версии *nslookup* обеспечивают большую часть функциональности (но не всю) *nslookup* версии 8.2.3.[1] В большинстве примеров мы использовали команды, доступные почти во всех версиях *nslookup*; случаи, когда это было невозможно, отмечены отдельно.

[1] Это верно также и для nslookup из комплекта поставки BIND версии 9.1.0. См. более подробно в главе 12 «nslookup и dig».

Что нового в четвертом издании?

Текст книги был обновлен, чтобы соответствовать наиболее поздним версиям BIND; мы также добавили в большом объеме новый материал:

- Более подробное рассмотрение динамических обновлений и механизма NOTIFY, включая и подписываемые динамические обновления (signed dynamic updates), а также описание нового для BIND 9 механизма *update-policy* – в главе 10.
- Поэтапная передача зоны – также в главе 10.
- Зоны ретрансляции, поддерживающие передачу по условию (conditional forwarding), – в главе 10.
- Прямое и обратное отображение адресов в контексте технологии IPv6 с использованием записей новых типов A6 и DNAME, а также бит-строковых меток – в конце главы 10.
- Новый механизм подтверждения подлинности транзакций – транзакционные подписи (transaction signatures, известные также как TSIG) – описан в главе 11.
- Более подробное рассмотрение вопросов обеспечения безопасности DNS-серверов – в главе 11.
- Более подробное рассмотрение работы с брандмауэрами в сети Интернет – в главе 11.
- Описаны расширения DNS, связанные с безопасностью (DNS Security Extensions или DNSSEC), представляющие собой новый механизм цифровой подписи зональных данных, – все в той же 11-ой главе.
- Раздел, посвященный совместной работе клиентов, серверов и контроллеров доменов Windows 2000 и BIND, – в главе 16.

Структура

Порядок следования глав настоящей книги приблизительно соответствует возможному развитию зоны и росту знаний ее администратора. В главах 1 и 2 обсуждается теория системы доменных имен. В главах с 3 по 6 рассматриваются вопросы, связанные с принятием решений по созданию собственных зон, а также действия администратора в случае необходимости создать зону. Следующая часть книги, главы с 7 по 11, посвящена сопровождению зон, настройке хостов для использования DNS-серверов, планированию развития зон, созданию доменов различных уровней и безопасности серверов. Наконец, главы с 12 по 16 посвящены разрешению сложностей, возникающих при работе с различными инструментами, общим проблемам и забытому искусству программирования с применением библиотек DNS-клиента.

Перечислим темы по главам:

Глава 1 «Основы» описывает исторический фон создания системы, и посвящена проблемам, приведшим к созданию DNS, а также собственно обзору теории системы доменных имен.

Глава 2 «Как работает DNS» посвящена более подробному рассмотрению теоретических основ DNS, в частности – организации пространства имен в системе DNS, доменов, зон и DNS-серверов. Там же рассматриваются такие важные понятия, как разрешение адресов и кэширование.

Глава 3 «С чего начать?» посвящена вопросам получения пакета BIND в случае его отсутствия, применениям пакета, когда он уже у вас в руках, определению и выбору доменного имени, а также установления связи с организацией, которая обладает полномочиями делегировать выбранную зону.

Глава 4 «Установка BIND» – это подробное рассмотрение того, как установить два первых DNS-сервера на основе BIND, включая создание базы данных серверов, запуск и диагностику их работы.

Глава 5 «DNS и электронная почта» рассказывает о записи DNS типа MX, которая позволяет администраторам задавать альтернативные узлы, которым передается на обработку почта для определенных адресов. В этой главе описаны стратегии маршрутизации почты для различных типов сетей и узлов, включая сети с интернет-брандмауэрами и узлы, не имеющие прямого подключения к сети Интернет.

Глава 6 «Конфигурирование узлов» рассказывает о том, как настраивать клиентскую часть (*resolver*) BIND, а также об особенностях реализаций клиента – как в составе распространенных Unix-систем, так и применяемых на платформах Windows 95/NT/2000.

Глава 7 «Работа с BIND» посвящена регулярным действиям администратора, выполнение которых необходимо для поддержания устойчивой работы зон, находящихся под его началом, в частности – проверке состояния DNS-сервера и вопросов, касающихся авторитативных серверов зоны.

Глава 8 «Развитие домена» рассказывает о планировании роста и эволюции зон, включая вопросы о том, как вырасти большим, а также о планировании переездов и перебоев в работе.

Глава 9 «Материнство» – о радостях, связанных с обретением потомства. Мы расскажем, когда имеет смысл заводить детей (создавать поддомены), как их называть, *как* их заводить (!) и как присматривать за ними.

Глава 10 «Дополнительные возможности» рассказывает о параметрах настройки сервера имен, которые используются не очень часто, но могут помочь в тонкой настройке DNS-сервера и в упрощении процесса администрирования.

Глава 11 «Безопасность» посвящена обеспечению безопасности и тем настройкам DNS-сервера, которые относятся к работе с интернет-

брандмауэрами, а также двум новым технологиям DNS, связанным с безопасностью: DNS Security Extensions и подписям транзакций (Transaction Signatures).

Глава 12 «nslookup и dig» подробно рассказывает о самых популярных инструментах DNS-отладки, и содержит описания способов извлечения неявной информации из удаленных DNS-серверов.

Глава 13 «Чтение отладочного вывода BIND» – это Розеттский камень[1] отладочной информации BIND. Глава поможет разобраться в таинственной отладочной информации, создаваемой пакетом BIND, а это, в свою очередь, поможет лучше понять, как работает DNS-сервер

Глава 14 «Разрешение проблем DNS и BIND» содержит описания и способы разрешения многих распространенных проблем, связанных с использованием DNS и BIND, а также рассказывает о более редких случаях, связанных с ошибками, диагностика которых может вызывать затруднения.

Глава 15 «Программирование с использованием библиотечных функций» рассказывает о том, как использовать функции библиотеки клиента BIND для опроса DNS-серверов и получения информации в программе на языке C или Perl. Приводится исходный текст полезной (как мы надеемся) программы, которая проверяет работоспособность DNS-серверов и их авторитативность.

Глава 16 «Обо всем понемногу» посвящена незатронутым темам. Она содержит описание использования масок (wildcards) в DNS, принципов работы с хостами и сетями, не имеющими постоянного подключения к сети Интернет, кодировки сетевых имен, экспериментальных типов записей и работы с DNS в Windows 2000.

Приложение A «Формат сообщений DNS и RR-записи (resource records)» содержит предельно подробный справочник по форматам, используемым в запросах и ответах DNS, а также полный перечень определенных в настоящее время типов RR-записей.

Приложение B «Таблица совместимости BIND» – это перечисление наиболее важных особенностей самых распространенных версий BIND.

Приложение C «Сборка и установка BIND на Linux-системах» содержит пошаговые инструкции по сборке BIND версии 8.2.3 в Linux.

---

[1] Розеттский камень – черная базальтовая плита с трехъязычной надписью на египетском иероглифическом, египетском демотическом (разговорном) и древнегреческом языках, обнаруженная в 1799 г. офицером наполеоновских войск Бушаром при сооружении форта Сен-Жюльен на берегу Розеттского рукава Нила. Расшифровка иероглифического текста в 1822 г. стала началом изучения египетской иероглифической письменности. – *Примеч. ред.*

Приложение D «Домены высшего уровня» – это перечисление существующих в настоящее время доменов высшего уровня сети Интернет.

Приложение E «Настройка DNS-сервера и клиента BIND» содержит справочник по синтаксису и семантике каждого из существующих параметров настройки серверов и библиотек клиента.

Для кого эта книга

Прежде всего эта книга предназначена для системных и сетевых администраторов, которым приходится управлять зонами и одним или несколькими DNS-серверами, но она содержит материал, который будет интересен проектировщикам сетей, почтовым администраторам и многим другим людям. Не все главы одинаково интересны для столь разношерстной аудитории, и, конечно же, читателю нет смысла копаться во всех шестнадцати главах, чтобы найти интересующий его материал. Мы надеемся, что следующая карта поможет выстроить правильный путь по главам книги.

Системным администраторам, впервые столкнувшимся с вопросами сопровождения зон, следует прочесть главы 1 и 2, чтобы получить теоретическую подготовку по DNS, главу 3 – в целях получения информации о первых шагах и выборе подходящего доменного имени, главы 4 и 5 – чтобы узнать, как происходит настройка зоны «с нуля». Глава 6 объясняет, как настроить хосты для работы с новыми DNS-серверами. Несколько позже следует обратиться к главе 7, в которой объясняется, как «подкачать» объем, добавляя серверы и данные в зону. Главы 12, 13 и 14 содержат описание инструментов и методов, помогающих в устранении проблем.

Опытным администраторам будет полезно прочитать главу 6, чтобы узнать, как настраивать DNS-клиенты на различных хостах, и главу 7, чтобы получить информацию о том, как грамотно сопровождать зоны. В главе 8 содержатся инструкции, связанные с планированием роста и развития зоны, которые должны быть особенно полезны людям, занятым в администрировании больших зон. Глава 9 рассказывает о том, как можно стать родителем – то есть, о создании поддоменов, и является учебником *этикета*, обязательным к прочтению теми, кто планирует совершить этот трудный шаг. В главе 10 рассмотрены многие новые возможности BIND версий 8.2.3 и 9.1.0. Глава 11 посвящена обеспечению безопасности DNS-серверов и для опытных администраторов может представлять особенный интерес. Главы с 12 по 14 содержат описание действий на случай возникновения проблем и сопутствующих инструментов, эти главы могут оказаться занимательным чтением даже для очень опытных администраторов.

Системным администраторам сетей, не имеющих постоянного подключения к сети Интернет, рекомендуется прочесть главу 5, чтобы изучить процесс настройки маршрутизации почты в таких сетях, и

главу 11, которая содержит описание создания независимой инфраструктуры DNS.

Программистам, в целях освоения теории DNS, предлагается прочесть главы 1 и 2, а затем главу 15, в которой содержится подробное рассмотрение программирования при помощи библиотечных функций BIND.

Сетевым администраторам, которые напрямую не вовлечены в процесс сопровождения зон, рекомендуется прочесть главы 1 и 2, в целях освоения теории DNS, главу 12, чтобы научиться использовать *nslookup* и *dig*, а затем главу 14, чтобы узнать о способах разрешения возникающих сложностей.

Почтовым администраторам следует прочесть главы 1 и 2, в целях освоения теории DNS, главу 5, чтобы узнать, как сосуществуют DNS и электронная почта, и главу 12, в которой описаны инструменты *nslookup* и *dig*, – эта глава научит извлекать информацию о маршрутизации почты из пространства доменных имен.

Заинтересованные пользователи могут прочесть главы 1 и 2, в целях освоения теории DNS, а затем – любые главы, по желанию!

Мы предполагаем, что читатель знаком с основами администрирования Unix-систем, сетевым взаимодействием TCP/IP, а также программированием на уровне простых сценариев командного интерпретатора или языка Perl. При этом никаких других специальных знаний не требуется. При появлении новых терминов и понятий они насколько возможно подробно объясняются в тексте книги. По возможности мы использовали аналогии с системами Unix (и реальным миром), чтобы облегчить читателю восприятие новых для него концепций.

Примеры программ

Исходные тексты программ-примеров, приводимых в книге, доступны для загрузки по протоколу FTP по следующим адресам:

ftp://ftp.uu.net/published/oreilly/nutshell/dnsbind/dns.tar.Z
ftp://ftp.oreilly.com/published/oreilly/nutshell/dnsbind/

В обоих случаях извлечь файлы из архива можно командой:

```
% zcat dns.tar.Z | tar xf -
```

На System V – системах необходимо использовать следующую *tar*-команду:

```
% zcat dns.tar.Z | tar xof -
```

Если команда *zcat* недоступна в системе, следует использовать отдельные команды *uncompress* и *tar*.

Если не удается получить тексты примеров напрямую по сети Интернет, но существует возможность посылать и получать сообщения электронной почты, можно воспользоваться службой *ftpmail*. Чтобы получить справку по использованию службы *ftpmail*, необходимо отправить сообщение на адрес *ftpmail@online.oreilly.com*. Следует оставить пустым поле темы сообщения; тело письма должно содержать единственное слово – «help».

Как связаться с издательством O'Reilly

Комментарии и вопросы, связанные с этой книгой, можно направлять непосредственно издателю:

O'Reilly & Associates, Inc.
101 Morris Street
Sebastopol, CA 95472
(800) 998-9938 (в США или Канаде)
(707) 829-0515 (международный/местный)
(707) 829-0104 (факс)

Издательством O'Reilly создана веб-страница, посвященная этой книге, на которой доступна информация о найденных ошибках и будут появляться разнообразные дополнительные сведения. Страница доступна по адресу:

http://www.oreilly.com/catalog/dns4

Если у вас есть технический вопрос или комментарий, связанный с этой книгой, задайте его, отправив сообщение по адресу:

bookquestions@oreilly.com

На веб-сайте издательства O'Reilly доступна дополнительная информация о книгах, конференциях, программном обеспечении, источниках информации и сети O'Reilly (O'Reilly Network):

http://www.oreilly.com

Типографские соглашения

Использованы следующие соглашения по шрифту и формату для команд, инструментов и системных вызовов Unix:

- Выдержки из сценариев или конфигурационных файлов оформлены моноширинным шрифтом:

```
if test -x /usr/sbin/named -a -f /etc/named.conf
then
    /usr/sbin/named
fi
```

- Примеры диалоговых сеансов, отображающие ввод в командной строке и соответствующую реакцию системы, оформлены непропорциональным шрифтом, причем ввод пользователя отмечен жирным выделением:

```
% cat /var/run/named.pid
78
```

- Если команда должна вводиться суперпользователем (администратором системы, или пользователем root), она предваряется символом диеза (#):

```
# /usr/sbin/named
```

- Заменяемые элементы кода оформлены моноширинным курсивом.
- Имена доменов, файлов, функций, команд, названия страниц руководства Unix, фрагменты кода оформлены курсивом, если они расположены внутри параграфа.

Цитаты

Цитаты из Льюиса Кэррола в каждой из глав приводятся по версии 2.9 издания Millenium Fulcrum электронного текста «Алисы в Стране чудес» из библиотеки Проекта Гутенберга (Project Gutenberg) и по изданию 1.7 текста «Алиса в Зазеркалье». Цитаты в главах 1, 2, 5, 5, 8 и 14 из «Алисы в стране чудес», а цитаты в главах 3, 4, 7, 9, 10, 11, 12, 13, 15 и 16 – из «Алисы в зазеркалье».[1]

Благодарности

Авторы выражают благодарность Кену Стоуну (Ken Stone), Джерри МакКоллому (Jerry McCollom), Питеру Джеффу (Peter Jeffe), Хэлу Стерну (Hal Stern), Кристоферу Дарему (Christopher Durham), Биллу Уизнеру (Bill Wisner), Дэйву Керри (Dave Curry), Джеффу Окамото (Jeff Okamoto), Брэду Ноулзу (Brad Knowles), Роберту Эльцу (K. Robert Elz), а также Полу Викси (Paul Vixie) за их бесценный вклад в написание этой книги. Мы также хотели бы поблагодарить наших рецензентов – Эрика Пирса (Eric Pearce), Джека Репенинга (Jack Repenning), Эндрю Черенсона (Andrew Cherenson), Дэна Тринкла (Dan Trinkle), Билла Лефевра (Bill LeFebvre) и Джона Секреста (John Sechrest) за их критику и предложения. Без помощи этих людей эта книга была бы совсем не такой (а была бы она гораздо короче!).

За второе издание этой книги авторы выражают благодарность безупречной команде рецензентов: Дэйву Бэрру (Dave Barr), Найджелу

[1] В русском издании для цитат используется перевод Нины Демуровой (М.:, ПРЕССА, 1992). – *Примеч. ред.*

Кэмпбеллу (Nigel Campbell), Биллу Лефевру, Майку Миллигану (Mike Milligan) и Дэну Тринклу.

За третье издание книги авторы отдают честь команде мечты технических рецензентов: Бобу Хэлли (Bob Halley), Барри Марголину (Barry Margolin) и Полу Викси.

Долг благодарности за четвертое издание причитается Кевину Данлэпу (Kevin Dunlap), Эдварду Льюису (Edward Lewis) и Брайану Веллингтону (Brian Wellington), первоклассной команде рецензентов.

Крикет хотел бы отдельно поблагодарить своего бывшего руководителя, Рика Норденстена (Rick Nordensten), образцового современного высокопроизводительного менеджера, под присмотром которого была написана первая версия этой книги; своих соседей, которые терпели его эпизодическую раздражительность в течение многих месяцев, и, конечно же, свою жену Пэйдж за постоянную поддержку и за то, что она мирилась с непрекращающимся, даже во время ее сна, стуком клавиш. Что касается второго издания, Крикет хотел бы добавить слова благодарности в адрес своих бывших руководителей Регины Кершнер (Regina Kershner) и Пола Клоуда (Paul Klouda) за их поддержку работы Крикета с сетью Интернет. За помощь в работе над третьим изданием Крикет считает своим долгом поблагодарить своего партнера, Мэтта Ларсона (Matt Larson), который участвовал в разработке Acme Razor; за четвертое он благодарит своих преданных пушистиков Дакоту и Энни – за их поцелуи и участие, а также замечательного Уолтера Б. (Walter B), который время от времени заглядывал в кабинет и проверял, как у Папы дела. Пол благодарит свою жену Катерину за ее терпение, за многочисленные разборы полетов и за доказательство того, что она в свободное время может гораздо быстрее сшить стеганое одеяло, чем ее муж напишет свою половину книги.

Мы хотим сказать спасибо ребятам из O'Reilly & Associates, за их тяжелый труд и терпение. В особенности этой благодарности заслуживают наши редакторы – Майк Лукидес (Mike Loukides) (издания с первого по третье) и Дебра Кэмерон (Debra Cameron) (четвертое издание), а также огромное количество других людей, которые работали над различными изданиями этой книги: Нэнси Котари (Nancy Kotary), Элли Фонтэйн Мэйден (Ellie Fountain Maden), Роберт Романо (Robert Romano), Стивен Абрамс (Steven Abrams), Кишмет МакДоноу-Чен (Kismet McDonough-Chan), Сет Мэйслин (Seth Maislin), Элли Катлер (Ellie Cutler), Майк Сьерра (Mike Sierra), Ленни Мельнер (Lenny Muellner), Крис Райли (Chris Reilley), Эмили Куилл (Emily Quill), Анна-Мария Вадува (Anne-Marie Vaduva) и Брэнда Миллер (Brenda Miller). Также спасибо Джерри Пику (Jerry Peek) за самую разнообразную поддержку, и Тиму О'Рейли за то, что он вдохновил нас на написание этой книги.

И спасибо Эди за сверчка[1] на обложке!

[1] Cricket (фамилия одного из авторов) переводится с англ. как «сверчок». – *Примеч. перев.*

1

Основы

- *(Очень) краткая история сети Интернет*
- *Интернет и интернет-сети*
- *Система доменных имен в двух словах*
- *История пакета BIND*
- *Надо ли мне использовать DNS?*

Кролик надел очки.
– С чего начинать, Ваше Величество? – спросил он.
– Начни с начала, – важно ответил Король, – продолжай, пока не дойдешь до конца. Как дойдешь – кончай!

Чтобы понять DNS, необходимо обратиться к истории сети ARPAnet. Система DNS была создана с целью решения конкретных проблем этой сети, а сеть Интернет, выросшая из ARPAnet, сейчас является главным потребителем этих решений.

Опытные пользователи сети Интернет, вероятно, могут пропустить эту главу. Все прочие, мы надеемся, найдут здесь достаточно информации для понимания причин, которые привели к появлению DNS.

(Очень) краткая история сети Интернет

В конце шестидесятых Управление передовых исследований Министерства обороны США (Department of Defense's Advanced Research Agency, или ARPA) – позднее DARPA – открыло финансирование ARPAnet, экспериментальной глобальной компьютерной сети, объединившей важные исследовательские организации страны. Первоначальной целью создания этой сети было разделение дорогостоящих или дефицитных компьютерных ресурсов между государственными подрядчиками. Но с самого начала пользователи ARPAnet использовали сеть и для совместной работы: они обменивались файлами и программами, сообщениями электронной почты (получившей теперь повсеместное распространение), объединяли усилия по разработке и исследованиям, используя разделяемые ресурсы компьютеров сети.

Набор протоколов TCP/IP (Transmission Control Protocol/Internet Protocol) был разработан в начале восьмидесятых годов и быстро получил

статус стандарта для сетевого обмена информацией между узлами ARPAnet. Включение этого набора протоколов в популярную операционную систему BSD Unix, разработанную в Калифорнийском университете Беркли, сыграло определяющую роль в процессе демократизации и объединения сетей. Система BSD Unix университетам была доступна практически бесплатно. Так работа с сетями и доступ к ARPAnet стали внезапно доступными и очень дешевыми для большого числа организаций, которые ранее никак не были связаны с ARPAnet. Машины, подключавшиеся к ARPAnet, входили также в состав локальных сетей, и в итоге это привело к объединению многочисленных разрозненных локальных сетей посредством ARPAnet.

Сеть разрослась с очень небольшого числа узлов до десятков тысяч. Первоначальная сеть ARPAnet стала основой объединения локальных и региональных сетей, работающих по протоколам TCP/IP. Это объединение носит название *Интернет*.

Однако в 1988 году организация DARPA пришла к заключению, что эксперимент окончен. Министерство обороны приступило к демонтажу сети ARPAnet. В этот момент несущим остовом для сети Интернет стала другая сеть, которая финансировалась национальным научным фондом (National Science Foundation) и носила название NSFNET.

Уже не так давно, весной 1995 года, сеть Интернет в очередной раз сменила главную магистраль; финансируемая обществом NSFNET уступила место целому ряду коммерческих магистралей, управляемых операторами дальней связи, такими как MCI и Sprint, а также такими опытными коммерческими сетевыми организациями как PSINet и UUNET.

Сегодня сеть Интернет объединяет миллионы узлов по всему миру. Большая часть не-PC машин подключена к сети Интернет. Некоторые из новых коммерческих информационных магистралей имеют пропускную способность, измеряемую гигабитами в секунду, что в десятки тысяч раз превышает пропускную способность когда-то существовавшей ARPAnet. Ежедневно десятки миллионов людей используют сеть для общения и совместной работы.

Интернет и интернет-сети

Следует сказать пару слов об Интернете и интернет-сетях. В тексте различие между названиями кажется незначительным: одно название всегда пишется с прописной буквы, второе всегда нет. Тем не менее, разница в значении – *существенна*. Интернет с заглавной буквы «И» – обозначение сети, которая началась с ARPAnet и существует сегодня, грубо говоря, как объединение всех TCP/IP-сетей, прямо или косвенно связанных с коммерческими информационными магистралями США. При внимательном рассмотрении – это целый ряд различ-

ных сетей – коммерческих опорных сетей TCP/IP, корпоративных и правительственных сетей США, а также TCP/IP-сетей других стран. Сети объединены маршрутизаторами и высокоскоростными цифровыми каналами.

Начинающийся со строчной буквы интернет – это просто любая сеть, объединяющая несколько сетей масштабом поменьше, причем с использованием все тех же протоколов межсетевого взаимодействия. Сеть интернет не всегда связана с сетью Интернет и не обязательно основана на сетевых протоколах TCP/IP. Существуют изолированные корпоративные интернет-сети, а также сети на основе протоколов Xerox XNS или DECnet.

Относительно новый термин «intranet», по сути дела, – это всего лишь рекламное название для интернет-сетей, которое используется, чтобы подчеркнуть применение технологий, обкатанных в Интернете, в рамках внутренних корпоративных сетей. С другой стороны, extranet-сети – это сети интернет, объединяющие сотрудничающие компании, либо компании с их агентами по продаже, поставщиками и клиентами.

История системы доменных имен

В семидесятых годах сеть ARPAnet представляла собой тесное сообщество из нескольких сотен узлов. Всю жизненно-важную информацию по узлам, в частности, необходимую для взаимных преобразований имен и адресов узлов ARPAnet, содержал единственный файл *HOSTS.TXT*. Известная Unix-таблица узлов, */etc/hosts*, – прямо унаследовала свою структуру от файла *HOSTS.TXT* (в основном с помощью удаления ненужных на Unix-системах полей).

За файл *HOSTS.TXT* отвечал Сетевой информационный центр (NIC, Network Information Center) Стэнфордского исследовательского института (SRI, Stanford Research Insitute). В тот период времени единственным источником, распространявшим файл, являлся узел SRI-NIC.[1] Администраторы ARPAnet, как правило, просто посылали изменения электронной почтой в NIC и периодически синхронизировали свои файлы *HOSTS.TXT* с копией на узле SRI-NIC с помощью протокола FTP. Присылаемые ими изменения добавлялись в файл *HOSTS.TXT* один или два раза в неделю. Однако по мере роста сети ARPAnet эта схема стала неработоспособной. Размер файла рос пропорционально количеству узлов ARPAnet. Еще быстрее рос информационный поток, связанный с необходимостью обновления файла на хостах: появление одного нового узла приводило не только к добавлению строки в *HOSTS.TXT*,

[1] Организация SRI International в настоящее время уже не связана жестко со Стэнфордским исследовательским институтом, расположенном в Менло-Парк (в Калифорнии); она проводит исследования во многих областях, включая и компьютерные сети.

но и к потенциальной необходимости синхронизации данных каждого узла с данными SRI-NIC.

После перехода ARPAnet на протоколы TCP/IP рост сети стал взрывным. Появился гордиев узел проблем, связанных с файлом *HOSTS.TXT*:

Информационные потоки и нагрузка

Нагрузка на SRI-NIC в смысле сетевого трафика и работы процессора, связанных с раздачей файла, приближалась к предельной.

Конфликты имен

Никакие два хоста, описанные в файле *HOSTS.TXT*, не могли носить одинаковые имена. Организация NIC могла контролировать присваивание адресов способом, гарантирующим их уникальность, но не имела никакого влияния на имена узлов. Не было способа предотвратить добавление узла с уже существующим именем, при том, что такое действие нарушало работу всей схемы. Так, добавление узла с именем, идентичным имени одного из крупных почтовых концентраторов, могло привести к нарушению работы почтовых служб большей части сети ARPAnet.

Синхронизация

Синхронизация файлов в масштабах быстро растущей сети становилась все более сложной задачей. К тому моменту, когда обновленный файл *HOSTS.TXT* достигал самых далеких берегов выросшей ARPAnet, адреса отдельных узлов успевали измениться, а также появлялись новые узлы, доступ к которым был необходим пользователям.

Основная проблема заключалась в том, что схема с файлом *HOSTS.TXT* не поддавалась масштабированию. По иронии судьбы, успех ARPAnet как эксперимента вел к моральному устареванию и провалу механизма *HOSTS.TXT*.

Административные советы ARPAnet начали исследование, которое должно было привести к созданию замены *HOSTS.TXT*. Целью его было создание системы, которая решила бы проблемы, присущие сводной таблице узлов. Новая система должна позволять производить локальное управление данными, но делать эти данные доступными всем. Децентрализация администрирования решила бы проблемы с трафиком и нагрузкой, непосильными для единственного узла. Локальное управление данными упростило бы задачу обновления и синхронизации информации. В новой системе следует использовать иерархическое пространство имен для присваивания идентификаторов узлам, что позволит гарантировать уникальность каждого отдельного имени.

Ответственным за разработку архитектуры новой системы стал Пол Мокапетрис, работавший тогда в Институте информационных наук (Information Sciences Institute). В 1984 году он издал документы RFC

882 и 883, в которых описывалась система доменных имен (Domain Name System, или DNS). Эти RFC-документы были обновлены документами RFC 1034 и 1035, которые и являются в настоящее время действующей спецификацией DNS.[1] RFC 1034 и 1035 к настоящему времени дополняются многими другими подобными документами, в которых описаны потенциальные проблемы DNS с точки зрения сетевой безопасности, возможные трудности реализации, проблемы административного плана, механизмы динамического обновления DNS-серверов, обеспечение безопасности зональных данных и многое другое.

Система доменных имен в двух словах

DNS – это *распределенная* база данных. Это свойство дает возможность локально управлять отдельными сегментами общей базы, а также позволяет сделать данные каждого сегмента доступными всей сети – посредством использования механизма клиент-сервер. Надежность и адекватная производительность основаны на репликации и кэшировании.

Серверная часть клиент-серверного механизма DNS представлена программами, которые называются *DNS-серверами* (*name servers*, дословно – серверами имен). DNS-серверы владеют информацией о некоторых сегментах базы данных и делают ее доступной клиентам, которые называются *поисковыми анализаторами* (*resolvers*).[2] Как правило, DNS-клиент – это просто набор библиотечных функций, которые создают запросы и посылают их по сети серверу имен.

Структура базы данных DNS очень похожа на структуру файловой системы Unix (рис. 1.1). Вся база данных (или файловая система) представлена в виде перевернутого дерева, корень (корневой узел) которого расположен на самом верху. Каждый узел дерева имеет прикрепленную текстовую метку, которая идентифицирует его относительно родительского узла, по аналогии с «относительным путевым именем» в файловой системе (например, *bin*). Одна из меток, пустая, (она обозначается как “”) закреплена за корневым узлом дерева. В тексте корневой узел обозначается точкой (.). В файловой системе Unix корень обозначается символом «слэш» (/).

Каждый узел является корнем новой ветви дерева. Каждая из ветвей (поддеревьев) является разделом базы данных – «каталогом» в ин-

---

[1] Документы RFC (Request for Comments, запрос комментариев) являются частью относительно неформальной процедуры введения новых технологий в сети Интернет. RFC-документы обычно свободно распространяются и содержат описания технического плана технологий, предназначенные, во многих случаях, для разработчиков.

[2] Далее в тексте будет использоваться как термин «поисковый анализатор», так и «клиентская часть DNS», или просто «DNS-клиент», в зависимости от контекста. – *Примеч. науч. ред.*

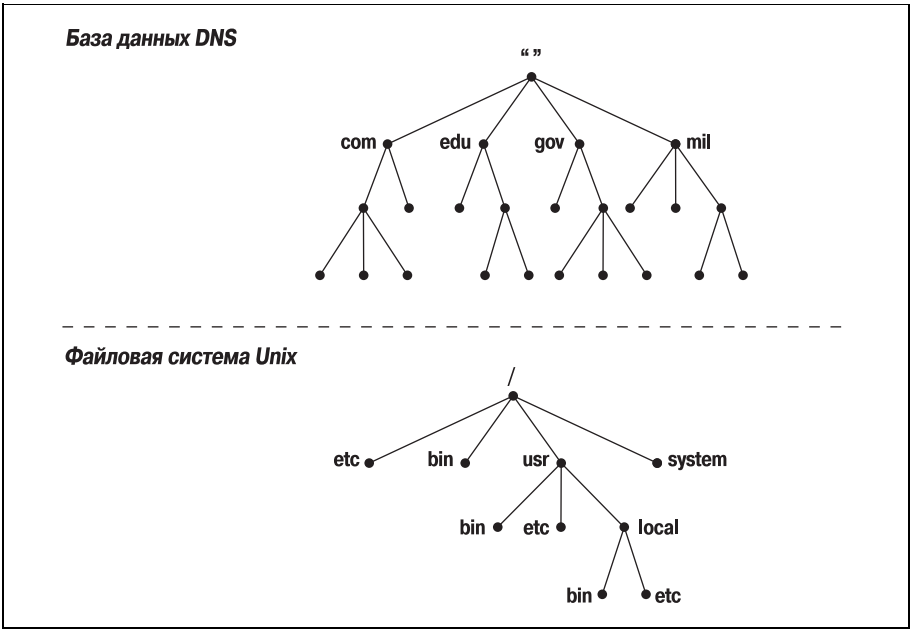

Рис. 1.1. *База данных DNS и файловая система Unix*

терпретации файловой системы Unix, или *доменом* в интерпретации системы доменных имен. Каждый домен или каталог может быть разбит на еще более мелкие подразделы, которые в DNS называются *поддоменами*, а в файловых системах – «подкаталогами». Поддомены, как и подкаталоги, изображаются как потомки соответствующих родительских доменов.

Имя домена, как и имя любого каталога, уникально. *Имя домена* определяет его расположение в базе данных, так же, как «абсолютный путь к каталогу» однозначно определяет его расположение в файловой системе. Имя домена в DNS – это последовательность меток от узла, корневого для данного домена, до корня всего дерева; метки в имени домена разделяются точками. В файловой системе Unix абсолютное путевое имя каталога – это последовательность относительных имен, начиная от корня дерева до конкретного узла (то есть чтение происходит в направлении, противоположном направлению чтения имен DNS; рис. 1.2), при этом имена разделяются символом «прямая наклонная черта» («слэш»).

В DNS каждый домен может быть разбит на поддомены, и ответственность за эти поддомены может распределяться между различными организациями. Допустим, компания Network Solutions сопровождает домен *edu* (*educational*, то есть образовательный), но делегирует ответственность за поддомен *berkeley.edu* Калифорнийскому университету Беркли (рис. 1.3). это похоже на удаленное монтирование файловой системы: определенные каталоги файловой системы могут в действи-

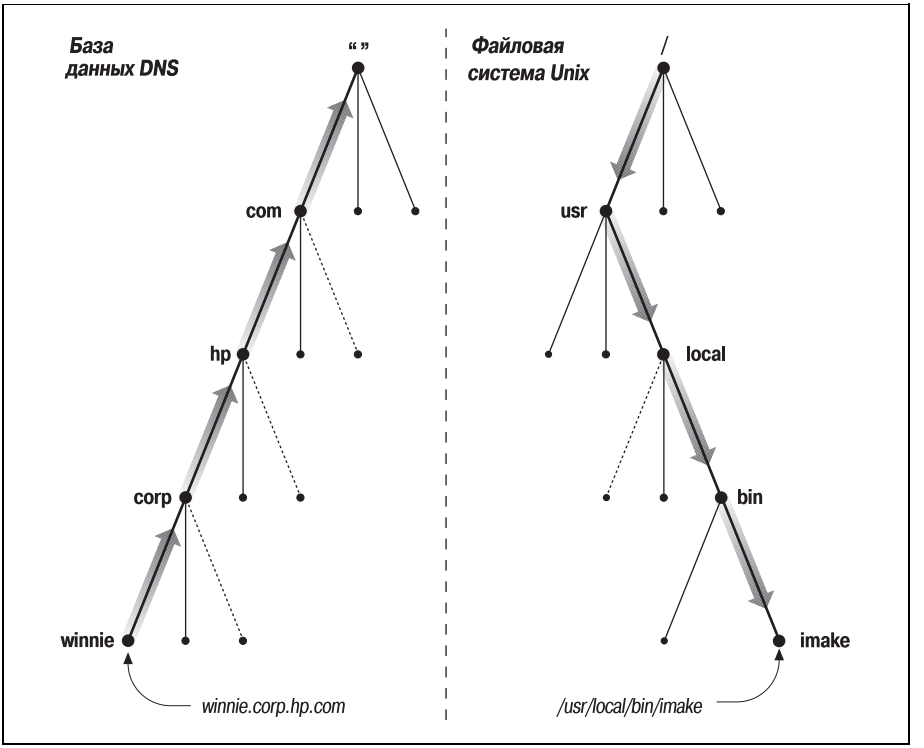

Рис. 1.2. *Чтение имен DNS и файловой системы Unix*

тельности являться файловыми системами, расположенными на других узлах и смонтированными удаленно. К примеру, администратор узла *winken* (рис. 1.3) отвечает за файловую систему, которая на локальном узле выглядит как содержимое каталога */usr/nfs/winken*.

Делегирование управления поддоменом *berkeley.edu* Калифорнийскому университету Беркли приводит к созданию новой *зоны* – независимо администрируемой части пространства имен. Зона *berkeley.edu* теперь не зависит от *edu* и содержит все доменные имена, которые заканчиваются на *berkeley.edu*. С другой стороны, зона *edu* содержит только доменные имена, оканчивающиеся на *edu*, но не входящие в делегированные зоны, такие, например, как *berkeley.edu*. *berkeley.edu* может быть поделен на поддомены с именами вроде *cs.berkeley.edu*, и некоторые из этих поддоменов могут быть выделены в самостоятельные зоны, если администраторы *berkeley.edu* делегируют ответственность за них другим организациям. Если *cs.berkeley.edu* является самостоятельной зоной, зона *berkeley.edu* не содержит доменные имена, которые заканчиваются на *cs.berkeley.edu* (рис. 1.4).

Доменные имена используются в качестве индексов базы данных DNS. Данные DNS можно считать «привязанными» к доменному имени. В файловой системе каталоги содержат файлы и подкаталоги. Анало-

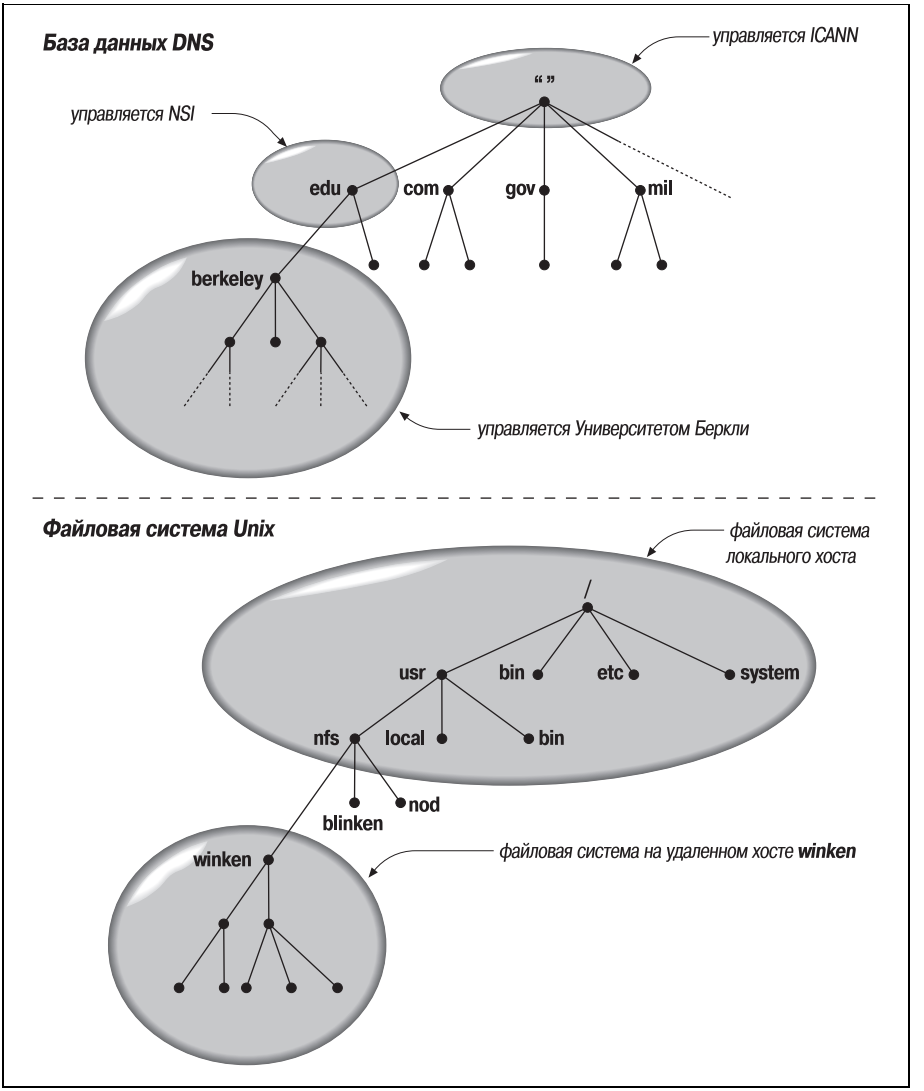

Рис. 1.3. *Удаленное управление доменами разных уровней и файловыми системами*

гичным образом домены могут содержать узлы и поддомены. Домен включает в себя те узлы и поддомены, доменные имена которых принадлежат этому домену.

У каждого узла в сети есть доменное имя, которое является указателем на информацию об узле (рис. 1.5). Эта информация может включать IP-адрес, информацию о маршрутизации почтовых сообщений и другие данные. Узел может также иметь один или несколько *псевдонимов доменного имени*, которые являются просто указателями на ос-

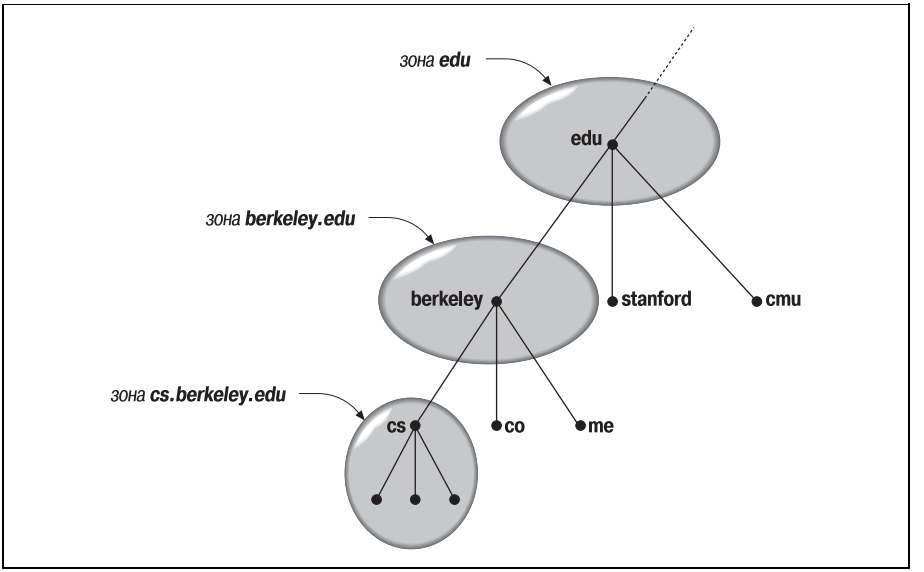

Рис. 1.4. *Зоны edu, berkeley.edu и cs.berkeley.edu*

новное (официальное, или *каноническое*) доменное имя. На рис. 1.5 *mailhub.nv...* является псевдонимом канонического имени *rincon.ba.ca...*

Для чего нужна столь сложная структура? Чтобы решить проблемы, существовавшие при использовании *HOSTS.TXT*. К примеру, строгая иерархичность доменных имен устраняет угрозу конфликтов имен.

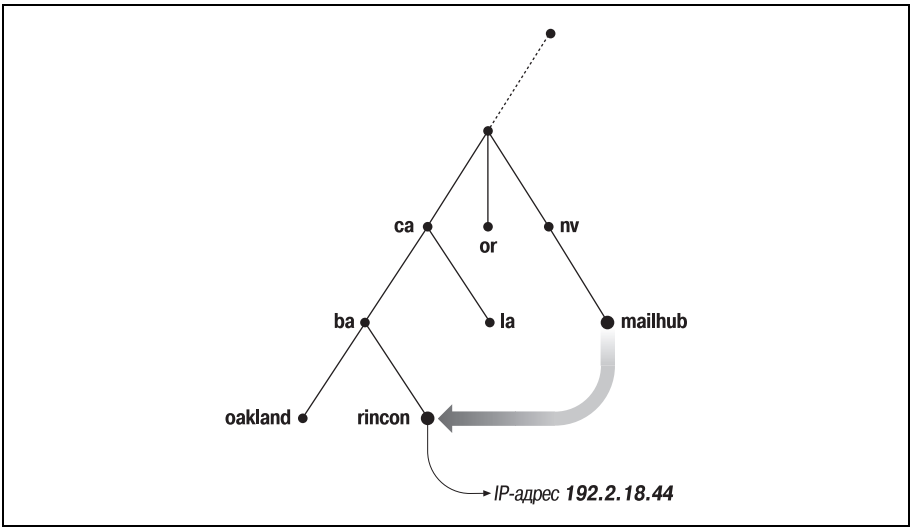

Рис. 1.5. *Псевдоним в DNS, ссылающийся на каноническое имя*

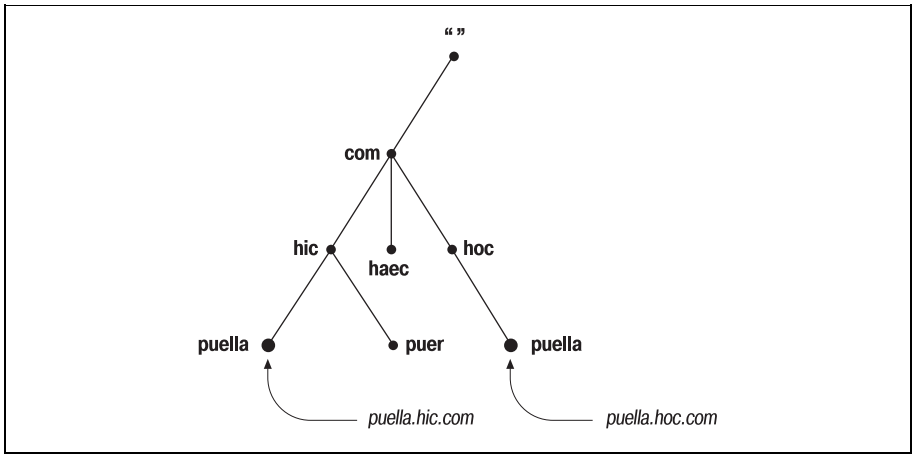

Рис. 1.6. *Решение проблемы конфликтов имен*

Имя каждого домена уникально, так что организация, управляющая доменом, вольна придумывать имена поддоменов, входящих в этот домен, самостоятельно. Независимо от выбранных имен, имена эти не будут конфликтовать с другими доменными именами, поскольку заканчиваются уникальным именем домена, сопровождаемого только этой организацией. Так, организация, ответственная за домен *hic.com* может дать узлу имя *puella* (рис. 1.6), поскольку известно, что доменное имя узла будет заканчиваться уникальным доменным именем *hic.com*.

История пакета BIND

Первая реализация системы доменных имен называлась JEEVES, и была разработана самим Полом Мокапетрисом. Более поздняя реализация носит название BIND, это аббревиатура от *Berkeley Internet Name Domain*, и была разработана для операционной системы 4.3 BSD Unix (Беркли) Кевином Данлэпом. В настоящее время развитием и сопровождением пакета BIND занимается Internet Software Consortium.[1]

На пакете BIND мы и сосредоточимся в данной книге, поскольку именно BIND является сегодня наиболее популярной и распространенной реализацией DNS. Пакет доступен на большинстве разновидностей системы Unix и входит в стандартную конфигурацию систем от большинства поставщиков Unix. BIND также был портирован на платформу Microsoft Windows NT.

---

[1] Более подробно об организации Internet Software Consortium и ее разработках в области BIND можно узнать по адресу *http://www.isc.org/bind.html*

Надо ли мне использовать DNS?

Несмотря на очевидную пользу DNS, существуют случаи, в которых ее применение неоправданно. Помимо DNS существуют и другие механизмы разрешения имен, некоторые из которых могут быть составной частью операционной системы. Иногда затраты сил и времени, связанные с сопровождением зон и DNS-серверов, превышают все возможные выгоды. С другой стороны, возможна ситуация, когда нет другого выбора, кроме как установить и поддерживать DNS-серверы. Вот некоторые указания, которые помогут сориентироваться и принять решение:

Если вы подключены к сети Интернет...

...DNS является жизненной необходимостью. DNS можно считать общепринятым языком сети Интернет: почти все сетевые службы в Интернете, включая World Wide Web, электронную почту, удаленный терминальный доступ и передачу файлов, используют DNS.

С другой стороны, подключение к сети Интернет вовсе не означает, что не удастся избежать самостоятельной установки и сопровождения нужных пользователю зон. В случае ограниченного числа узлов всегда можно найти уже существующую зону и стать ее частью (подробнее – в главе 3 «С чего начать?»). Как вариант, можно найти кого-то, кто позаботится о размещении зоны. Если пользователь платит интернет-провайдеру за подключение, обычно существует возможность разместить свою зону на технологических мощностях этого провайдера. Существуют также компании, предоставляющие подобную услугу за отдельную плату.

Если же узлов много или очень много, то, скорее всего, понадобится самостоятельная зона. Если вы хотите иметь непосредственный контроль над зоной и серверами имен, то вступаете на путь администрирования и сопровождения. Читайте дальше!

Если у вас интернет-сеть на основе протоколов TCP/IP...

...то DNS, вероятно, не помешает. В данном случае под интернет-сетью мы не подразумеваем простую сеть из одного сегмента Ethernet и нескольких рабочих станций, построенную на протоколах TCP/IP (такой вариант описан в следующем разделе), но достаточно сложную «сеть сетей». Например – множество Appletalk-сегментов плюс несколько сетей Apollo token ring, объединенных вместе.

Если интернет-сеть является преимущественно гомогенной, и узлы не нуждаются в службе DNS (скажем, в случае крупной интернет-сети DECnet или OSI), вполне возможно, что можно обойтись без нее. Но в случае разнородных хостов, в особенности, если некоторые из них работают под управлением Unix, DNS пригодится. Система упростит распространение информации об узлах и избавит ад-

министратора от необходимости выдумывания своей схемы распространения таблиц хостов.

Если у вас собственная локальная сеть...

...и эта сеть не соединена с большей сетью, вполне возможно обойтись без DNS. Можно попробовать использовать службу Windows Internet Name Service (WINS) от Microsoft, таблицы хостов, или Network Information Service (NIS) от Sun.

В случаях, когда требуется распределенное администрирование, либо присутствуют сложности с синхронизацией данных в сети, использование DNS может иметь смысл. И если планируется подключение вашей сети к другой, скажем, к корпоративной интернет-сети, либо к Интернету, стоит заранее заняться настройкой собственных зон.

2

- *Пространство доменных имен*
- *Пространство доменных имен сети Интернет*
- *Делегирование*
- *DNS-серверы и зоны*
- *DNS-клиенты*
- *Разрешение имен*
- *Кэширование*

Как работает DNS

– Что толку в книжке, – подумала Алиса, – если в ней нет ни картинок, ни разговоров?

Система доменных имен – это, прежде всего, база данных, содержащая информацию об узлах сети. Да, вкупе с этим вы получаете целый набор всякой всячины: чудные имена с точками, серверы, которые подключаются к сети, загадочное «пространство имен». И все же следует помнить, что, в конечном итоге, услуга, предоставляемая DNS, сводится к получению информации об узлах сети.

Мы уже рассмотрели некоторые важные аспекты работы DNS, включая архитектуру «клиент-сервер» и структуру базы данных. Однако мы не особенно вдавались в детали и не объясняли работу основных механизмов DNS.

В этой главе мы объясним и проиллюстрируем процессы, на которых построена работа системы доменных имен. Будет представлена терминология, которая позволит прочесть и понять оставшуюся часть книги (а также вести интеллектуальные беседы с друзьями – администраторами DNS).

Но сначала все-таки взглянем чуть внимательнее на концепции, представленные в предшествующей главе. Попробуем углубиться в детали и придать им особый ракурс.

Пространство доменных имен

Распределенная база данных системы доменных имен индексируется по именам узлов. Каждое доменное имя является просто путем в ог-

ромном перевернутом дереве, которое носит название *пространства доменных имен*. Иерархическая структура дерева, отображенная на рис. 2.1, похожа на структуру файловой системы Unix. Единственный корень дерева расположен наверху.[1] В файловых системах Unix эта точка называется корневым каталогом и представлена символом «слэш» (/). В DNS же это просто «корень» («root»). Как и файловая система, дерево DNS может иметь любое количество ответвлений в любой точке пересечения, или *узле*. Глубина дерева ограничена и может достигать 127 уровней (предел, до которого вы вряд ли когда-нибудь доберетесь).

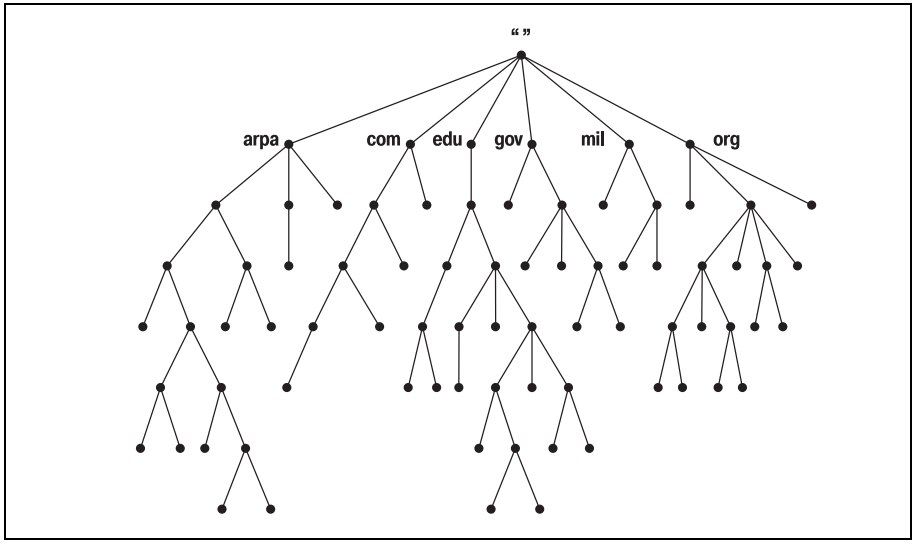

Рис. 2.1. *Структура пространства имен DNS*

Доменные имена

Каждому узлу дерева соответствует текстовая метка, длина которой не может превышать 63 символов, причем использование символа точки недопустимо. Пустая (нулевой длины) метка зарезервирована для корня. Полное *доменное имя* произвольного узла дерева – это последовательность меток в пути от этого узла до корня. Доменные имена всегда читаются от собственно узла к корню («вверх» по дереву), причем метки разделяются точкой.

Если метка корневого узла должна быть отображена в доменном имени, она записывается как символ точки, например, так: «www.oreilly.com.». (На самом деле имя заканчивается точкой-разделителем и пустой меткой корневого узла.) Сама по себе метка корневого узла за-

[1] Понятно, что это дерево компьютерщика, а не ботаника.

писывается исключительно из соображений удобства, как самостоятельная точка (.). Как следствие, некоторые программы интерпретируют имена доменов, заканчивающиеся точкой, как *абсолютные*. Абсолютное доменное имя записывается относительно корня и однозначно определяет расположение узла в иерархии. Абсолютное доменное имя известно также под названием *полного доменного имени*, обозначаемого аббревиатурой *FQDN* (*fully qualified domain name*). Имена без завершающей точки иногда интерпретируются относительно некоторого доменного имени (не обязательно корневого) точно так же, как имена каталогов, не начинающиеся с символа «/» (слэш), часто интерпретируются относительно текущего каталога.

В DNS «братские» узлы, то есть узлы, имеющие общего родителя, должны иметь разные метки. Такое ограничение гарантирует, что доменное имя единственно возможным образом идентифицирует отдельный узел дерева. Это ограничение на практике не является ограничением, поскольку метки должны быть уникальными только для братских узлов одного уровня, но не для всех узлов дерева. То же ограничение существует в файловых системах Unix: двум «единоутробным» каталогам или двум файлам в одном каталоге не могут быть присвоены одинаковые имена. Невозможно создать два узла *hobbes.pa.ca.us* в пространстве доменных имен, и невозможно создать два каталога */usr/bin* (рис. 2.2). Тем не менее, можно создать пару узлов с именами *hobbes.pa.ca.us* и *hobbes.lg.ca.us*, точно так же, как можно создать пару каталогов с именами */bin* и */usr/bin*.

Домены

Домен – это просто поддерево в пространстве доменных имен. Доменное имя домена идентично доменному имени узла на вершине домена. Так, к примеру, вершиной домена *purdue.edu* является узел с именем *purdue.edu* (рис. 2.3).

Аналогичным образом, в корне файловой системы */usr* мы ожидаем увидеть каталог с именем */usr* (рис. 2.4).

Каждое доменное имя в поддереве считается принадлежащим домену. Поскольку доменное имя может входить в несколько поддеревьев, оно также может входить в несколько доменов. К примеру, доменное имя *pa.ca.us* входит в домен *ca.us* и при этом является также частью домена *us* (рис 2.5).

Так что теоретически домен – это просто сегмент пространства доменных имен. Но если домен состоит только из доменных имен и других доменов, где узлы сети – хосты? Ведь домены-то – это группы хостов, верно?

Узлы сети, разумеется, присутствуют, и представлены они доменными именами. Следует помнить, что доменные имена являются просто указателями в базе данных DNS. «Хосты» – это доменные имена, которые указывают на информацию по каждому конкретному хосту. Домен со-

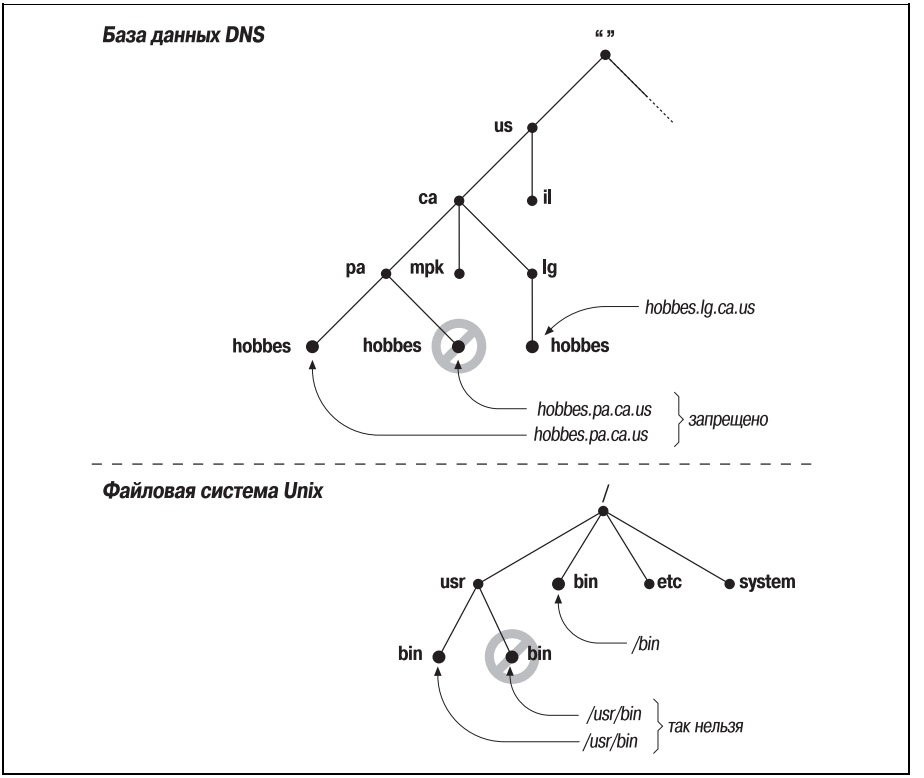

Рис. 2.2. *Обеспечение уникальности доменных имен и путевых имен в файловых системах Unix*

держит все узлы сети, доменные имена которых в него входят. Узлы сети связаны *логически*, зачастую по географическому или организационному признаку, и совсем необязательно – сетью, адресом или типом используемого оборудования. Десяток узлов, входящих в разные сети, возможно, даже расположенных в разных странах, может принадлежать одному-единственному домену.[1]

Доменные имена, соответствующие листьям дерева, как правило, относятся к отдельным узлам сети и могут указывать на сетевые адреса,

[1] Предостережение: не стоит путать домены DNS с доменами в службе NIS, Network Information Service от Sun. Несмотря на то, что домен NIS – это тоже группа узлов, а доменные имена в обеих службах имеют сходную организацию, концептуальные различия достаточно велики. В NIS используется иерархическая организация имен, но иерархия на этом и кончается: узлы сети, входящие в один домен NIS, разделяют определенную информацию об узлах и пользователях, но не могут опрашивать пространство имен NIS с целью поиска информации в других доменах NIS. Домены NT, обеспечивающие управление доступом и службами безопасности, также не имеют никакого отношения к доменам DNS.

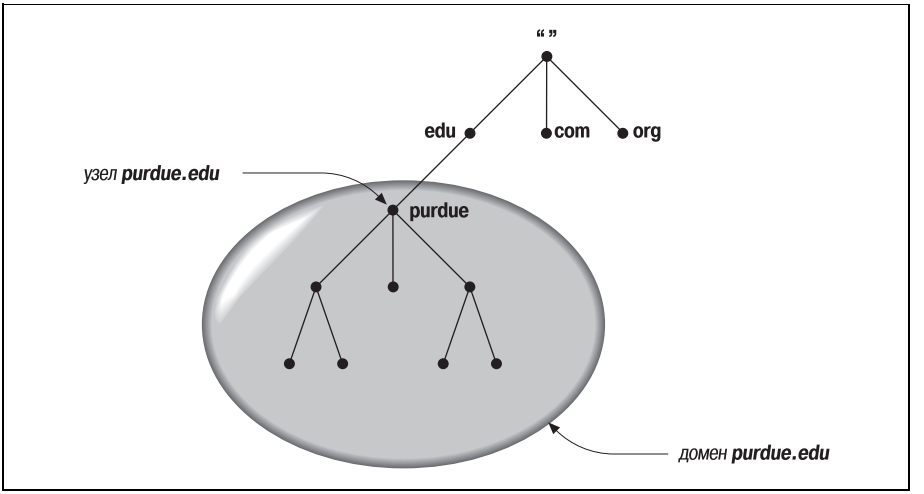

Рис. 2.3. *Домен purdue.edu*

информацию об оборудовании и о маршрутизации почты. Доменные имена внутри дерева могут идентифицировать отдельные узлы, *а также* могут указывать на информацию об этом домене. Имена доменов внутри дерева не привязаны жестко к тому или другому варианту. Они могут представлять как домен (имени которого соответствуют), так и отдельный узел в сети. Так, *hp.com* является именем домена компании Hewlett-Packard и доменным именем узлов, на которых расположен главный веб-сервер Hewlett-Packard.

Тип информации, получаемой при использовании доменного имени, зависит от контекста применения имени. Посылка почтовых сообщений кому-то в домен *hp.com* приводит к получению информации о маршру-

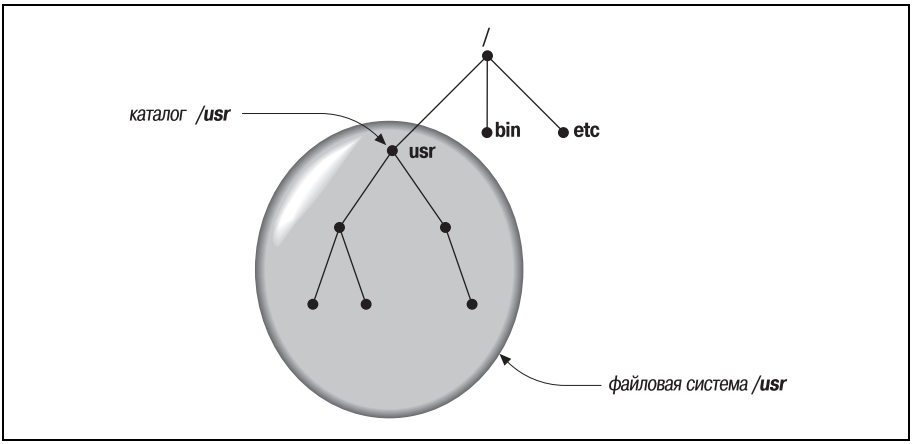

Рис. 2.4. *Каталог /usr*

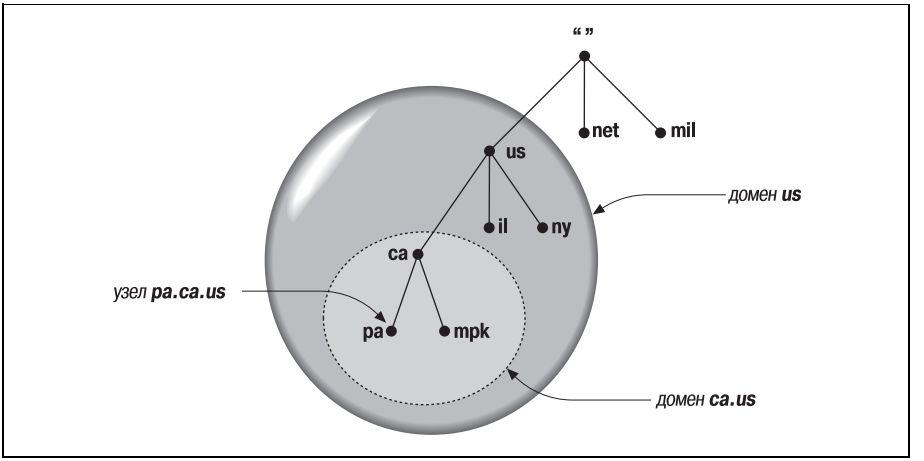

Рис. 2.5. *Узел, входящий в несколько доменов*

тизации почты, открытие telnet-сеанса связи с этим доменом приводит к поиску информации об узле (на рис. 2.6, к примеру, это IP-адрес узла *hp.com*).

Домен может содержать несколько поддеревьев, которые носят название *поддоменов*.[1]

Самый простой способ выяснить, является ли домен поддоменом другого домена, – сравнить их доменные имена. Доменное имя поддомена заканчивается доменным именем родительского домена. К примеру,

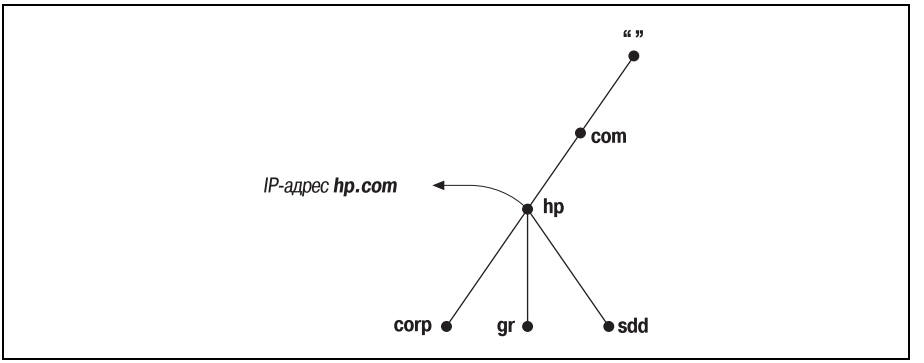

Рис. 2.6. *Внутренний узел дерева, связанный как с информацией о конкретном узле сети, так и с иерархической*

[1] Термины *домен* и *поддомен* в документации по DNS и BIND зачастую используются в качестве взаимозаменяемых. В настоящей книге мы используем термин «поддомен» в качестве относительного: домен является поддоменом другого домена, если корень поддомена принадлежит включающему домену.

домен *la.tyrell.com* должен являться поддоменом домена *tyrell.com*, поскольку имя *la.tyrell.com* заканчивается именем *tyrell.com*. Аналогичным образом этот домен является поддоменом домена *com*, как и домен *tyrell.com*.

Помимо относительных степеней в идентификации доменов в качестве входящих в другие домены, используется также *уровневая* классификация доменов. В списках рассылки и конференциях Usenet можно встретить термины *домен высшего уровня* и *домен второго уровня*. Эти термины просто определяют положение домена в пространстве доменных имен:

- Домен высшего уровня в качестве непосредственного родителя имеет корневой домен.
- Домен первого уровня в качестве непосредственного родителя (также) имеет корневой домен (то есть он является доменом высшего уровня).
- Домен второго уровня в качестве непосредственного родителя имеет домен первого уровня, и т. д.

Записи ресурсов

Информация, связанная с доменными именами содержится в *записях ресурсов* (RRs, resource records).[1] Записи разделяются на классы, каждый из которых определяет тип сети или программного обеспечения. В настоящее время существуют классы для интернет-сетей (на основе семейства протоколов TCP/IP), сетей на основе проколов Chaosnet, а также сетей, которые построены на основе программного обеспечения Hesiod. (Chaosnet – старая сеть, имеющая преимущественно историческое значение).

Популярность класса интернет-сетей значительно превосходит популярность остальных классов. (Мы не уверены, что где-то до сих пор используется класс Chaosnet, а использование класса Hesiod в основном ограничивается пределами Массачусетского технологического института – MIT). В настоящей книге мы сосредоточимся на классе интернет-сетей.

В пределах класса записи делятся на типы, которые соответствуют различным видам данных, хранимых в пространстве доменных имен. В различных классах определяются различные типы записей, хотя некоторые типы могут являться общими для нескольких классов. Так, практически в каждом классе определен тип *адрес*. Каждый тип записи в конкретном классе определяет формат, который должны соблюдать все RR-записи, принадлежащие этому классу и имеющие данный

[1] В дальнейшем мы будем использовать выражение RR-запись, или просто RR. – *Примеч. науч. ред.*

тип. (Подробно типы и форматы RR-записей для интернет-сетей описаны в приложении А «Формат сообщений DNS и RR-записей».)

Не беспокойтесь, если информация кажется излишне схематичной: записи класса интернет будут более подробно рассмотрены позже. Наиболее часто используемые RR-записи описаны в главе 4 «Установка BIND», а более полный перечень приводится в приложении А.

Пространство доменных имен сети Интернет

До сих пор мы говорили о теоретической структуре пространства доменных имен и о том, какого сорта данные могут в нем содержаться, и даже обозначили – в наших (порой выдуманных) примерах – типы имен, которые можно встретить в этом пространстве. Но это ничем не поможет в расшифровке доменных имен, которые ежедневно встречаются пользователям в сети Интернет.

Система доменных имен довольно либеральна в отношении меток, составляющих доменные имена, и не определяет *конкретные* значения для меток определенного уровня. Если администратор управляет сегментом пространства имен, он и определяет семантику доменных имен, входящих в сегмент. Черт возьми, администратор может присвоить поддоменам в качестве имен буквы алфавита от A до Z, и никто его не остановит (хотя сильно будут советовать этого не делать).

При этом существующее пространство доменных имен сети Интернет обладает некоторой сложившейся структурой. Это в особенности касается доменов верхних уровней, доменные имена в которых подчиняются определенным традициям (которые не являются правилами, поскольку их можно нарушать, и так не раз бывало). Эти традиции вносят в доменные имена некоторую упорядоченность. Понимание традиций – это большое подспорье для попыток расшифровки доменных имен.

Домены высшего уровня

Изначально домены высшего уровня делили пространство имен сети Интернет на семь доменов:

com

Коммерческие организации, такие как Hewlett-Packard (*hp.com*), Sun Microsystems (*sun.com*) или IBM (*ibm.com*).

edu

Образовательные организации, такие как Калифорнийский университет Беркли (*berkeley.edu*) и университет Пердью (*purdue.edu*).

gov

Правительственные организации, такие как NASA (*nasa.gov*) и Национальный научный фонд (*nsf.gov*).

mil

Военные организации, такие как армия (*army.mil*) и флот (*navy.mil*) США.

net

В прошлом организации, обеспечивающие работу сетевой инфраструктуры, такие как NSFNET (*nsf.net*) и UUNET (*uu.net*). В 1996 году домен *net*, как и *com*, стал доступен для всех коммерческих организаций.

org

В прошлом некоммерческие организации, такие как Фонд электронной границы (Electronic Frontier Foundation) (*eff.org*). Как и в случае с доменом *net*, ограничения были сняты в 1996 году.

int

Международные организации, такие как NATO (*nato.int*).

Существовал еще один домен высшего уровня, который назывался *arpa* и использовался в процессе перехода сети ARPAnet от таблиц узлов к системе доменных имен. Изначально все узлы ARPAnet имели доменные имена, принадлежавшие домену *arpa*, так что их было несложно отличать. Позже они разбрелись по различным поддоменам организационных доменов высшего уровня. При этом домен *arpa* все еще используется, и чуть позже мы расскажем, как именно.

Можно заметить некоторую националистическую предрасположенность в примерах – прежде всего речь идет об организациях США. Это легче понять – и простить – если вспомнить, что сеть Интернет началась с сети ARPAnet, исследовательского проекта, который финансировался правительством США. Никто и не предполагал, что создание ARPAnet увенчается подобным успехом и что этот успех в итоге приведет к созданию международной сети Интернет.

В наши дни эти домены называются *родовыми доменами высшего уровня* (generic top-level domains или gTLDs). В начале 2001 года их список расширится, и будет включать домены *name*, *biz*, *info*, а также *pro*, которые создаются, чтобы приспособить иерархию к взрывному росту сети Интернет и удовлетворить потребность в доменном «пространстве». Организация, ответственная за управление системой доменных имен сети Интернет, – Internet Corporation for Assigned Names and Numbers (ICANN) – утвердила добавление новых gTLD, а также явно конкретизированных *aero*, *coop* и *museum* в конце 2000 года. Информация о работе организации ICANN и новых доменах высшего уровня доступна по адресу *http://www.icann.org*.

Чтобы справиться с потребностями быстро растущих сегментов сети Интернет, расположенных во многих странах мира, пришлось пойти на определенные компромиссы, связанные с пространством доменных имен Интернета. Было решено не придерживаться схемы организаци-

онного деления доменов высшего уровня, но разрешить использование географических обозначений. Новые домены высшего уровня были зарезервированы (но не во всех случаях созданы) для каждой страны. Эти доменные суффиксы соответствуют существующему международному стандарту ISO 3166.[1] Стандарт ISO 3166 определяет официальные двухбуквенные сокращения для каждой страны мира. Текущий список доменов высшего уровня приведен в приложении D «Домены высшего уровня».

По нисходящей

В пределах упомянутых доменов верхних уровней существующие традиции и степень их соблюдения начинает варьироваться. Некоторые из доменов высшего уровня, упомянутых в стандарте ISO 3166, в общем и целом следуют исходной организационной схеме США. К примеру, в домен высшего уровня Австралии, *au*, входят такие поддомены, как *edu.au* и *com.au*. Некоторые из прочих доменов ISO 3166 следуют примеру домена *uk* и порождают поддомены организационного деления, например *co.uk*, для корпоративного использования, а скажем *ac.uk* – для нужд академического сообщества. Тем не менее в большинстве случаев географические домены высшего уровня имеют организационное деление.

Однако это не относится к домену высшего уровня *us*. В домен *us* входит 50 поддоменов, которые соответствуют – попробуйте угадать! – пятидесяти штатам.[2] Имя каждого из этих поддоменов соответствует стандартному двухбуквенному сокращению названия штата, именно эти сокращения стандартизированы почтовой службой США. В пределах каждого поддомена деление в основном такое же, географическое: большинство поддоменов соответствуют отдельным городам. В пределах поддомена города все прочие поддомены обычно соответствуют отдельным узлам.

Чтение доменных имен

Теперь, познакомившись со свойствами имен доменов высшего уровня и структурой пространства доменных имен, читатели, вероятно, смогут гораздо быстрее определять кроющийся в именах доменов смысл. Попрактикуемся на следующих примерах:

---

[1] За исключением Великобритании. В соответствии со стандартом ISO 3166 и традициями Интернета доменный суффикс высшего уровня для Великобритании должен быть *gb*. На деле же большинство организаций Великобритании и Северной Ирландии (то есть Соединенного Королевства) используют суффикс *uk*. А еще они ездят по неправильной стороне дороги.

[2] В действительности поддоменов в домене *us* чуть больше: один для Вашингтона (округ Колумбия), один для острова Гуам, и т. д.

lithium.cchem.berkeley.edu

Здесь у читателей фора, поскольку мы уже упоминали, что *berkeley.edu* – это домен университета Беркли. (Даже если бы мы не рассказали заранее, можно было бы догадаться, что имя, вероятнее всего, принадлежит одному из университетов США, поскольку принадлежит домену высшего уровня *edu*.) *cchem* – поддомен *berkeley.edu*, принадлежащий факультету химии. И наконец, *lithium* (литий) – имя конкретного узла в домене, помимо которого в поддомене есть, вероятно, еще около ста узлов, если под каждый химический элемент выделен отдельный узел.

winnie.corp.hp.com

Этот пример посложнее, но ненамного. Домен *hp.com*, по всей видимости, принадлежит компании Hewlett-Packard (кстати, и об этом мы уже говорили). Поддомен *corp,* вне всякого сомнения, подразумевает корпоративную штаб-квартиру. А *winnie* – вероятно, просто неудачно выбранное имя узла.

fernwood.mpk.ca.us

В этом примере придется использовать знания о структуре домена *us*. *ca.us* – очевидно, домен штата Калифорния, но *mpk* может означать что угодно. В данном случае очень трудно догадаться, что это доменное имя Менло-Парка, если не знать географию района Сан-Франциско. (Нет, это не тот же самый Менло-Парк, в котором жил Эдисон, тот находится в Нью-Джерси.)

daphne.ch.apollo.hp.com

Этот пример мы приводим для того, чтобы у читателей не появилось ложного ощущения, будто все доменные имена состоят из четырех меток. *apollo.hp.com* – бывший поддомен компании Apollo Computer, принадлежащий домену *hp.com*. (Когда компания HP приобрела компанию Apollo, то не забыла купить и интернет-домен Apollo, *apollo.com*, который позже был переименован в *apollo.hp.com*.) *ch.apollo.hp.com* – отделение Apollo в Челмсфорде (штат Массачусетс). *daphne* – просто узел в Челмсфорде.

Делегирование

Надеемся, читатели еще не забыли, что одной из основных целей разработки системы доменных имен была децентрализация администрирования? Эта цель достигается путем *делегирования*. Делегирование доменов во многом схоже с распределением рабочих задач. Начальник может разделить крупный проект на небольшие задачи и делегировать ответственность за каждую из них разным подчиненным.

Аналогичным образом организация, сопровождающая домен, может разделить его на несколько поддоменов, каждый из которых может

быть *делегирован* какой-либо другой организации. Организация, получившая домен таким путем, несет ответственность за работу с данными этого поддомена. Она может свободно изменять эти данные, разделять поддомен на несколько более мелких поддоменов, а те, в свою очередь, снова делегировать. Родительский домен сохраняет только указатели на источники данных для поддоменов, в целях перенаправления запросов по соответствующим адресам. К примеру, домен *stanford.edu*, делегирован тем ребятам в Стэнфорде, которые управляют университетскими сетями (рис. 2.7.)

Не все организации делегируют домены целиком, как не каждый начальник делегирует всю свою работу. Домен может иметь несколько делегированных поддоменов, но также включать узлы, которые не принадлежат этим поддоменам. К примеру, корпорация Acme (которая снабжает одного известного койота различными приспособлениями) имеет отделение в городе Рокавей, а ее штаб-квартира расположена в Каламазу, поэтому могут существовать поддомены *rockaway.acme.com* и *kalamazoo.acme.com*. Однако несколько узлов, соответствующих отделам сбыта Acme, разбросаны по всей стране и скорее принадлежат собственно домену *acme.com*, нежели какому-либо из поддоменов.[1]

Позже мы расскажем, как создавать и делегировать поддомены. Сейчас же важно понять, что термин *делегирование* относится к передаче ответственности за поддомен сторонней организации.

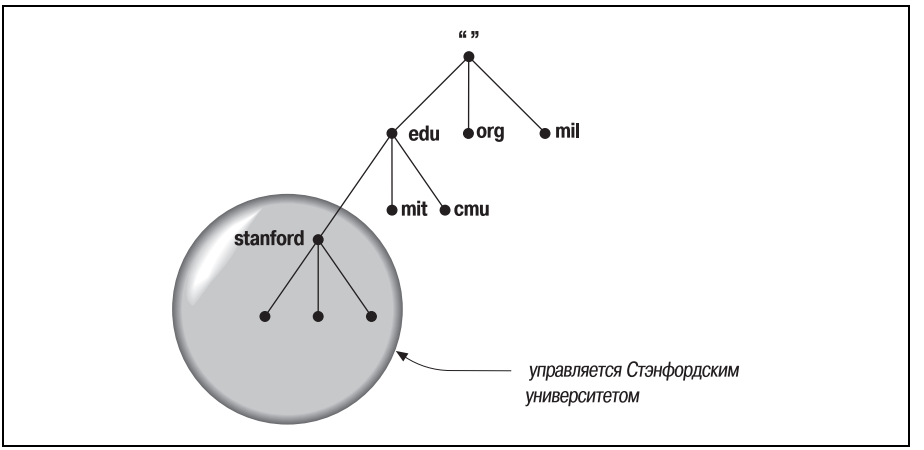

Рис. 2.7. *stanford.edu делегирован Стэнфордскому университету*

[1] «ACME Co.» – не существующая в действительности корпорация из мультипликационного сериала «Bugs Bunny & Roadrunner». Домен *acme.com* принадлежит, на самом деле, организации, разрабатывающей программное обеспечение для Unix-систем. – *Примеч. ред.*

DNS-серверы и зоны

Программы, владеющие информацией о пространстве доменных имен, называются *DNS-серверами*. DNS-серверы обычно обладают полной информацией по определенным сегментам (или *зонам*) пространства доменных имен, которая загружается из файла либо может быть получена от других серверов имен. В таких случаях говорят, что сервер имен является *авторитативным* для конкретной зоны. DNS-серверы могут быть авторитативными также и для нескольких зон.

Разница между зоной и доменом важна, но не очень заметна. Все домены высшего уровня и домены уровней от второго и ниже, такие как *berkeley.edu* и *hp.com*, разбиваются на более мелкие, легко управляемые единицы, путем делегирования. Эти единицы называются зонами. Домен *edu* (рис. 2.8) разделен на много зон, включая зону *berkeley.edu*, зону *purdue.edu* и зону *nwu.edu*. На вершине домена существует также зона *edu*. Естественно, что ребята, управляющие *edu*, разбивают домен *edu* на более мелкие единицы: в противном случае им пришлось бы самим сопровождать поддомен *berkeley.edu*. Гораздо более разумно делегировать *berkeley.edu* – Беркли. Что же остается тем, кто управляет *edu*? Зона *edu*, которая содержит преимущественно информацию о делегировании для поддоменов, входящих в *edu*.

Поддомен *berkeley.edu*, в свою очередь, разбивается на несколько зон путем делегирования (рис. 2.9). Делегированные поддомены носят име-

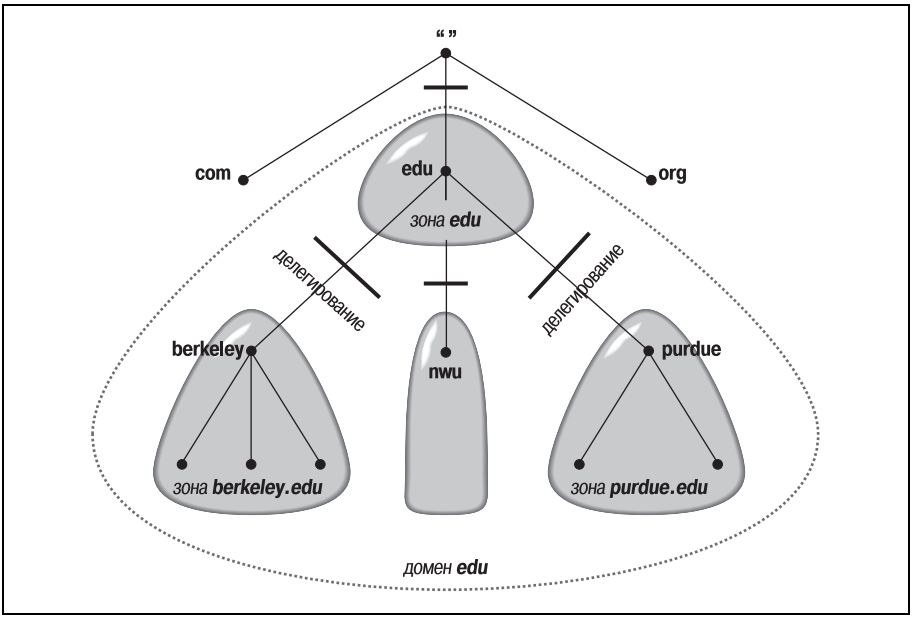

Рис. 2.8. *Разбивка домена edu на зоны*

на *cc*, *cs*, *ce*, *me*, и т. д. Каждый из этих поддоменов делегируется ряду серверов имен, некоторые из которых являются компетентными и для *berkeley.edu*. Тем не менее, зоны живут самостоятельно, и могут иметь совершенно отдельный набор авторитативных DNS-серверов.

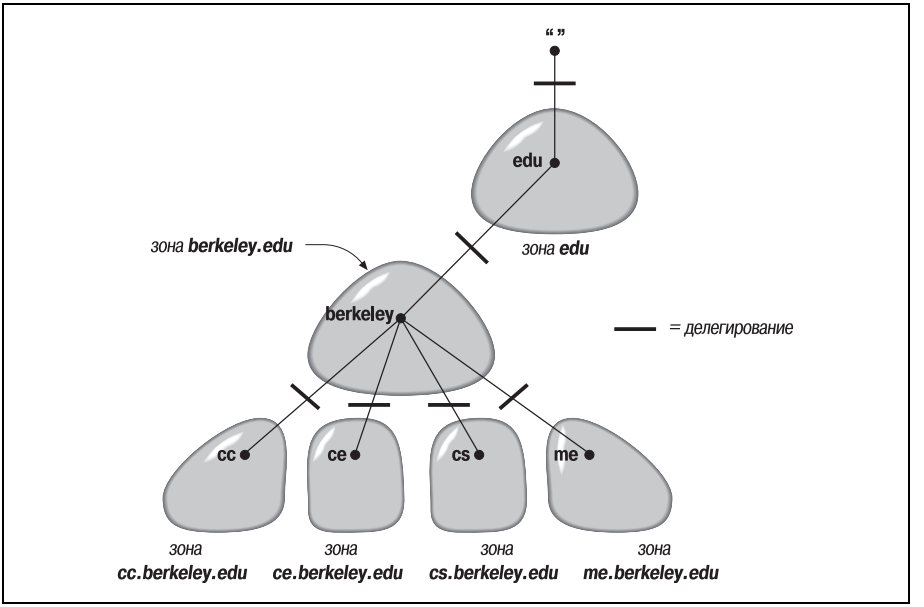

Рис. 2.9. *Разбивка домена berkeley.edu на зоны*

Зона и домен могут включать одни и те же доменные имена, но содержать различные наборы узлов. В частности, зона не содержит никаких узлов, входящих в делегируемые поддомены. Так, домен высшего уровня *ca* (Канада) включает поддомены *ab.ca*, *on.ca* и *qc.ca*, относящиеся к провинциям Альберта, Онтарио и Квебек. Ответственность за сопровождение данных в поддоменах *ab.ca*, *on.ca* и *qc.ca* может быть делегирована серверам имен в каждой из этих провинций. Домен *ca* содержит всю информацию для *ca*, а также всю информацию в *ab.ca*, *on.ca* и *qc.ca*. Но зона *ca* содержит данные только для *ca* (рис. 2.10), и эти данные, вероятно, в основном являются указателями на делегированные поддомены. В то же время *ab.ca*, *on.ca* и *qc.ca* являются отдельными от *ca* зонами.

При этом, если поддомен какого-либо домена не делегирован, зона содержит имена доменов и данные этого поддомена. Так что поддомены *bc.ca* и *sk.ca* (Британская Колумбия и Саскачеван) домена *ca* могут существовать, но могут не быть делегированы. (Возможно, власти провинций Б.К. и Саскачеван еще не готовы управлять собственными зонами, а люди, ответственные за зону высшего уровня *ca*, желают сохранить согласованность пространства имен и немедленно создать поддомены для всех провинций Канады.) В таком случае, зона *ca* бу-

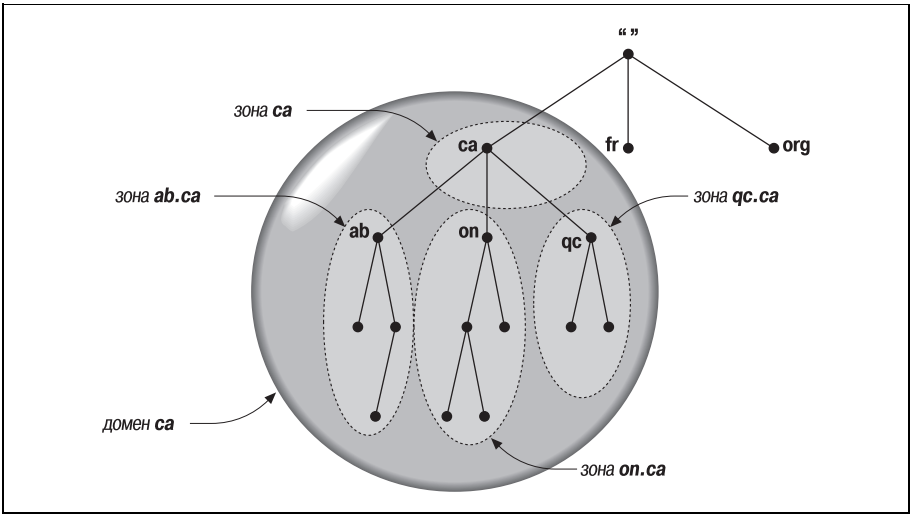

Рис. 2.10. *Домен ca...*

дет иметь неровную нижнюю границу, включая *bc.ca* и *sk.ca*, но не другие поддомены *ca* (рис. 2.11).

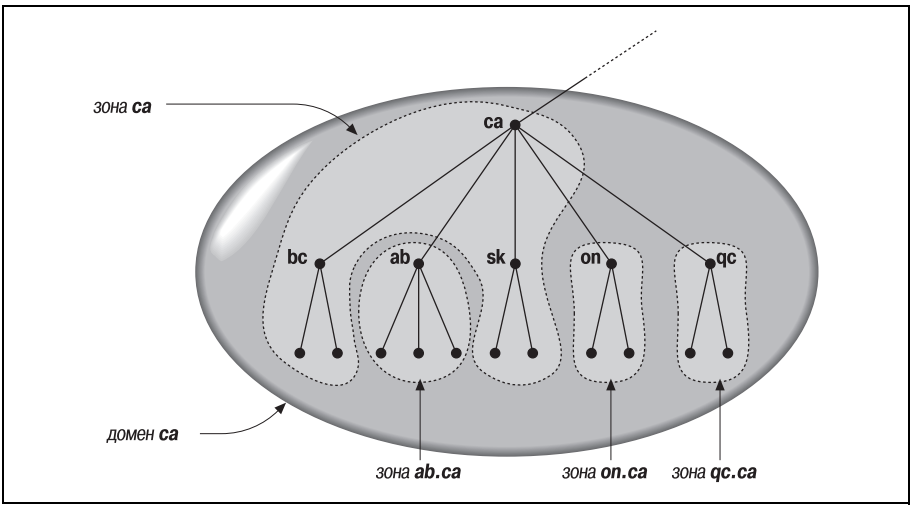

Рис. 2.11. *...и зона ca*

Теперь становится понятно, почему объектом, загружаемым DNS-сервером, является зона, а не домен: домен может содержать больше информации, чем требуется для работы сервера.[1] Домен может содер-

[1] Представьте, что получится, если корневой сервер имен загрузит корневой домен вместо зоны: он загрузит информацию обо всем пространстве имен!

жать данные, делегированные другим серверам. Поскольку зоны ограничиваются делегированием, они никогда не включают делегированные данные.

Обычно, в начале существования у домена еще нет никаких поддоменов. В таком случае, поскольку делегирования не происходит, домен и зона содержат одинаковые данные.

Делегирование поддоменов

Даже в случае отсутствия насущной необходимости делегировать части домена полезно более широко понимать процедуру делегирования поддомена. Делегирование теоретически включает передачу ответственности за какую-то часть домена другой организации. На деле же, происходит назначение различных DNS-серверов в качестве авторитативных в делегируемых поддоменах (мы не случайно употребляем здесь множественное число, именно серверов).

Хранимая зоной информация после делегирования включает уже не информацию по делегированному поддомену, а информацию о серверах имен, являющихся для этого поддомена авторитативными. В таком случае, если у DNS-сервера родительского домена запрашиваются данные для поддомена, в ответ предоставляется список DNS-серверов, которые обладают соответствующей информацией.

Разновидности DNS-серверов

Спецификация DNS определяет два типа DNS-серверов: первичный мастер-сервер (primary master) и дополнительный, или вторичный мастер-сервер (secondary master). *Первичный мастер-сервер* производит загрузку данных для зоны из файла на машине-сервере. *Вторичный мастер-сервер* получает данные зоны от другого DNS-сервера, который является авторитативным для этой зоны и называется его *мастером* (*master server*). Довольно часто мастер-сервер является первичным мастером для зоны, но не обязательно: вторичный мастер-сервер может получать зональные данные и от другого вторичного. Когда запускается вторичный сервер, он устанавливает связь с мастером и, в случае необходимости, получает зональные данные. Этот процесс называется *передачей*, или *трансфером зоны* (*zone transfer*). В наше время предпочтительным термином для вторичного мастер-сервера имен является *slave* (*подчиненный, ведомый*)[1], хотя многие люди (и многие программы, в частности, Microsoft DNS Manager) все еще пользуются прежним термином.

---

[1] В переводе книги в большинстве случаев использован термин «вторичный», то есть практически везде, где в оригинале используется «slave», мы будем использовать термин «вторичный». Это связано со сложившимся профессиональным языком. – *Примеч. перев.*

Как первичный, так и вторичный мастер-серверы зоны являются для этой зоны авторитативными. Несмотря на несколько уничижительное название, вторичные, или slave-серверы не являются серверами второго сорта. Эти два типа серверов предусмотрены в DNS с целью облегчения администрирования. После создания зональных данных и установки первичного мастера можно не беспокоиться о копировании данных с узла на узел при создании дополнительных DNS-серверов зоны. Достаточно просто установить вторичные DNS-серверы, которые будут получать данные от первичного мастер-сервера для этой зоны. После этого передача и получение зональных данных будут происходить автоматически, по необходимости.

Вторичные серверы имеют большое значение, поскольку иметь несколько DNS-серверов для каждой зоны – идея очень правильная. Понадобится больше одного сервера, чтобы обеспечить избыточность, распределить нагрузку, гарантировать, что каждому из узлов зоны легко доступен хотя бы один DNS-сервер. Использование вторичных серверов делает такую архитектуру выгодной и в плане управления.

Однако было бы несколько неточно называть *конкретный* DNS-сервер первичным или вторичным. Ранее мы говорили, что сервер может являться авторитативным для более чем одной зоны. Точно так же, DNS-сервер может являться первичным для одной зоны и вторичным для другой. Но в большинстве случаев сервер имен является либо первичным, либо вторичным для большинства загружаемых им зон. Поэтому, когда мы называем конкретный сервер имен первичным или вторичным, то имеем в виду, что он является таковым для *большинства* зон, которые входят в сферу его компетенции.

Файлы данных зоны

Файлы, из которых первичные DNS-серверы производят чтение зональных данных, называются, и вполне логично, файлами данных зоны. Мы достаточно часто называем их файлами данных или файлами базы данных. Вторичные DNS-серверы также могут загружать зональные данные из файлов. Обычно вторичные серверы настраиваются таким образом, что при каждом получении зональных данных с основного сервера происходит сохранение резервной копии полученной информации в файлах данных. При последующем перезапуске или сбое происходит сначала чтение файлов с резервной копией в целях определения актуальности зональных данных. Это позволяет устранить необходимость в передаче зональных данных для случаев, когда они не изменились, и обеспечивает вторичный DNS-сервер рабочим набором данных в случае недоступности первичного сервера.

Файлы данных содержат RR-записи, описывающие зону. RR-записи описывают все узлы сети в зоне и помечают делегирование поддоменов. В BIND также существуют специальные директивы для включе-

ния содержимого других файлов данных в файл данных зоны аналогично директиве #*include* языка программирования C.

Клиенты DNS

Клиенты DNS (resolvers) позволяют осуществлять доступ к DNS-серверам. Программы, которым требуется информация из пространства доменных имен, используют DNS-клиент. Клиент решает следующие задачи:

- Опрашивание DNS-серверов.
- Интерпретация полученных ответов (RR-записей или сообщений об ошибках).
- Возврат информации в программу, которая ее запросила.

В пакете BIND клиент – это просто набор библиотечных процедур, которые вызываются из программ вроде Telnet и FTP. Клиент – это даже не отдельный процесс. Его хватает ровно на то, чтобы создать запрос, отправить его и ждать ответа, повторно послать запрос, если ответ не получен, но это практически все, на что он способен. Большая часть работы, связанной с поиском ответа на вопрос, ложится на сервер имен. Спецификация DNS называет этот тип анализатора *примитивным* (*stub resolver*).

В других реализациях системы DNS существуют более совершенные клиенты, способные делать гораздо более сложные вещи (например, кэшировать информацию, полученную от DNS-серверов[1]). Но они встречаются гораздо реже, чем примитивный клиент, реализованный в пакете BIND.

Разрешение имен

DNS-серверы умеют исключительно хорошо получать данные из пространства доменных имен. Что неудивительно, учитывая ограниченную интеллектуальность большинства DNS-клиентов. Серверы могут не только поставлять информацию по зонам, для которых являются авторитативными, но и производить поиск в пространстве доменных имен, получая данные зон, в их компетенцию не входящих. Этот процесс называется *разрешением имен* или просто *разрешением*.

Поскольку пространство имен имеет структуру перевернутого дерева, серверу нужна лишь частичка информации, чтобы пробраться к любому узлу дерева: доменные имена и адреса корневых DNS-серверов (разве это больше, чем частичка?). Сервер может обратиться к корневому

[1] К примеру, анализатор CHIVES Роба Остейна, разработанный для системы TOPS-20, имел функцию кэширования.

DNS-серверу с запросом по любому доменному имени пространства доменных имен, после чего получает руководящие указания по продолжению поиска.

Корневые DNS-серверы

Корневые серверы обладают информацией об авторитативных DNS-серверах для каждой из зон высшего уровня. (На деле многие корневые DNS-серверы являются авторитативными для родовых зон высшего уровня, gTLD.) Получив запрос по любому доменному имени, корневой сервер может вернуть, по меньшей мере, список имен и адресов DNS-серверов, авторитативных для зоны высшего уровня, в иерархии которой расположен домен. А DNS-серверы высшего уровня могут вернуть список авторитативных серверов для зоны второго уровня, в иерархии которой расположен домен. Каждый из опрашиваемых серверов либо возвращает информацию о том, как подобраться «поближе» к искомому ответу, либо сразу конечный ответ.

Корневые серверы, очевидно, имеют очень большое значение для процесса разрешения имен. Поэтому в DNS существуют механизмы (скажем, кэширование, которое мы обсудим чуть позже), позволяющие разгрузить корневые серверы. Но в отсутствие другой информации разрешение должно начинаться с корневых DNS-серверов. И это делает корневые серверы ключевым элементом работы DNS. Недоступность всех корневых DNS-серверов сети Интернет в течение длительного времени привела бы к невозможности разрешения имен в сети. Чтобы исключить подобные ситуации, в сети Интернет существует (на момент написания этой книги) тринадцать корневых серверов, расположенных в различных частях сети. Один из них находится в сети PSI-Net, коммерческой информационной магистрали Интернета, один в научной сети NASA, два в Европе, а один в Японии.

Корневые серверы находятся под постоянной нагрузкой, поскольку на них направлено огромное число запросов; даже с учетом того, что серверов тринадцать, трафик для каждого из них очень велик. Недавний неофициальный опрос администраторов корневых DNS-серверов показал, что некоторые из серверов обрабатывают тысячи запросов в секунду.

Несмотря на такую нагрузку, разрешение имен в сети Интернет работает вполне надежно. На рис. 2.12 показан процесс разрешения для адреса реального узла в реальном домене, с учетом того, как производится обход дерева пространства доменных имен.

Локальный DNS-сервер запрашивает адрес *girigiri.gbrmpa.gov.au* у корневого сервера и получает ссылку на DNS-серверы домена *au*. Локальный сервер повторяет запрос, отправляя его одному из DNS-серверов *au*, и получает ссылку на серверы *gov.au*. DNS-сервер *gov.au* отсылает локальный DNS-сервер к серверам *gbrmpa.gov.au*. И наконец, ло-

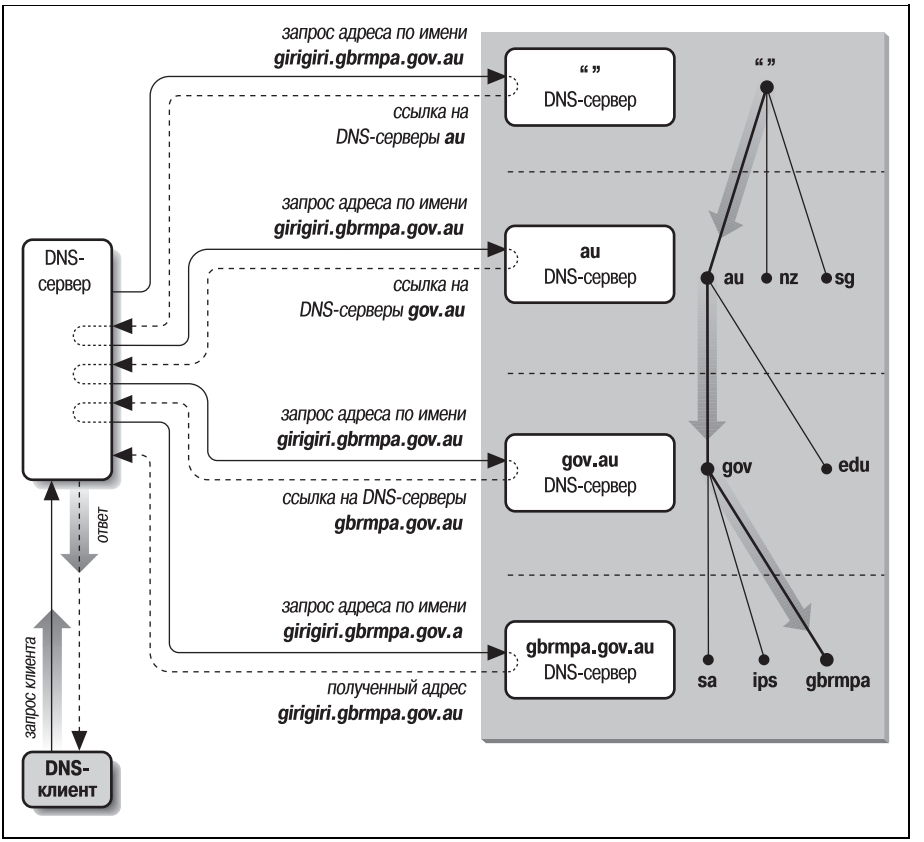

Рис. 2.12. *Разрешение имени girigiri.gbrmpa.gov.au в сети Интернет*

кальный DNS-сервер запрашивает DNS-сервер *gbrmpa.gov.au* и получает искомый адрес.

Рекурсия

Читатели, вероятно, заметили разницу в количестве работы, выполняемой разными DNS-серверами в последнем примере. Четыре сервера, получая запросы, просто возвращали наилучший из уже имевшихся у них ответов – как правило, ссылку на другой DNS-сервер. Эти серверы не выполняли собственные запросы с целью поиска запрошенной информации. Но один DNS-сервер – тот, к которому обратился клиент, – следовал по предлагаемым ссылкам, пока не получил окончательный ответ.

Почему локальный DNS-сервер попросту не перенаправил клиент к другому серверу? Потому что примитивный клиент не способен следовать по таким ссылкам. Каким образом сервер понял, что отвечать

клиенту ссылкой – пустая трата времени? Очень просто: клиент сделал рекурсивный запрос.

Существуют запросы двух видов: *рекурсивные* и *итеративные* (или *нерекурсивные*). Рекурсивные запросы возлагают большую часть работы по разрешению имени на единственный DNS-сервер. *Рекурсия*, или *рекурсивное разрешение имен*, – это название последовательности действий DNS-сервера при получении им рекурсивного запроса. Сервер DNS повторяет какую-то базовую последовательность действий (посылает запрос удаленному серверу и следует по ссылкам), пока не получит ответ, то есть действует аналогично рекурсивному алгоритму программирования. *Итерация*, или *итеративное разрешение имен*, описанное в следующем разделе, – это название последовательности действий DNS-сервера при получении им итеративного запроса.

В случае рекурсии клиент посылает серверу рекурсивный запрос информации для определенного доменного имени. Сервер в таком случае обязан вернуть ответ на запрос (запрошенную информацию) либо ошибку, сообщающую, что данные указанного типа не существуют либо не существует доменное имя.[1] Сервер не может вернуть ссылку на другой DNS-сервер, если запрос рекурсивный.

Если DNS-сервер, получивший запрос, не авторитативен для запрашиваемой информации, ему придется опрашивать другие серверы в поисках ответа. В этом случае сервер может делать либо рекурсивные запросы, возлагая, таким образом, ответственность за нахождение ответа на опрашиваемые DNS-серверы (и снимая с себя ответственность), либо итеративные запросы, возможно, получая ссылки на DNS-серверы, расположенные «ближе» к искомому доменному имени. Существующие реализации очень воспитаны и действуют по второму варианту, следуя по ссылкам, пока не будет найден ответ.[2]

DNS-Сервер, получивший рекурсивный запрос, на который он сам не в состоянии ответить, отправляет запрос «ближайшим известным» серверам. Ближайшие известные серверы – это те, которые являются авторитативными для зоны, ближе всего расположенной к доменному имени, о котором идет речь. К примеру, если сервер получает рекурсивный запрос для адреса доменного имени *girigiri.gbrmpa.gov.au*, он, прежде всего, выяснит, известно ли, какие серверы являются авторитативными для *girigiri.gbrmpa.gov.au*, и, если это известно, отправит

---

[1] DNS-серверы пакета BIND 8 могут быть настроены таким образом, чтобы игнорировать или отказывать в выполнении рекурсивных запросов; причины и способы поступать именно так описаны в главе 11 «Безопасность».

[2] Исключением является DNS-сервер, настроенный таким образом, чтобы передавать все запросы, на которые он не располагает ответом, определенному DNS-серверу. Сервер, настроенный таким образом, называется ретранслятором (forwarder). Подробно об использовании ретрансляторов рассказано в главе 10 «Дополнительные возможности».

запрос одному из них. В противном случае будет произведен аналогичный поиск DNS-серверов для *gbrmpa.gov.au*, а затем для *gov.au* и *au*. По умолчанию, поиск не будет продолжаться дальше корневой зоны, поскольку каждому DNS-серверу известны доменные имена и адреса корневых серверов имен.

Использование ближайших известных DNS-серверов позволяет во всех случаях максимально сократить процесс разрешения. Если DNS-сервер *berkeley.edu* получает рекурсивный запрос адреса для *waxwing.ce.berkeley.edu*, он не будет опрашивать корневые серверы, а использует информацию о делегировании и отправится за данными прямо к DNS-серверам *ce.berkeley.edu*. Точно так же сервер, который только что производил поиск адреса для доменного имени в *ce.berkeley.edu*, не будет начинать поиск с корня, если необходимо повторить процедуру для *ce.berkeley.edu* (или для *berkeley.edu*); причины этого мы опишем чуть позже, в разделе «Кэширование».

Сервер DNS, получающий рекурсивный запрос от DNS-клиента, при поиске повторяет этот запрос в точности так, как он получен от клиента. В случае рекурсивного запроса адреса для *waxwing.ce.berkeley.edu* сервером никогда не будет отправлен явный запрос на информацию о DNS-серверах *ce.berkeley.edu* или *berkeley.edu*, несмотря на то, что эта информация также хранится в пространстве имен. Прямые запросы могут приводить к осложнениям: DNS-серверы *ce.berkeley.edu* могут не существовать (то есть *ce.berkeley.edu* может являться частью зоны *berkeley.edu*). Помимо этого, вполне возможно, что сервер имен *edu* или *berkeley.edu* уже знает адрес *waxwing.ce.berkeley.edu*. Прямой запрос о DNS-серверах *berkeley.edu* или *ce.berkeley.edu* не приведет к получению этой информации.

Итеративное взаимодействие

С другой стороны, итеративное разрешение не требует от DNS-сервера такой серьезной работы. При итеративном разрешении сервер имен возвращает наилучший ответ *из уже известных ему*. Выполнение дополнительных запросов в таком случае не требуется. Сервер имен, получивший запрос, сверяется с локальными данным (в том числе и данными кэша, о котором мы поговорим позже) в поисках запрошенной информации. Если конечный ответ в этих данных не содержится, сервер находит в локальных данных имена и адреса DNS-серверов, наиболее близко расположенных к доменному имени, для которого сделан запрос, и возвращает их в качестве указания для продолжения процесса разрешения. Заметим, что ответ включает *все* серверы имен, содержащиеся в локальных данных, и выбор следующего DNS-сервера для опроса совершается отправителем итеративного запроса.

Выбор авторитативного DNS-сервера

Некоторые из читателей (состоящие в обществе Менса[1]), возможно, задаются вопросом: каким образом сервер, получивший рекурсивный запрос, выбирает DNS-сервер из списка авторитативных? Мы говорили о том, что существует 13 корневых DNS-серверов в сети Интернет. Посылает ли наш DNS-сервер запрос первому из серверов в списке? Или выбирает позицию в списке случайно?

DNS-серверы BIND используют метрику, называемую *временем отклика* (*roundtrip time*, или RTT), для выбора среди авторитативных DNS-серверов одной зоны. Время отклика определяет задержку, с которой приходит ответ на запросы от удаленного сервера. Каждый раз, при передаче запроса удаленному серверу, DNS-сервер BIND запускает внутренний таймер. Таймер останавливается при получении ответа, и метрика фиксируется локальным сервером. Если серверу приходится выбирать один из нескольких авторитативных серверов, выбор падает на сервер с наименьшим показателем RTT.

До того как сервер BIND впервые послал запрос некоему серверу и получил от него ответ, удаленному серверу присваивается случайное значение RTT, которое меньше, чем все прочие, полученные на основании замеров. Таким образом, DNS-сервер BIND гарантированно опросит все авторитативные серверы для определенной зоны случайным образом, прежде чем начнет выбирать предпочтительный на основании метрики.

В общем и целом, это простой, но элегантный алгоритм, позволяющий DNS-серверам BIND достаточно быстро «зацикливаться» на ближайших DNS-серверах, не прибегая к нестандартным, требовательным к ресурсам, механизмам измерения производительности.

Картина в целом

В итоге мы можем наблюдать процесс разрешения, который в целом выглядит примерно так, как показано на рис. 2.13.

DNS-клиент отправляет запрос локальному DNS-серверу, который посылает итеративные запросы ряду других серверов в поисках ответа. Каждый из опрашиваемых серверов возвращает ссылку на другой сервер, который является авторитативным для зоны, расположенной ближе к искомому доменному имени и глубже в дереве пространства имен. В итоге локальный сервер посылает запрос тому авторитативному, который и возвращает ответ. На протяжении всего процесса поис-

[1] Международное сообщество, объединяющее людей, которые попадают в первые 2% населения по своему IQ. Основали его в 1946 году в Англии адвокат Роланд Беррилл (Roland Berrill) и доктор Ланс Вэр (Lance Ware), ученый и юрист. – *Примеч. ред.*

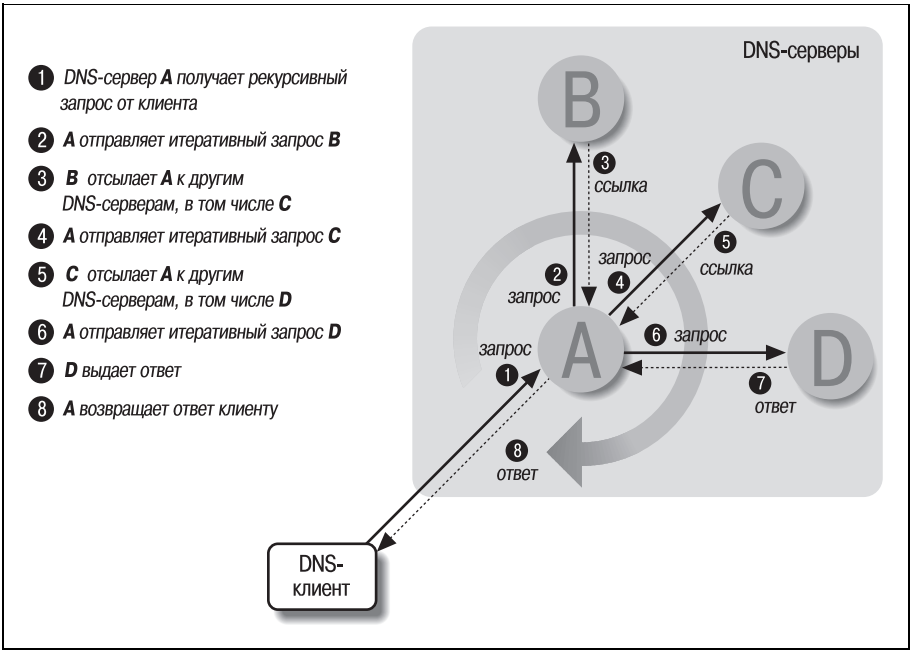

Рис. 2.13. *Процесс разрешения*

ка локальный DNS-сервер использует каждый из получаемых ответов, будь то ссылка или искомая информация, для обновления метрики RTT для реагирующих на запросы DNS-серверов, что впоследствии позволяет принимать осмысленные решения при выборе серверов, используемых в процессе разрешения доменных имен.

Отображение адресов в имена

До сих пор в разговоре о процессе разрешения мы не затрагивали один важный элемент функциональности, а именно – отображение адресов в имена доменов. Отображение адрес-имя необходимо для получения вывода, который легко воспринимается людьми (скажем, при чтении log-файлов). Это отображение также применяется в авторизации. Например, Unix-узлы преобразуют адреса в доменные имена с целью сравнения их с записями в файлах *.rhosts* и *hosts.equiv*. При использовании таблиц узлов отображение адресов в доменные имена довольно тривиально. Требуется обычный последовательный поиск адреса в таблице узлов. Поиск возвращает указанное в таблице официальное имя узла. Но в DNS преобразование адреса в имя происходит несколько сложнее. Данные, в том числе и адреса, которые хранятся в пространстве доменных имен, индексируются по именам. Если есть доменное имя, поиск адреса – процедура относительно простая. Но поиск до-

менного имени, которому соответствует заданный адрес, казалось бы, потребует полного перебора данных для всех доменных имен дерева.

На деле же существует другое решение, более разумное и эффективное. Несложно производить поиск по доменным именам, поскольку они являются своеобразными индексами базы данных, и точно таким же образом можно создать сектор пространства доменных имен, в котором в качестве меток будут использоваться адреса. В пространстве доменных имен сети Интернет таким свойством обладает домен *in-addr.arpa*.

Узлам домена *in-addr.arpa* в качестве меток присваиваются числа в нотации IP-адреса (dotted octet representation – октеты, разделенные точками, – распространенный метод записи 32-битных IP-адресов в виде четырех чисел, принадлежащих интервалу от 0 до 255 и разделяемых точками). Так, домен *in-addr.arpa* может содержать до 256 поддоменов, каждый из которых будет соответствовать одному из возможных значений первого октета IP-адреса. Каждый из этих поддоменов может содержать до 256 собственных поддоменов, каждый из которых будет соответствовать одному из возможных значений второго октета. Наконец, на четвертом уровне существуют RR-записи, ассоциированные с последним октетом, которые содержат полное доменное имя узла по данному IP-адресу. Результатом подобных построений является невероятно объемный домен: *in-addr.arpa*, отображенный на рис. 2.14, достаточно вместительный, чтобы охватить все IP-адреса сети Интернет.

Заметим, что при чтении в доменном имени, IP-адрес оказывается записанным наоборот, поскольку имя читается от листа дерева к корню. К примеру, если IP-адрес узла *winnie.corp.hp.com* – 15.16.192.152, соответствующий узел в домене *in-addr.arpa* – *152.192.16.15.in-addr.arpa*, который отображается в доменное имя *winnie.corp.hp.com*.

IP-адреса могли бы быть представлены в пространстве имен другим способом так, чтобы первый октет IP-адреса был дальше от корня домена *in-addr.arpa*. Тогда в доменном имени IP-адрес читался бы в «правильном» направлении.

Однако IP-адреса, как и доменные имена, образуют иерархию. Сетевые номера выделяются почти так же, как и доменные имена, администраторы вольны разделять «свое» адресное пространство на подсети и делегировать нумерацию в сети другим администраторам. Разница заключается в том, что конкретизация узла для IP-адреса возрастает при чтении слева направо, тогда как для доменных имен – справа налево. Суть этого явления отражена на рис. 2.15.

То, что первые октеты IP-адреса расположены выше в дереве, дает администраторам возможность делегировать ответственность за зоны *in-addr.arpa* в соответствии с топологией сети. Возьмем для примера зону *15.in-addr.arpa*, которая содержит данные обратного преобразования для всех узлов, адреса которых начинаются с цифры 15: эта зона может быть делегирована администраторам сети 15.0.0.0. Это стало бы

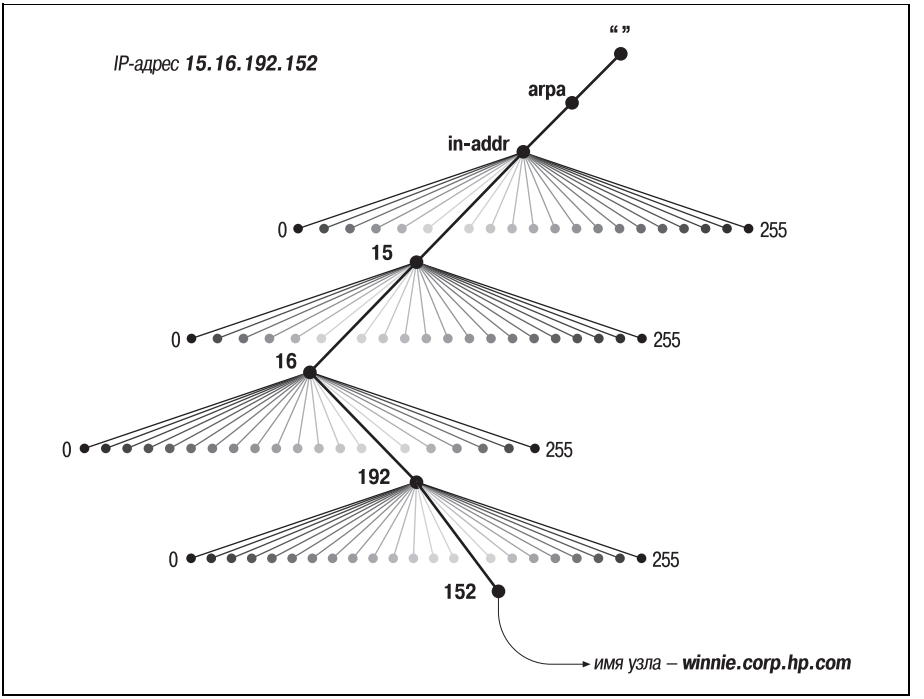

Рис. 2.14. *Домен in-addr.arpa*

невозможным, если бы октеты были расположены в обратном порядке. Если бы IP-адреса записывались наоборот (в другом порядке), зона *15.in-addr.arpa* включала бы каждый узел, IP-адрес которого *заканчивается* цифрой 15, и делегировать эту зону было бы немыслимо.

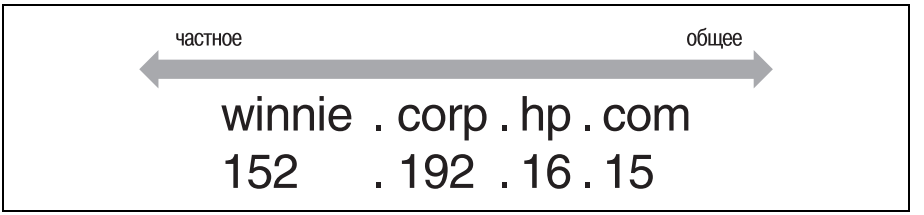

Рис. 2.15. *Иерархия имен и адресов*

Инверсные запросы

Домен *in-addr.arpa* очевидно полезен только для преобразования IP-адресов в доменные имена. Поиск в пространстве доменных имен, которые были бы индексированы по некой *произвольной* информации, чему-нибудь, что не является адресом, потребовал бы другого пространства имен, такого же, как *in-addr.arpa*, либо поиска перебором.

Поиск перебором до некоторой степени является реальностью и носит название *инверсных запросов* (*inverse queries*). Инверсный запрос – это поиск доменного имени, индексирующего указанные данные. Такие запросы обрабатываются исключительно DNS-сервером, который их получает. Этот сервер производит поиск в локальных данных, и, если возможно, возвращает искомое доменное имя. В противном случае поиск завершается. Никакого обмена информацией с другими серверами не происходит.

Поскольку любой отдельно взятый сервер владеет информацией лишь о части доменного пространства, нет гарантий, что инверсный запрос приведет к получению ответа. Так, если DNS-сервер получает инверсный запрос по IP-адресу, о котором ему ничего не известно, то не может вернуть ответ на запрос, но не может также и утверждать, что такого адреса не существует, поскольку владеет лишь частью базы данных DNS. Более того, согласно спецификации системы доменных имен, реализация инверсных запросов является необязательной. В BIND версии 4.9.8 содержится код, реализующий инверсные запросы, но по умолчанию он закомментирован. BIND версий 8 и 9 не включает такого кода вовсе, хотя серверы имен в этих версиях BIND распознают инверсные запросы и могут возвращать поддельные ответы.[1] Нас такое положение вещей вполне устраивает, поскольку очень немногие программы (скажем, древние версии *nslookup*) до сих пор пользуются инверсными запросами.

Кэширование

По сравнению с простым поиском в таблице узлов процесс разрешения может показаться ужасно запутанным и нескладным. В действительности же этот процесс довольно быстр. Одна из возможностей, существенно его ускоряющих, – *кэширование*.

Чтобы обработать рекурсивный запрос, DNS-серверу приходится самостоятельно сделать довольно много запросов. Однако в процессе сервер получает большое количество информации о пространстве доменных имен. Каждый раз, получая список DNS-серверов в ссылке, он знакомится с серверами, авторитативными для какой-то зоны, и узнает адреса этих серверов. Когда завершается процесс разрешения, и необходимые данные возвращаются клиенту, сделавшему исходный запрос, новые знания могут быть сохранены для последующего использования. В версии BIND 4.9, а также во всех версиях BIND 8 и 9, в серверах имен реализовано даже *отрицательное кэширование*: если авторитативный сервер возвращает информацию о том, что запрошенное имя домена или указанный тип данных не существует, локальный

[1] Более подробно этот момент описан в разделе «Запрос отвергнут» в главе 12 «nslookup и dig».

сервер временно кэширует и эту информацию. DNS-серверы кэшируют получаемые данные, чтобы ускорить обработку последующих запросов. Когда в следующий раз DNS-клиент сделает запрос по доменному имени, о котором серверу уже что-то известно, процесс разрешения пройдет быстрее. Если сервер кэшировал ответ, положительный или отрицательный, ответ просто возвращается клиенту. Но даже если готового ответа у DNS-сервера нет, он, возможно, «помнит в лицо» те DNS-серверы, которые являются авторитативными для зоны этого конкретного доменного имени, и будет напрямую работать с ними.

Предположим, наш DNS-сервер уже выяснял адрес для *eecs.berkeley.edu*. В процессе были кэшированы имена и адреса DNS-серверов *eecs.berkeley.edu* и *berkeley.edu* (а также IP-адрес *eecs.berkeley.edu*). Если теперь клиент пошлет DNS-серверу запрос, касающийся адреса для имени *baobab.cs.berkeley.edu*, при обработке этого запроса наш сервер сможет пропустить отправку запросов корневым DNS-серверам. Опознав в *berkeley.edu* ближайшего предка *baobab.cs.berkeley.edu*, о котором уже что-то известно, наш сервер начнет с запроса к DNS-серверу *berkeley.edu* (рис. 2.16). С другой стороны, если бы наш DNS-сервер выяснил, что адреса для *eecs.berkeley.edu* не существует, в следующий раз на подобный запрос он вернул бы соответствующий ответ из кэша.

Помимо ускорения разрешения кэширование устраняет необходимость вовлекать в процесс разрешения корневые DNS-серверы для случаев,

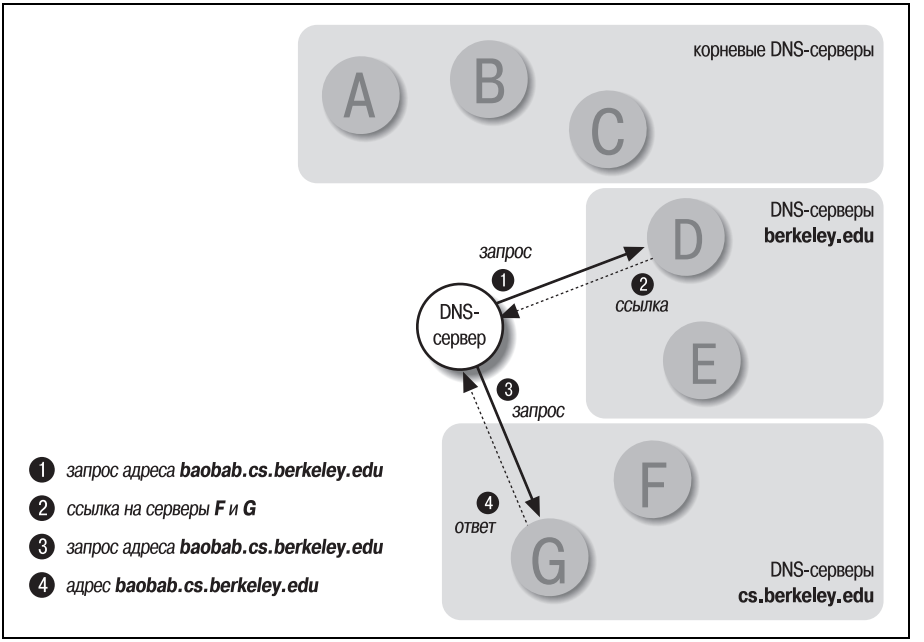

Рис. 2.16. *Процесс разрешения для baobab.cs.berkeley.edu*

когда ответ может быть получен локально. Это означает уменьшение зависимости от корневых серверов и снижение нагрузки на них.

Время жизни

Разумеется, DNS-серверы не могут кэшировать данные навсегда. Иначе изменения на авторитативных серверах никогда не распространялись бы по сети. Удаленные серверы просто продолжали бы использовать кэшированную информацию. Поэтому администратор зоны, которая содержит данные, обычно определяет для этих данных *время жизни* (*time to live*, или TTL). Время жизни – это интервал времени, в течение которого произвольному DNS-серверу разрешается пользоваться кэшированными данными. По окончании этого временного интервала сервер обязан удалить кэшированную информацию и получить новую от авторитативных DNS-серверов. Это касается также кэшируемых отрицательных ответов, сервер имен обязан удалять их на случай, если на авторитативных DNS-серверах появилась новая информация.

Когда администратор выбирает время жизни для своих данных, то фактически пытается найти золотую середину между производительностью и согласованностью данных. Небольшой показатель TTL гарантирует, что данные о зоне будут согласованы по всей сети, поскольку удаленные серверы будут обязаны быстрее выбросить данные из кэша и обратиться на авторитативные серверы за новыми данными. С другой стороны, увеличивается нагрузка на DNS-серверы зоны и увеличивается время разрешения, когда речь идет об информации из этой зоны.

Напротив, большой показатель TTL сокращает среднее время, затрачиваемое на получение информации из зоны, поскольку данные о зоне кэшируются в течение длительного времени. Минус же в том, что информация на удаленных DNS-серверах в течение долгого времени может не соответствовать действительности в случае изменения данных локальных DNS-серверов.

Но хватит уже теории – читателям, вероятно, не терпится с ней закончить. Однако прежде чем можно будет перейти к реальным зонам и серверам имен, предстоит сделать домашнее задание, которое мы дадим в следующей главе.

- *Приобретение пакета BIND*
- *Выбор доменного имени*

3

С чего начать?

– Как тебя зовут? – спросила Лань.
У нее был мягкий и нежный голос.
– Если б я только знала! – подумала бедная Алиса.
Вслух она грустно промолвила: – Пока никак...
– Постарайся вспомнить, – сказала Лань. – Так нельзя... Алиса постаралась, но все было бесполезно.
– Скажите, а как вас зовут? – робко спросила она. – Вдруг это мне поможет...
– Отойдем немного, – сказала Лань. – Здесь мне не вспомнить...

Теперь, когда вы познакомились с теоретической базой DNS, можно переходить к практике. Прежде чем начать работу с зонами, необходимо получить копию пакета BIND. Как правило, этот пакет входит в стандартную поставку операционных систем семейства Unix. Однако довольно часто есть необходимость в обновлении пакета с целью получения требуемой функциональности и уровня безопасности.

Получив BIND, следует выбрать доменное имя для основной зоны, и эта задача не столь проста, как кажется на первый взгляд, поскольку связана с поиском подходящего места в пространстве имен сети Интернет. Выбрав имя, следует связаться с администраторами родительской зоны для выбранного доменного имени.

Но все по порядку. Поговорим о том, где можно получить пакет BIND.

Приобретение пакета BIND

В случае необходимости создавать зоны и управлять DNS-серверами этих зон, следует обзавестись пакетом BIND. Даже если размещением зон будет заниматься кто-то другой, пакет может оказаться полезным в любой момент. К примеру, можно использовать локальный DNS-сервер для проверки файлов данных до передачи их администратору удаленных DNS-серверов.

Большинство коммерческих Unix-систем содержат пакет BIND в составе сетевых TCP/IP-приложений, набор которых обычно входит в стандартную поставку, так что пользователи получают BIND бесплатно. Даже в случаях, когда за сетевые приложения взимается отдельная плата, те из пользователей, кому из-за активной сетевой работы нужен BIND, скорее всего его уже приобрели.

При этом, если не существует готовой версии BIND для конкретной разновидности Unix-системы, но нужна лучшая из последних версий, всегда можно получить исходные тексты пакета. К счастью, они распространяются совершенно свободно. Исходные тексты самых последних версий BIND (на момент написания книги это BIND 8.2.3 и 9.1.0) доступны для загрузки с публичного FTP-сервера консорциума Internet Software Consortium по адресу *ftp.isc.org*, файлы называются */isc/bind/src/cur/bind-8/bind-src.tar.gz* и */isc/bind9/9.1.0/bind-9.1.0.tar.gz* соответственно. Сборка этих двух версий на большинстве распространенных Unix-систем – дело достаточно простое.[1] Консорциум ISC включает список Unix-подобных операционных систем, на которых собирается и работает BIND, и приводит его в файле *src/INSTALL*: в список входят версии Linux, Digital Unix и Solaris 2. Присутствует также список других Unix- (и не совсем Unix) операционных систем (кто-нибудь работает на MPE?), которые BIND поддерживал в прошлом и для которых сборка последних версий пакета может производиться без существенных усилий.[2] Независимо от того, какая именно операционная система используется, мы настоятельно рекомендуем прочитать все связанные с этой системой разделы файла *src/INSTALL*. Мы также включили инструкции по сборке BIND версий 8.2.3 и 9.1.0 для RedHat Linux 6.2 в виде приложения C «Сборка и установка BIND на Linux-системах». И это поразительно короткое приложение.

У кого-то из читателей, вполне возможно, уже есть версия пакета BIND, которая входила в поставку операционной системы, и они задаются вопросом, нужна ли самая последняя и самая лучшая версия BIND? Что в ней есть такого, чего нет в более ранних версиях? Перечислим кратко:

Лучшая защищенность

Чуть ли не самой важной причиной использовать последнюю версию BIND является тот факт, что эта версия наименее уязвима для внешних атак. BIND версий 8.2.3 или 9.1.0 успешно противостоит

---

[1] Сборка более ранних версий BIND 9 (предшествующих версии 9.1.0) может оказаться не столь тривиальной, поскольку этим версиям требуется механизм *pthreads*, реализация которого некорректна во многих операционных системах. BIND 9.1.0 и более поздних версий может собираться без *pthreads* посредством выполнения команды *configure --disable-threads*.

[2] Достоверно известно, что BIND версии 8.2.3 без проблем собирается на нескольких из этих операционных систем.

всем широко известным атакам, а BIND версии 4.9.8 – значительному их числу. У более ранних версий BIND много уязвимых мест, известных злоумышленникам. Если DNS-сервер работает в сети Интернет, мы рекомендуем использовать BIND версии 8.2.3 или 9.1.0, в самом крайнем случае – BIND 4.9.8 либо самую последнюю на момент прочтения этой книги версию.

Механизмы безопасности

BIND 8 и 9 поддерживают списки управления доступом для запросов, передачи зоны, а также динамических обновлений. Серверы BIND 4.9 поддерживают списки управления доступом для запросов и передачи зоны, а более ранние версии не реализуют списки доступа. Определенным DNS-серверам, в особенности тем, которые работают на узлах-бастионах или узлах, на которых предъявляются повышенные требования к безопасности, может потребоваться использование возможностей списков управления доступом.

Подробно об этом механизме мы расскажем в главе 11 «Безопасность».

DNS UPDATE

BIND 8 и 9 поддерживают стандарт динамических обновлений (Dynamic Update), описанный в документе RFC 2136. Это позволяет уполномоченным агентам обновлять данные зоны путем посылки специальных сообщений обновления, содержащих команды добавления или удаления записей ресурсов. В серверах BIND 4 механизм динамического обновления не реализован.

Подробно о динамических обновлениях мы расскажем в главе 10 «Дополнительные возможности».

DNS NOTIFY

BIND 8 и 9 поддерживают уведомления об изменениях зоны, механизм, позволяющий первичному мастер-серверу зоны уведомлять вторичные серверы в случае увеличения порядкового номера (serial) зоны. В серверах BIND 4 механизм NOTIFY не реализован.

Подробно мы расскажем о возможностях NOTIFY в главе 10.

Инкрементальная передача зоны

BIND версии 8.2.3 и BIND 9 реализуют механизм инкрементальной передачи зоны, позволяющий вторичным DNS-серверам запрашивать у первичных серверов только изменения зональных данных. Этот механизм делает передачу зон более быстрой и намного более эффективной, он в особенности важен для крупных, часто меняющихся зон.

Синтаксис настройки

Синтаксис настройки, используемый в BIND 8 и 9, существенно отличается от используемого в BIND 4. Будучи более гибким и более мощным, он также требует изучения совершенно иной системы на-

стройки пакета BIND. Впрочем, для решения этой задачи вполне подойдет настоящая книга.

Синтаксис настройки BIND 8 и 9 будет представлен в главе 4 «Установка BIND» и описан в оставшейся части книги.

Сводку возможностей, реализованных в четырех наиболее распространенных версиях BIND (4.9.8, 8.1.2, 8.2.3, 9.1.0), мы включили в книгу в виде приложения B «Таблица совместимости BIND». В случае наличия сомнений, связанных с выбором версии пакета, либо в случае необходимости использовать экзотическую возможность, про которую неизвестно, поддерживается ли она в BIND 9, следует обратиться к этому приложению.

Если после изучения представленного списка и приложения вы приходите к выводу, что нужен BIND 8 или 9, но ни та ни другая версия не входят в состав используемой операционной системы, придется загрузить исходные тексты пакета и собрать его в нужной конфигурации.

Полезные списки рассылки и конференции Usenet

Инструкции, касающиеся процесса переноса пакета BIND на произвольную Unix-систему, могли бы вылиться в еще одну книгу подобного размера, поэтому за информацией подобного рода мы отсылаем читателей к списку рассылки пользователей BIND (*bind-users@isc.org*) и соответствующей конференции Usenet (*comp.protocols.dns.bind*).[1] Для BIND 9 существует самостоятельный список рассылки, *bind9-users@isc.org*.[2] Люди, которые читают списки рассылки пользователей BIND и делятся в этих списках своими ценными мыслями, могут быть невероятно полезны в плане вопросов, связанных с переносом BIND. Но прежде чем спрашивать в списке рассылки, существует ли порт BIND для конкретной платформы, не забудьте свериться с результатами поиска по архиву списка рассылки, который доступен по адресу *http://www.isc.org/ml-archives/bind-users*. Помимо этого, взгляните на веб-страницу ISC, посвященную пакету BIND (*http://www.isc.org/products/BIND*), где могут быть доступны примечания и ссылки, непосредственно касающиеся используемой вами операционной системы, а

[1] Чтобы задать вопрос в списке рассылки сети Интернет, следует всего лишь отправить сообщение на адрес списка рассылки. Однако прежде следует подписаться на список, послав просьбу о добавлении администратору списка. Не следует посылать запросы такого рода непосредственно в список рассылки, это считается невежливым. По принятой в сети Интернет практике с администратором можно связаться по адресу *list-request@domain*, где *list@domain* – это адрес списка рассылки. Так, связаться с администратором списка рассылки пользователей BIND можно по адресу *bind-users-request@isc.org*.

[2] Большинство разработчиков BIND 9 читает исключительно список рассылки *bind9-users*.

также на каталог ресурсов DNS Андраса Саламона на предмет поиска собранных пакетов BIND. Каталог содержит краткий перечень собранных пакетов (*http://www.dns.net/dnsrd/bind.html*).

Еще один список рассылки, который может оказаться интересным, – это *namedroppers*. Люди, участвующие в списке рассылки *namedroppers*, являются членами рабочего комитета организации IETF, который занимается разработкой расширенной спецификации DNS, или DNSEXT. Так, к примеру, обсуждение нового типа записи DNS, скорее всего, будет происходить в списке *namedroppers*, а не в общем списке рассылки BIND. Более подробная информация о группе DNSEXT доступна по адресу *http://www.ietf.org/html.charters/dnsext-charter.html*.

Адрес списка рассылки *namedroppers* – *namedroppers@ops.ietf.org*, соответствующая конференция Интернета называется *comp.protocols.dns.std*. Чтобы подписаться на рассылку *namedroppers*, следует отправить почтовое сообщение на адрес *namedroppers-request@ops.ietf.org*, текст сообщения должен представлять собой строку «subscribe namedroppers».

Как узнать IP-адрес

Наверное, многие заметили, что мы привели несколько доменных имен FTP-узлов, на которых доступны для скачивания различные программы, и что упоминаемые адреса списков рассылки также содержат доменные имена. Этот факт должен подчеркнуть важность DNS: очень много полезных программ и советов доступны с помощью именно DNS. К сожалению, здесь присутствует проблематика курицы и яйца: невозможно послать электронное сообщение на адрес, который содержит доменное имя, если система DNS еще не установлена, т. е. оказывается невозможным задать в списке рассылки вопрос, касающийся установки DNS.

Конечно, мы могли бы приводить IP-адреса упоминаемых узлов, но поскольку IP-адреса меняются часто (во всяком случае, во временных понятиях книгоиздания), вместо этого мы объясним, как можно *временно* использовать чужой DNS-сервер для получения информации. Если узел имеет подключение к сети Интернет и программу *nslookup*, существует возможность получать информацию из пространства имен Интернета. Чтобы получить, к примеру, IP-адрес для *ftp.isc.org*, можно воспользоваться такой командой:

```
% nslookup ftp.isc.org. 207.69.188.185
```

В данном случае программе *nslookup* предписывается послать запрос DNS-серверу, который работает на узле, имеющем IP-адрес 207.69.188.185, с целью выяснения IP-адреса для *ftp.isc.org*. Должен получиться примерно такой результат:

```
Server:  ns1.mindspring.com
```

```
Address:  207.69.188.185

Name:    isrv4.pa.vix.com
Address: 204.152.184.27
Aliases:  ftp.isc.org
```

Теперь можно использовать IP-адрес узла *ftp.isc.org* (204.152.184.27) для проведения FTP-сеанса.

Откуда мы узнали, что на узле с IP-адресом 207.69.188.185 работает DNS-сервер? От нашего провайдера интернет-услуг компании Mindspring, которая сообщила нам адрес одного из своих серверов. Если провайдер интернет-услуг предоставляет своим клиентам DNS-серверы (как это происходит в большинстве случаев), эти серверы можно использовать по назначению. Если какой-либо провайдер не предоставляет DNS-серверы (как не стыдно!), можно *временно* использовать один из DNS-серверов, упоминаемых в этой книге. Если использовать эти серверы только для поиска нескольких адресов и небольшого объема другой информации, администраторы, скорее всего, не будут возражать. Однако считается недопустимым указывать DNS-клиенту или другому инструменту, выполняющему запросы к DNS, адрес чужого DNS-сервера в качестве постоянного.

Разумеется, если у вас уже есть доступ к узлу с подключением к сети Интернет *и* работающей DNS, можно воспользоваться этим каналом для FTP-копирования нужных программ.

Получив рабочую версию пакета BIND, можно начинать думать о выборе доменного имени.

Выбор доменного имени

Выбор доменного имени – задача более сложная, чем может показаться, поскольку она связана не только с выбором имени, *но и* с выяснением, кто заведует родительской зоной. Другими словами, необходимо выяснить, где в пространстве доменных имен сети Интернет место нового узла, а затем – кто управляет этим сектором пространства.

Первый шаг в процессе выбора доменного имени – выяснение того, к какому сектору пространства доменных имен принадлежит ваш узел. Легче всего начать с вершины и двигаться вниз: выбрать сначала домен высшего уровня, затем поддомен в этом домене, которому вы соответствуете.

Помните, чтобы узнать, как выглядит пространство доменных имен сети Интернет (не считая того, о чем мы уже рассказали), понадобится доступ к сети Интернет. Необязательно иметь доступ к узлу с работающими службами DNS, но это не будет лишним. Если доступа к такому узлу нет, придется «позаимствовать» услуги службы DNS у других DNS-серверов (как в примере с именем *ftp.isc.org*), чтобы начать работу.

О регистраторах и регистрах

Прежде чем двинуться дальше, необходимо определить несколько терминов: *регистр*, *регистратор* и *регистрация*. Эти термины не определены в спецификации DNS, но применимы к существующей структуре управления пространством доменных имен сети Интернет.

Регистр – это организация, отвечающая за сопровождение файлов данных доменов высшего уровня (а в действительности – зон), то есть информации о делегировании всех поддоменов. При существующей сегодня структуре сети Интернет, каждый домен высшего уровня привязан не более чем к одному регистру. *Регистратор* является интерфейсом между клиентами и регистром, он обеспечивает процедуры регистрации и предоставляет дополнительные платные услуги. Регистратор передает в регистр зональные и прочие данные (включая контактную информацию) по каждому из своих клиентов, входящих в домен высшего уровня.

Таким образом, *регистрация* – это действие, посредством которого клиент сообщает регистратору, каким DNS-серверам следует делегировать поддомен, и также снабжает регистратора контактной и платежной информацией. Регистратор передает информацию в регистр.

Компания Network Solutions Inc. является эксклюзивным регистром и регистратором для доменов высшего уровня *com*, *net*, *org* и *edu*. А теперь вернемся к нашим задачам.

Где мое место?

Если ваша организация подключена к сети Интернет вне США, прежде всего необходимо решить, в каком из доменов высшего уровня запрашивать поддомен – в одном из родовых, таких как *com*, *net*, либо *org*, либо в домене высшего уровня, соответствующем конкретной стране. Родовые домены высшего уровня вовсе не отведены исключительно под организации США. Для компании, которая является мульти- или транснациональной, и для которой не подходит домен высшего уровня какой-то конкретной страны, либо в случае, когда родовой домен просто более предпочтителен, имеет смысл регистрация в одном из родовых доменов. Выбрав такой путь регистрации, читатели могут перейти к разделу «Родовые домены высшего уровня».

В случае выбора поддомена в домене высшего уровня конкретной страны следует проверить, зарегистрирован ли для этой страны домен высшего уровня, и если да, то каким структурным делением он обладает. Если необходимо уточнить имя домена высшего уровня для страны, обращайтесь к приложению D «Домены высшего уровня».

В доменах высшего уровня некоторых стран, таких как Новая Зеландия (*nz*), Австралия (*au*) и Великобритания (*uk*), существует организационное деление для доменов второго уровня. То есть имена доменов

второго уровня, скажем, *co* или *com* для коммерческих сущностей, отражают принадлежность организации. Домены других стран, скажем, Франции (*fr*) или Дании (*dk*) делятся на множество поддоменов, которые управляются отдельными университетами и компаниями; домен Университета Сент-Этьена носит название *univ-st-etienne.fr*, а домен датской группы пользователей Unix-систем – *dkuug.dk*. Для многих доменов высшего уровня существуют веб-страницы, описывающие их структуру. Если вы не знаете URL веб-страницы домена высшего уровня конкретной страны, сверьтесь с каталогом ссылок на подобные страницы, расположенным по адресу *http://www.allwhois.com*.

В случае отсутствия подобной веб-страницы для домена высшего уровня, возможно, придется воспользоваться инструментом вроде *nslookup*, чтобы покопаться в пространстве доменных имен и оценить структуру поддоменов. (Те из читателей, кто чувствует дискомфорт при использовании еще не изученного средства, могут на время отвлечься на прочтение главы 12 «nslookup и dig».) Вот так, к примеру, с помощью *nslookup* можно получить список поддоменов домена *au*:

```
% nslookup - 207.69.188.185 – использовать DNS-сервер по адресу
                             207.69.188.185
Default Server:  ns1.mindspring.com
Address:  207.69.188.185

>set type=ns                – Поиск DNS-серверов (ns)
> au.                       – для зоны au
Server:  ns1.mindspring.com
Address: 207.69.188.185

au      nameserver = MUNNARI.OZ.AU
au      nameserver = MULGA.CS.MU.OZ.AU
au      nameserver = NS.UU.NET
au      nameserver = NS.EU.NET
au      nameserver = NS1.BERKELEY.EDU
au      nameserver = NS2.BERKELEY.EDU
au      nameserver = VANGOGH.CS.BERKELEY.EDU
MUNNARI.OZ.AU     internet address = 128.250.1.21
MULGA.CS.MU.OZ.AU       internet address = 128.250.1.22
MULGA.CS.MU.OZ.AU       internet address = 128.250.37.150
NS.UU.NET         internet address = 137.39.1.3
NS.EU.NET         internet address = 192.16.202.11
NS1.BERKELEY.EDU  internet address = 128.32.136.9
NS1.BERKELEY.EDU  internet address = 128.32.206.9
NS2.BERKELEY.EDU        internet address = 128.32.136.12
NS2.BERKELEY.EDU        internet address = 128.32.206.12
> server ns.uu.net.   – Теперь сделать запрос к одному из этих DNS-
                        серверов, предпочтительно к самому близкому!
Default Server:  ns.uu.net
Addresses:  137.39.1.3

> ls -t au. – листинг для зоны au
```

– NS-записи зоны отмечают делегирование поддоменов, поэтому
– имена поддоменов могут быть получены.
– Следует иметь в виду, что из соображений безопасности не все серверы позволяют выполнять перечисление для зон.

```
 [ns.uu.net]
$ORIGIN au.
@                 3D IN NS        mulga.cs.mu.OZ
                  3D IN NS        vangogh.CS.Berkeley.EDU.
                  3D IN NS        ns1.Berkeley.EDU.
                  3D IN NS        ns2.Berkeley.EDU.
                  3D IN NS        ns.UU.NET.
                  3D IN NS        ns.eu.NET.
                  3D IN NS        munnari.OZ
ORG               1D IN NS        mulga.cs.mu.OZ
                  1D IN NS        rip.psg.COM.
                  1D IN NS        munnari.OZ
                  1D IN NS        yalumba.connect.COM
info              1D IN NS        ns.telstra.net.
                  1D IN NS        ns1.telstra.net.
                  1D IN NS        munnari.oz
                  1D IN NS        svc01.apnic.net.
otc               4H IN NS        ns2.telstra.com
                  4H IN NS        munnari.oz
                  4H IN NS        ns.telstra.com
OZ                1D IN NS        mx.nsi.NASA.GOV.
                  1D IN NS        munnari.OZ
                  1D IN NS        mulga.cs.mu.OZ
                  1D IN NS        dmssyd.syd.dms.CSIRO
                  1D IN NS        ns.UU.NET.
csiro             1D IN NS        steps.its.csiro
                  1D IN NS        munnari.OZ
                  1D IN NS        manta.vic.cmis.csiro
                  1D IN NS        dmssyd.nsw.cmis.csiro
                  1D IN NS        zoiks.per.its.csiro
COM               1D IN NS        mx.nsi.NASA.GOV.
                  1D IN NS        yalumba.connect.COM
                  1D IN NS        munnari.OZ
                  1D IN NS        mulga.cs.mu.OZ
                  1D IN NS        ns.ripe.NET.
> ^D
```

Техника процесса достаточно проста: получить список DNS-серверов для домена высшего уровня (поскольку лишь они обладают полной информацией о соответствующей зоне), затем сделать запрос к одному из этих DNS-серверов на предмет получения списка DNS-серверов для делегированных поддоменов.

Если по именам поддоменов невозможно определить, какому сектору должен принадлежать ваш будущий домен, всегда можно найти контактную информацию для соответствующей зоны и отправить свой вежливый вопрос электронной почтой в службу технической поддерж-

ки. Если вам кажется, что новый домен должен входить в один из существующих, но полной уверенности нет, всегда можно уточнить этот момент у людей, которые администрируют выбранный поддомен.

Чтобы узнать, к кому обращаться с вопросами по конкретному поддомену, придется найти соответствующую зоне RR-запись типа SOA (start of authority, начало авторитативности). В SOA-записи каждой зоны существует поле, содержащее адрес электронной почты лица, отвечающего за техническую поддержку зоны.[1] (Остальные поля SOA-записи содержат общую информацию о зоне, чуть позже мы обсудим их более подробно.) SOA-запись зоны также можно просмотреть с помощью программы *nslookup*.

К примеру, если бы мы хотели узнать о назначении поддомена *csiro*, то могли бы узнать, кто за него отвечает из содержимого SOA-записи для *csiro.au*:

```
% nslookup - 207.69.188.185
Default Server:  ns1.mindspring.com
Address:  207.69.188.185

>set type=soa      – Поиск информации из RR-записи типа SOA
>csiro.au.         – для csiro.au
Server:  ns1.mindspring.com
Address: 207.69.188.185

csiro.au
         origin = steps.its.csiro.au
         mail addr = hostmaster.csiro.au
         serial = 2000041301
         refresh = 10800 (3H)
         retry   = 3600 (1H)
         expire  = 3600000 (5w6d16h)
         minimum ttl = 86400 (1D)
```

Поле *mail addr* содержит интернет-адрес контактного лица *csiro.au*. Чтобы преобразовать этот адрес в формат электронного адреса сети Интернет, следует заменить первый символ «.» в адресе на символ «@». Так, *hostmaster.csiro.au* превращается в *hostmaster@csiro.au*.[2]

---

[1] Поддомен и зона идентифицируются одним и тем же доменным именем, но SOA-запись в действительности принадлежит зоне, а не поддомену. Человек, адрес которого указан в SOA-записи, возможно, не отвечает за весь поддомен (поскольку он может содержать делегированные поддомены), но совершенно определенно знает, каково предназначение этого поддомена.

[2] Такая форма почтового адреса Интернет является наследием двух некогда существовавших типов записей DNS, MB и MG. Записи MB (mailbox, почтовый ящик) и MG (mail group, почтовая группа) определяли почтовые ящики и почтовые группы (списки) сети Интернет как поддомены соответствующих доменов. Записи и MB и MG не получили широкого распространения, но формат адреса, который ими предлагался, используется в SOA-записях, возможно, из соображений сентиментальности.

Использование whois

Служба *whois* также может содействовать выяснению назначения определенного домена. К сожалению, существует много *whois*-серверов – большинство хороших администраторов доменов высшего уровня заводят такую службу – но, в отличие от DNS-серверов, whois-серверы не общаются один с другим. Следовательно, прежде чем воспользоваться *whois*-службой, необходимо найти нужный *whois*-сервер.

Проще всего начать поиск нужного сервера *whois* по адресу *http://www. allwhois.com* (рис. 3.1). Мы уже говорили, что на этом сайте присутствует список веб-страниц доменов высшего уровня разных стран; на нем же есть перечень доменов высшего уровня и привязанных к ним whois-страниц, предоставляющих HTML-интерфейс для работы с *whois*-сервером.

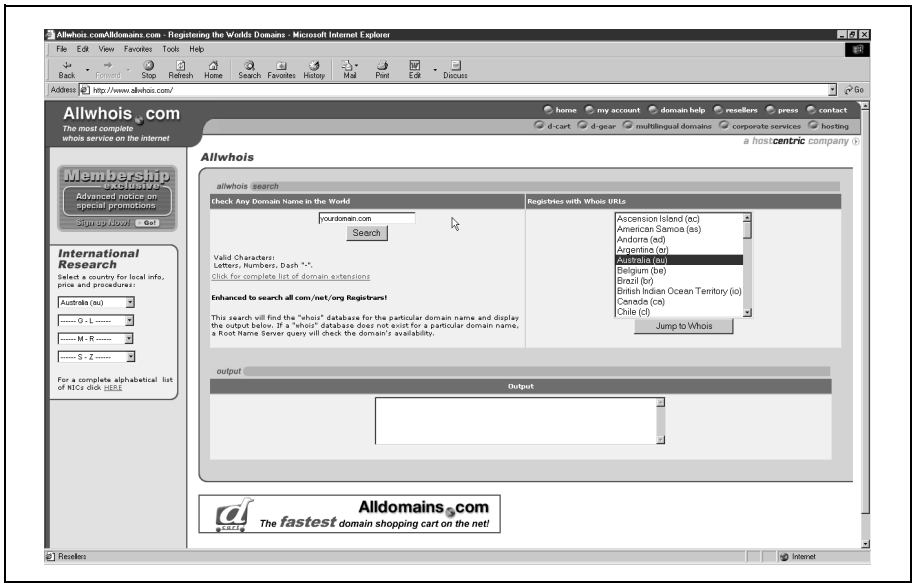

Рис. 3.1. *Сайт Allwhois.com*

Выбрав из списка позицию «Australia (au)», мы можем перейти по ссылке «Jump to Whois» и попасть прямо на страницу, которая позволит получить информацию по *csiro.au* (рис. 3.2).

После ввода доменного имени и нажатия кнопки «Submit» сервер whois возвращает найденную информацию (рис. 3.3).

Для особенно ленивых будет более интересна инициатива WebMagic, цель которой – обеспечить универсальный веб-интерфейс к службам *whois*. Веб-сайт *http://www.webmagic.com/whois/index.html* предоставляет посетителям возможность выбрать домен высшего уровня (и иногда – домен второго уровня), который содержит поддомен, инфор-

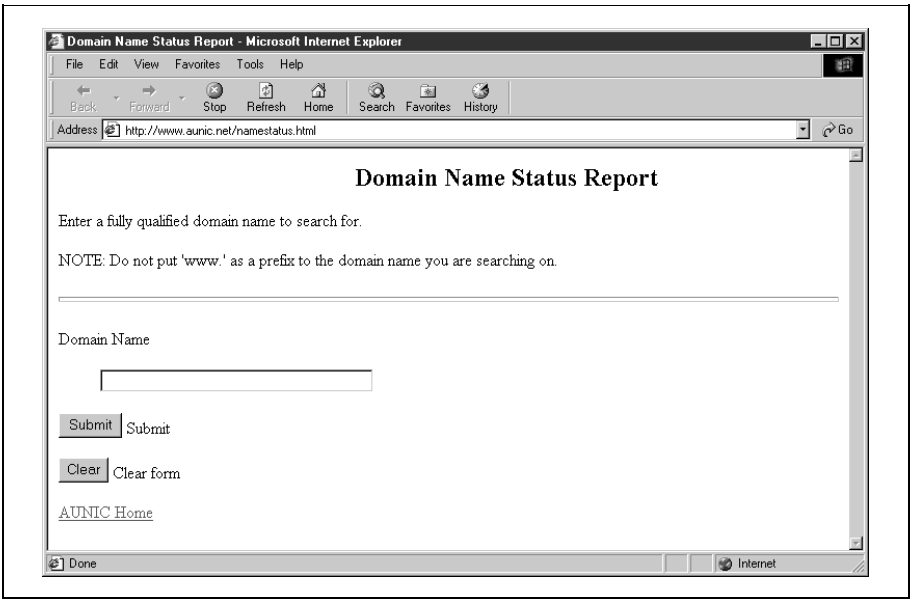

Рис. 3.2. *Веб-интерфейс whois-сервера домена au*

мацией по которому посетитель интересуется, а затем прозрачным образом посылает запрос нужному *whois*-серверу.

Очевидно, два этих сайта исключительно полезны для случаев, когда необходимо связаться с администрацией домена вне США.

Найдя нужный веб-сайт или нужного человека, мы сможем без особого труда найти и регистратора. За пределами США у большинства доменов по одному регистратору. При этом у некоторых доменов регистраторов много, например, у датского *dk* или доменов Великобритании *co.uk* и *org.uk*. Но и в этом случае описанный процесс приведет нас к нужным регистраторам.

Снова в Штаты

Будучи истинными космополитами, мы рассмотрели сначала международные домены. Что же делать тем, кто из добрых старых Соединенных Штатов?

В Соединенных Штатах принадлежность домена зависит в основном от того, чем занимается владеющая доменом организация, какие у нее предпочтения, касающиеся вида доменного имени, и сколько она готова платить. Если организация попадает в одну из следующих категорий, рекомендуется выбирать доменное имя в домене высшего уровня *us*:

- Школы K-12 (от детского сада до двенадцатого класса).

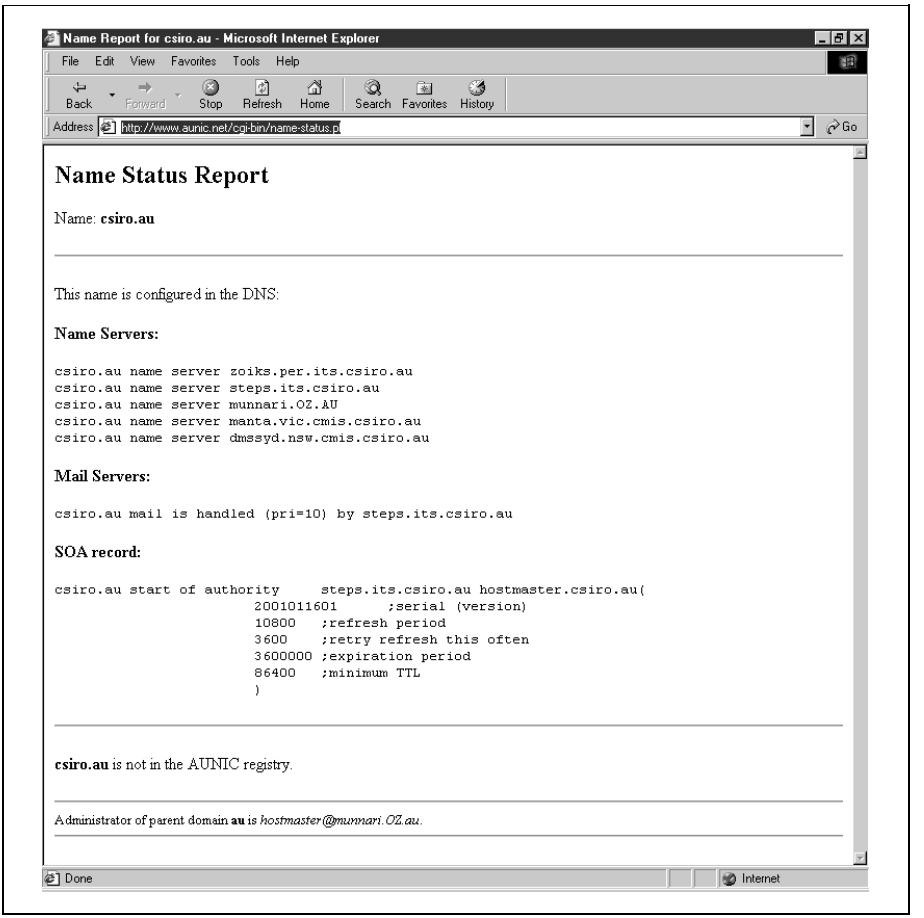

Рис. 3.3. *Информация о csiro.au, полученная от whois-сервера au*

- Средние учебные заведения и профессиональные технические училища.
- Федеральные и местные правительственные учреждения.

Даже в том случае, когда домен не попадает ни в одну из перечисленных категорий, но желательно, чтобы его имя отражало географическое расположение, и выглядело, например, как *acme.boulder.co.us*, можно зарегистрировать этот поддомен в домене высшего уровня *us*. Домен *us* делегирует поддомены, начиная с третьего уровня, сразу за «местностью» (обычно это город или округ); домены второго уровня имеют имена, соответствующие двухбуквенным обозначениям штатов, принятым почтовой службой США (напомним, мы уже рассказывали об этой особенности в разделе «Пространство доменных имен сети Интернет» в главе 2 «Как работает DNS»). Так, к примеру, если существует потребность в домене, который объединял бы два узла, со-

единенных сетью в вашем подвале где-нибудь в Колорадо-Спрингс, то можно зарегистрировать имя *toms-basement.colorado-springs.co.us*.

Есть еще и вопрос стоимости. Обычно оказывается дешевле зарегистрировать поддомен в домене высшего уровня *us*, чем в доменах *com*, *net* или *org*; иногда первый вариант вообще бесплатен.

Более подробную информацию о структуре домена *us* и правилах, которым он подчиняется, можно найти на веб-сайте NIC-центра США, по адресу *http://www.nic.us*.

Разумеется, американцы могут зарегистрироваться в любом из родовых доменов высшего уровня, будь то *com*, *net* или *org*. Требовать получения уже занятого доменного имени бесполезно, во всех остальных случаях проблем возникать не должно. Регистрацию в родовых доменах высшего уровня мы рассмотрим чуть позже.

Домен us

Разберем теперь один пример, чтобы читатели получили представление о том, как изучать доменное пространство сектора *us* в поисках идеального доменного имени. Предположим, вы пытаетесь помочь детскому садику, в который ходит ваш сын. Садик расположен в Боулдер-Сити, штат Колорадо, и вам нужно зарегистрировать для него доменное имя.

Пользуясь доступом к одному из узлов Калифорнийского университета, который остался с тех времен, когда вы там учились, вы можете проверить существование домена для Боулдер-Сити. (Если такого доступа нет, но есть подключение к сети Интернет, можно точно так же пользоваться программой *nslookup* для обращения к известному DNS-серверу.)

```
% nslookup
Default Server:  boulder.colorado.edu
Address: 128.138.238.18, 128.138.240.1

> set type=ns            – Поиск DNS-серверов
> co.us.                 – в co.us
Default Server:  boulder.colorado.edu
Address:  128.138.238.18, 128.138.240.1

co.us   nameserver = VENERA.ISI.EDU
co.us   nameserver = NS.ISI.EDU
co.us   nameserver = RS0.INTERNIC.NET
co.us   nameserver = NS.UU.NET
co.us   nameserver = ADMII.ARL.MIL
co.us   nameserver = EXCALIBUR.USC.EDU
```

Итак, получены имена DNS-серверов *co.us*. Теперь изменим сервер на DNS-сервер *co.us*, к примеру *venera.isi.edu*, и проверим, существуют ли поддомены (помните, мы все еще работаем в сеансе *nslookup*):

```
> server venera.isi.edu.  – Изменить сервер на venera.isi.edu
Default Server:  venera.isi.edu
Address:  128.9.0.32

> ls -t co.us.   – Получить список для зоны co.us с целью поиска NS-
                   записей
 [venera.isi.edu]
$ORIGIN co.us.
@                       1W IN NS        NS.ISI.EDU.
                        1W IN NS        RS0.INTERNIC.NET.
                        1W IN NS        NS.UU.NET.
                        1W IN NS        ADMII.ARL.MIL.
                        1W IN NS        EXCALIBUR.USC.EDU.
                        1W IN NS        VENERA.ISI.EDU.
officemate1.monument    1W IN NS        ns1.direct.ca.
                        1W IN NS        ns2.direct.ca.
la-junta                1D IN NS        ns2.cw.net.
                        1D IN NS        usdns.beltane.com.
                        1D IN NS        usdns2.beltane.com.
morrison                1W IN NS        NS1.WESTNET.NET.
                        1W IN NS        NS.UTAH.EDU.
littleton               1W IN NS        NS1.WESTNET.NET.
                        1W IN NS        NS.UTAH.EDU.
mus                     1W IN NS        NS1.WESTNET.NET.
                        1W IN NS        NS.UTAH.EDU.
ci.palmer-lake          1W IN NS        DNS1.REGISTEREDSITE.COM.
                        1W IN NS        DNS2.REGISTEREDSITE.COM.
co.adams                1W IN NS        ns1.rockymtn.net.
                        1W IN NS        ns2.rockymtn.net.
[...]
```

Ага! Значит, все-таки *есть* жизнь в Колорадо! Присутствуют поддомены *la-junta*, *morrison*, *littleton*, *mus*, а также многие другие. Существует даже поддомен для Боулдер-Сити, который, как это ни удивительно, носит имя *boulder*:

```
boulder                 1W IN NS        NS1.WESTNET.NET.
                        1W IN NS        NS.UTAH.EDU.
```

Как теперь узнать координаты администратора домена *boulder.co.us*? Можно попробовать использовать службу *whois*, но *boulder.co.us* не является доменом высшего уровня какой-либо страны, как не является и поддоменом родового домена высшего уровня, так что многого таким путем не узнать. К счастью, NIC-центр США приводит перечень соответствующих адресов электронной почты для каждого поддомена третьего уровня домена *us* по адресу *http://www.isi.edu/in-notes/us-domain-delegated.txt*. Если требуемой информации нет в этом списке, попрежнему можно воспользоваться инструментом *nslookup* и найти SOA-запись для зоны *boulder.co.us* аналогично тому, как мы нашли такую запись для *csiro.au*. И хотя люди, отвечающие на почту, отправляемую по указанным в SOA-записях, сами могут не иметь отношения к

регистрации (технические и административные обязанности в пределах одной зоны могут быть разделены), они почти наверняка знают тех, кто таковое отношение имеет, и могут поделиться нужной информацией.

Вот как следует использовать *nslookup*, чтобы найти SOA-запись для *boulder.co.us*:

```
% nslookup
Default Server:  boulder.colorado.edu
Address:  128.138.238.18, 128.138.240.1

> set type=soa      – Поиск SOA-записи
> boulder.co.us.    – для boulder.co.us
Default Server:  boulder.colorado.edu
Address:  128.138.238.18, 128.138.240.1

boulder.co.us
         origin = ns1.westnet.net
         mail addr = cgarner.westnet.net
         serial = 200004101
         refresh = 21600 (6H)
         retry   = 1200 (20M)
         expire  = 3600000 (5w6d16h)
         minimum ttl = 432000 (5D)
```

Как и в случае с *csiro.au*, следует заменить первый символ «.» в поле *mail addr* на символ «@», прежде чем воспользоваться адресом. Так, *cgarner.westnet.net* превращается в *cgarner@westnet.net*.

Чтобы запросить делегирование поддомена в *boulder.co.us*, можно получить бланк регистрационного заявления по адресу *http://www.nic.us/cgi-bin/template.pl* и отправить заполненное заявление соответствующему человеку.

Однако в том случае, когда поддомен для «места» создания домена еще не существует, следует прочесть правила делегирования поддоменов для домена *us* по адресу *http://www.nic.us/register/locality.html* и лишь после этого заполнить регистрационную форму по адресу *http://www.nic.us/cgi-bin/template.pl*.

Родовые домены высшего уровня

Как уже говорилось ранее, существует множество причин, по которым может требоваться поддомен, входящий в один из родовых доменов высшего уровня, скажем, *com*, *net* или *org*: если речь идет о мульти- или транснациональной компании, то ей не помешает широкая известность или просто красивое имя, которое заканчивается на *com*. Рассмотрим короткий пример для выбора доменного имени в одном из родовых доменов высшего уровня.

Представим себе сетевого администратора научно-исследовательского института в городе Хопкинс, штата Миннесота, который только что

обзавелся подключением к сети Интернет через коммерческого провайдера интернет-услуг. У компании никогда не было ничего, кроме UUCP-соединений, так что она никаким образом не зарегистрирована в пространстве имен сети Интернет.

Поскольку дело происходит в США, есть выбор между доменом *us* и родовыми доменами высшего уровня. Однако научно-исследовательский институт известен всему миру, так что домен *us* – не очень хороший выбор. Лучше всего подойдет поддомен домена *com*.

Научно-исследовательский институт известен как Институт мудреных штуковин (The Gizmonic Institute), так что администратору приходит в голову логичная мысль, что доменное имя *gizmonics.com* вполне подойдет. Пользуясь доступом к одному из серверов Университета Миннесоты, он пытается проверить, не занято ли уже имя *gizmonics.com*:

```
% nslookup
Default Server:  ns.unet.umn.edu
Address:  128.101.101.101

> set type=any     – Поиск любых записей
> gizmonics.com.   – для gizmonics.com
Server:  ns.unet.umn.edu
Address:  128.101.101.101

gizmonics.com   nameserver = NS2.SFO.WENET.NET
gizmonics.com   nameserver = NS1.SFO.WENET.NET
```

Однако! Похоже, имя *gizmonics.com* уже занято (кто бы мог подумать?)[1]. Что ж, имя *gizmonic-institute.com* ненамного длиннее, но попрежнему достаточно понятное:

```
% nslookup
Default Server:  ns.unet.umn.edu
Address:  128.101.101.101

> set type=any                – Поиск любых записей
> gizmonic-institute.com.     – для gizmonic-institute.com
Server:  ns.unet.umn.edu
Address:  128.101.101.101

*** ns.unet.umn.edu can't find gizmonic-institute.com.: Non-existent host/
domain
```

К счастью, имя *gizmonic-institute.com* свободно, и можно переходить к следующему шагу – выбору регистратора.

---

[1] В действительности, имя *gizmonics.com* занято Джоэлом Ходжсоном (Joel Hodgson), человеком, который придумал Институт мудреных штуковин и Театр таинственной науки 3000.

Выбор регистратора

Выбрать регистратора? Добро пожаловать в дивный новый мир соревнования! Вплоть до весны 1999 года и регистром и регистратором для доменов *com*, *net*, *org* и *edu* являлась единственная компания – Network Solutions Inc. Чтобы зарегистрировать поддомен в любом из родовых доменов высшего уровня, следовало обращаться в Network Solutions.

В июне 1999 ICANN, организация, управляющая доменным пространством (мы говорили о ней в предыдущей главе), привнесла элемент соревнования в функции регистрации для доменов *com*, *net* и *org*. Сегодня существует выбор из десятков регистраторов для доменов *com*, *net* и *org*. Их список доступен по адресу *http://www.internic.net/regist.html*.

Не желая давать советы по выбору регистратора, заметим, что следует взглянуть на цены и предоставляемые регистратором дополнительные услуги, которые могут быть полезны. Возможно, удастся заключить выгодную сделку по регистрации с откатом в алюминии, например.

Проверка регистрации сети

Прежде чем двинуться дальше, необходимо проверить зарегистрирована ли ваша IP-сеть (или сети, если их несколько). Некоторые регистраторы не делегируют поддомен DNS-серверам, расположенным в незарегистрированных сетях, а сетевые регистры (о которых мы скоро поговорим) не делегируют зону в домене *in-addr.arpa*, соответствующую незарегистрированной сети.

IP-сеть определяет диапазон IP-адресов. К примеру, сеть 15/8 состоит из всех IP-адресов диапазона от 15.0.0.0 до 15.255.255.255. Сеть 199.10.25/24 начинается с адреса 199.10.25.0 и заканчивается адресом 199.10.25.255.

Организация InterNIC некогда была официальным арбитром всех IP-сетей: она присваивала IP-сети сегмент адресов в сети, подключенной к Интернету, и следила за тем, чтобы эти сегменты не пересекались. В наши дни прежняя роль организации InterNIC во многом переложена на плечи многочисленных провайдеров интернет-услуг, которые для пользовательских целей выделяют пространство в своих собственных, уже существующих сетях. Если вам известно, что ваша сеть построена на адресах провайдера, более крупная сеть уже скорее всего зарегистрирована (этим же провайдером). Разумеется, есть смысл перепроверить факт регистрации провайдером своей сети, но в случае, если таковое действие произведено не было, вы ничего не захотите (и не сможете) сделать, ну разве что на провайдера поворчать. Уточнив положение с регистрацией, можно пропустить оставшуюся часть этого раздела и читать дальше.

CIDR

Когда-то давным-давно, когда мы работали над первым изданием этой книги, 32-битное адресное пространство сети Интернет было разделено на три основных класса сетей: класс A, класс B и класс C. Сети класса A имели то свойство, что первый октет (восемь битов) IP-адреса определял собственно сеть, а оставшиеся биты использовались организацией, которая управляла сетью, для того, чтобы различать узлы сети. Большинство организаций, управляющих сетями класса A, разделяли их на подсети, добавляя к схеме адресации еще один уровень иерархии. В сетях класса B два первых октета использовались для определения сети, а оставшиеся два – для определения отдельных узлов, а в сетях класса C для определения сети отводилось три октета и лишь один для определения узлов.

К сожалению, эта система мелких, средних и крупных сетей не всегда была удобна. Многие организации были достаточно велики, чтобы выйти за пределы сети класса C, которая могла содержать максимум 254 узла, но недостаточно велики, чтобы занять целый класс B, сеть которого могла вместить 65534 узла. Но многие из этих организаций получили все-таки сети класса B в свое распоряжение. Как следствие, свободные сети класса B были занесены в красную книгу.

Для решения этой проблемы и создания сетей, которые имели бы соответствующий требованиям размер, была разработана *бесклассовая междоменная маршрутизация* (Classless Inter-Domain Routing, или CIDR, произносится как «сайдр»). Как видно из названия, CIDR избавляется от классов A, B и C. В системе CIDR для идентификации сети может использоваться не фиксированное число октетов (один, два или три), но любое число битов IP-адреса. Так, к примеру, если организации нужно адресное пространство примерно в четыре раза большее, чем адресное пространство сети класса B, власть предержащие могут определить длину идентификатора сети в 14 битов, таким образом, оставляя 18 битов (в четыре раза больше узлов, чем в сети класса B) на используемое адресное пространство.

Совершенно естественно, что пришествие CIDR сделало «классовую» терминологию устаревшей, хотя она до сих пор довольно часто используется в разговорах. Итак, чтобы обозначить конкретную CIDR-сеть, следует указать конкретное значение старших битов, присваиваемое организации в записи через точку, а также число битов, определяющих сеть. Две части записи разделяются символом «слэш». 15/8 – прежняя сеть класса A, которая «начинается» с восьмибитной последовательности 00001111. Прежняя сеть класса B 128.32.0.0 теперь идентифицируется как 128.32/16. А сеть 192.168.0.128/25 состоит из 128 IP-адресов, начиная с адреса 192.168.0.128 и заканчивая адресом 192.168.0.255.

Но если ваша сеть была создана с помощью InterNIC в давние времена либо вы сами являетесь организацией-провайдером, следует проверить, зарегистрирована ли сеть. Как это сделать? Ну конечно, обратившись в те самые организации, которые занимаются регистрацией сетей. Эти организации называются сетевыми регистраторами (а как же еще?) и отвечают за регистрацию сетей в различных частях планеты. В западном полушарии выделением IP-адресного пространства и регистрацией сетей занимается организация ARIN (American Registry of Internet Numbers), *http://www.arin.net*. За Азию и Тихоокеанский бассейн отвечает APNIC (Asia Pacific Network Information Center), *http://www.apnic.net*. В Европе это сетевой координационный центр RIPE (*http://www.ripe.net*). Каждый регистратор может делегировать свои полномочия в определенном регионе; к примеру, ARIN делегирует регистрационные полномочия по Мексике и Бразилии сетевым регистраторам каждой из этих стран. Не забудьте уточнить, кто является сетевым регистратором для вашей страны.

В случае сомнений в регистрации сети наилучший способ уточнить этот вопрос – воспользоваться службой *whois*, которая предоставляется сетевыми регистраторами, и просто поискать свою сеть. Вот URL-адреса *whois*-страниц сетевых регистраторов:

ARIN

http://www.arin.net/whois/index.html

APNIC

http://whois.apnic.net

RIPE

http://www.ripe.net/cgi-bin/whois

Обнаружив, что ваша сеть не зарегистрирована, вы должны зарегистрировать ее до получения зоны в *in-addr.arpa*. У каждого регистратора своя процедура регистрации сетей, но в основном этот процесс заключается в смене владельца денег (к сожалению, в данном случае деньги уходят из ваших рук).

Вы можете обнаружить, что ваша сеть уже зарегистрирована провайдером интернет-услуг. В этом случае нет необходимости в независимой регистрации.

Итак, все ваши узлы, подключенные к сети Интернет, находятся в зарегистрированных сетях; настало время регистрации зон.

Регистрация зон

Различные регистраторы предлагают различные правила и процедуры регистрации, но на данном этапе большинство из них предлагают регистрацию в online-режиме, на собственных веб-страницах. Посколь-

ку ранее в этой главе мы уже занимались выбором регистраторов, будем считать, что нужные адреса читателям уже известны.

Основная информация, требуемая каждым регистратором, – это доменные имена и адреса DNS-серверов, а также минимальное количество сведений о клиенте, которое требуется, чтобы послать счет или произвести транзакцию с кредитной картой. Если вы не подключены к сети Интернет, сообщите регистратору адреса узлов Интернета, которые используются вашими DNS-серверами. Некоторые регистраторы требуют наличия рабочих DNS-серверов для вашей зоны. (Все остальные могут задавать вопросы о том, сколько времени вам потребуется, чтобы привести ваши DNS-серверы в полностью рабочее состояние.) В таком случае переходите к главе 4 на предмет установки DNS-серверов, а затем связывайтесь с регистратором и сообщайте ему нужную информацию.В большинстве случаев требуется также предоставить некоторые сведения об организации, регистрирующей зону, включая адреса администратора и лица, отвечающего за технические моменты, связанные с зоной (это может быть один и тот же человек). Если эти люди еще не внесены в *whois*-базу регистратора, необходимо будет предоставить информацию для такой регистрации. Обычно эта информация включает имена, традиционные почтовые адреса, номера телефонов и адреса электронной почты. Если эти люди уже зарегистрированы в службе *whois*, достаточно просто указать их уникальный *whois*-идентификатор при регистрации.

Существует еще один аспект регистрации новой зоны, о котором следует упомянуть: цена. Большинство регистраторов являются коммерческими предприятиями, и берут деньги за регистрацию доменных имен. Компания Network Solutions, самый старый регистратор в доменах *com*, *net* и *org*, взимает $35 в год за регистрацию поддомена в родовом домене высшего уровня. (Если у вас уже есть поддомен в *com*, *net* или *org*, а счет от Network Solutions в последнее время на приходил, неплохо бы перепроверить свою контактную информацию в службе *whois*, чтобы удостовериться, что компании известен ваш текущий адрес и телефонный номер.)

Если вы имеете прямое подключение к сети Интернет, следует также проверить, что зоны *in-addr.arpa*, соответствующие вашим IP-сетям, делегированы вам же. К примеру, если ваша компания получила в свое распоряжение сеть 192.201.44/24, вам придется управлять зоной *44.201.192.in-addr.arpa*. Таким образом, вы сможете контролировать отображение IP-адресов в имена для узлов этой сети. Настройка для зон *in-addr.arpa* также описана в главе 4.

В предыдущем разделе («Проверка регистрации сети») мы просили читателей выяснить, является ли их сеть частью сети провайдера интернет-услуг? Зарегистрирована ли ваша сеть, или сеть, частью которой она является? Через какого сетевого регистратора? Ответы на эти

вопросы понадобятся, чтобы осуществить делегирование вам зон *in-addr.arpa*.

Если ваша сеть является частью сети, зарегистрированной провайдером интернет-услуг, следует связаться с этим провайдером, чтобы соответствующие поддомены были делегированы вам в виде зон *in-addr.arpa*. Каждый провайдер проводит подготовку к делегированию *in-addr.arpa* по-своему. Веб-сайт провайдера является неплохим источником информации по этому процессу. Если нужной информации нет на веб-сайте, попытайтесь найти SOA-запись для зоны *in-addr.arpa*, которая соответствует сети вашего провайдера. К примеру, если ваша сеть является частью сети 153.35/16 UUNET, можно поискать SOA-запись для *35.153.in-addr.arpa* в целях нахождения адресов электронной почты техподдержки этой зоны.

Если ваша сеть зарегистрирована напрямую через одного из региональных сетевых регистраторов, для регистрации соответствующей зоны *in-addr.arpa* следует связаться с этой организацией. Каждый сетевой регистратор предоставляет на своем веб-сайте информацию о процессе делегирования.

Теперь, зарегистрировав свои зоны, читатели могут заняться наведением порядка у себя дома. Предстоит установка DNS-серверов, и в следующей главе мы поделимся подробностями этого действа.

- *Наша зона*
- *Создание данных для зоны*
- *Создание файла настройки BIND*
- *Сокращения*
- *Проверка имени узла (BIND 4.9.4 и более поздние версии)*
- *Инструменты*
- *Запуск первичного мастер-сервера DNS*
- *Запуск вторичного DNS-сервера*
- *Добавление зон*
- *Что дальше?*

4

Установка BIND

– Очень милые стишки, – сказала Алиса задумчиво, – но понять их не так-то легко. (Знаешь, ей даже самой себе не хотелось признаться, что она ничего не поняла.)
– Наводят на всякие мысли – хоть я и не знаю, на какие...

Тем из читателей, кто прилежно прочитал предыдущие главы, вероятно, не терпится обзавестись наконец-то работающим DNS-сервером. В этой главе мы расскажем, как это сделать. Давайте установим пару DNS-серверов. К тому же кто-то, быть может, посмотрел в оглавление и открыл книгу прямо на этой главе (как вам не стыдно!). Если вы принадлежите к последним, имейте в виду, что в этой главе используются концепции из предшествующих, и для дальнейшего чтения вам они должны быть полностью ясны.

Существует несколько факторов, влияющих на особенности установки DNS-серверов. Самый главный – тип доступа к Интернету: полный доступ (например, доступна служба FTP и узел *ftp.uu.net*), ограниченный доступ (путь в сеть ограничен брандмауэром), либо вовсе никакого доступа. В этой главе подразумевается, что есть полный доступ, прочие же случаи обсуждаются к главе 11 «Безопасность».

В этой главе, в качестве примера, которому читатели могут следовать при создании собственных зон, мы установим два DNS-сервера для не-

скольких воображаемых зон. Темы, которые будут затронуты, изложены достаточно подробно, чтобы у читателей не возникало затруднений при установке своих первых серверов. Последующие главы восполняют пробелы и содержат дальнейшие подробности. В случае наличия работающих DNS-серверов, вы можете быстро пробежаться по этой главе, чтобы ознакомиться с терминологией, которую мы используем, либо просто убедиться, что вы ничего не пропустили при установке своих DNS-серверов.

Наша зона

Наша выдуманная зона будет обслуживать колледж. Кинематографический университет занимается изучением всех аспектов киноиндустрии и разрабатывает новейшие способы распространения кинопродукции. Один из наиболее многообещающих проектов включает исследования по использованию IP в качестве транспорта для распространения фильмов. Посетив веб-сайт нашего регистратора, мы остановились на доменном имени *movie.edu*. Недавно полученный грант позволил нам подключиться к сети Интернет.

В Кинематографическом университете в настоящее время существует две Ethernet-сети, а также планы по добавлению еще одной или двух. Эти сети имеют сетевые номера 192.249.249/24 и 192.253.253/24. Часть нашей таблицы узлов содержит следующие записи:

```
127.0.0.1      localhost

# Это наши убойные компы[1]

192.249.249.2  robocop.movie.edu robocop
192.249.249.3  terminator.movie.edu terminator bigt
192.249.249.4  diehard.movie.edu diehard dh

# Эти машины наводят ужас (или сами находятся
# в ужасающем состоянии), и скоро их предстоит заменить[2]

192.253.253.2  misery.movie.edu misery
192.253.253.3  shining.movie.edu shining
192.253.253.4  carrie.movie.edu carrie

# Червоточина (wormhole) - выдуманное явление, которое позволяет осуществить
# мгновенную транспортировку космических путешественников на огромные
# расстояния; оно до сих пор не признано стабильным. Единственная разница
# между червоточинами и маршрутизаторами состоит в том, что маршрутизаторы
```

[1] Компьютеры названы в честь главных героев известных боевиков: «Робот-полицейский», «Терминатор» и «Крепкий Орешек». – *Примеч. ред.*

[2] А эти компьютеры названы именами героев фильмов-ужасов, снятых по романам Стивена Кинга: «Мизери», «Сияющий», «Воспламеняющая взглядом». – *Примеч. ред.*

```
# не транспортируют пакеты мгновенно, а в особенности наши маршрутизаторы.

192.249.249.1  wormhole.movie.edu wormhole wh wh249
192.253.253.1  wormhole.movie.edu wormhole wh wh253
```

Сама сеть изображена на рис. 4.1.

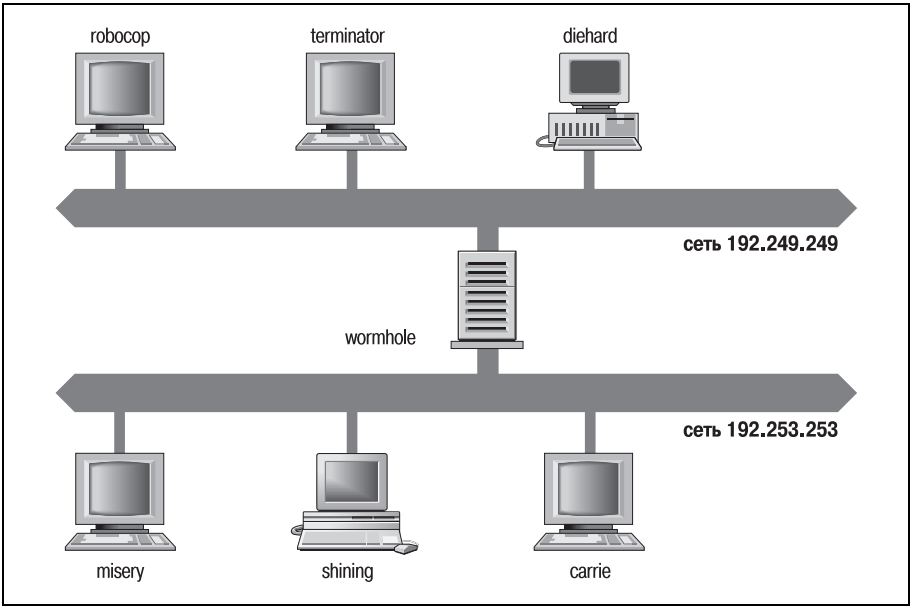

Рис. 4.1. *Сеть Кинематографического университета*

Создание данных для зоны

Первый шаг в установке DNS-серверов для Университета – преобразовать таблицу узлов в эквивалентные зональные данные. В DNS-версии данные разбиты по нескольким файлам. Один из файлов содержит отображения имен узлов в адреса. Прочие файлы содержат отображения адресов обратно в имена. Поиск адреса для имени иногда называется *прямым отображением*, а поиск имени по адресу – *обратным отображением*. Каждая сеть имеет собственный файл с данными для обратного отображения.

В этой книге мы будем пользоваться следующим: файл, содержащий данные преобразования имен хостов в адреса, носит имя вида *db.DOMAIN*. Для домена *movie.edu* этот файл называется *db.movie.edu*. Соответственно, файлы, содержащие данные преобразования адресов в имена хостов, носят имена вида *db.ADDR*, где ADDR – номер сети без последних нулей или маски сети. В нашем примере файлы будут называться *db.192.249.249* и *db.192.253.253*; по одному файлу на каждую

сеть. Префикс *db* – это сокращение для базы данных (database). Этот набор файлов *db.DOMAIN* и *db.ADDR* мы будем называть *файлами данных зоны*. Существуют и другие файлы данных зоны: *db.cache* и *db.127.0.0*. Это своего рода нагрузка. Каждый DNS-сервер должен иметь такие файлы, и для каждого сервера они более или менее похожи.

Чтобы связать файлы данных зоны, DNS-серверу требуется файл настройки – в BIND версии 4 этот файл обычно называется */etc/named.boot*. В BIND версий 8 и 9 файл обычно называется */etc/named.conf*. Формат файлов данных зон одинаков во всех реализациях DNS, он называется *форматом мастер-файла*. Однако формат файлов настройки является специфичным для реализации DNS-сервера – в нашем случае для DNS-сервера BIND.

Файлы данных зоны

Большинство записей в файлах данных зоны называются *RR-записями* DNS. При поиске DNS не обращает внимания на регистр символов, так что имена в файлах данных зоны можно набирать в произвольном регистре, даже в смешанном. Мы практически всегда используем строчные буквы. Несмотря на то, что поиск нечувствителен к регистру символов, регистр сохраняется при возвращении результатов. Таким образом, если добавить к файлам данных зоны записи с именем *Tootsie.movie.edu*, результаты поиска для *tootsie.movie.edu* будут содержать эти записи, но с заглавной буквой «T» в имени домена.

RR-записи должны начинаться с первой позиции строки. RR-записи в примерах, приводимых в этой книге, начинаются с первой позиции, а видимые отступы появляются из-за своеобразного форматирования книги. В RFC-документах по DNS RR-записи в примерах приводятся в определенном порядке. Многие люди следуют этому порядку (и мы не исключение), но такой порядок следования записей не является обязательным. Итак, выбранный порядок следования записей:

SOA-запись

Указывает на *авторитативность* для зоны

NS-запись

Перечисляет *DNS-серверы* зоны

Прочие записи

Данные об узлах зоны

Из прочих записей в данной главе рассмотрены:

A

Отображение имен узлов в адреса

PTR

Отображение адресов в имена узлов

CNAME

Каноническое имя (для псевдонимов)

Те из читателей, кто имеет опыт общения с форматом мастер-файла, наверняка при виде наших данных произнесут «Было бы гораздо короче написать вот так...». Мы не используем сокращения при создании данных для зоны (во всяком случае, поначалу), чтобы читатели до конца поняли полный синтаксис каждого типа RR-записи. Когда полная версия будет всем понятна, мы вернемся и «сожмем» наши файлы.

Комментарии

Файлы данных зоны легче читать, если они содержат комментарии и пустые строки. Комментарии начинаются с символа точки с запятой (;) и заканчиваются концом строки. Как вы, вероятно, догадались, DNS-сервер игнорирует комментарии и пустые строки.

Установка стандартного значения TTL для зоны

Прежде чем начать создание файлов данных зоны, необходимо выяснить, какой версией BIND мы будем пользоваться. Версия имеет значение, поскольку способ задания стандартного значения времени жизни (TTL, time to live) для зоны изменился в BIND версии 8.2. В предшествующих версиях значение TTL по умолчанию определялось последним полем SOA-записи. Но непосредственно перед выходом BIND версии 8.2 был опубликован документ RFC 2308, который изменил значение последнего поля SOA-записи на *время жизни отрицательного кэширования*. Этот показатель определяет, как долго удаленные DNS-серверы имеют право сохранять информацию об *отрицательных ответах*, связанных с зоной, то есть ответах, суть которых заключается в том, что доменное имя или тип данных не существует в конкретном домене.

Как установить значение TTL по умолчанию в BIND 8.2 и более поздних? С помощью новой директивы – $TTL. $TTL позволяет указать время жизни для всех записей в файле, которые следуют за этой директивой (но предшествуют любым другим директивам $TTL) и не имеют явно заданного времени жизни.

Сервер имен передает указанное значение TTL вместе с ответами на запросы, что позволяет другим DNS-серверам кэшировать полученные данные на указанный интервал времени. Если данные изменяются не часто, разумным значением для времени жизни будет интервал в несколько дней. Неделя – наибольший разумный интервал. Можно использовать значение в один час, но, как правило, не рекомендуется использовать еще более короткие интервалы, из-за объема DNS-трафика, который они провоцируют.

Поскольку мы используем новую версию пакета BIND, необходимо установить значение TTL по умолчанию для наших зон с помощью опе-

ратора $TTL. Нам кажется, что три часа – вполне разумное значение, так что мы начинаем файлы данных зоны со строчки:

```
$TTL 3h
```

Если используется DNS-сервер более старый, чем BIND версии 8.2, не пытайтесь использовать оператор $TTL, поскольку DNS-сервер его не опознает и посчитает за ошибку.

SOA-записи

Следующая часть (или первая – для DNS-серверов более старых, чем BIND версии 8.2) в каждом из наших файлов – SOA-запись (RR-запись типа SOA). SOA-запись показывает, что данный DNS-сервер является самым надежным источником информации в пределах этой зоны. Наш DNS-сервер является авторитативным для зоны *movie.edu* по причине наличия SOA-записи. SOA-запись должна присутствовать в каждом файле *db.DOMAIN* и *db.ADDR*. В файле данных зоны может присутствовать одна и только одна SOA-запись.

Мы добавили следующую SOA-запись к файлу *db.movie.edu*:

```
movie.edu. IN SOA terminator.movie.edu. al.robocop.movie.edu. (
                        1        ; Порядковый номер
                        3h       ; Обновление через 3 часа
                        1h       ; Повторение попытки через 1 час
                        1w       ; Устаревание через 1 неделю
                        1h )     ; Отрицательное TTL в 1 час
```

Имя *movie.edu.* должно начинаться в первой позиции строки файла. Убедитесь, что имя заканчивается точкой, как в этом примере, иначе результат вас сильно удивит! (Чуть позже в этой главе мы объясним, как именно.)

Буквы IN обозначают Internet. Это один из классов данных – существуют и другие, но ни один из них в настоящее время широко не применяется. В наших примерах встречается только класс IN. Класс можно не указывать. Если класс не указан, DNS-сервер определяет его по выражению, диктующему чтение файла данных в файле настройки; чуть позже мы рассмотрим и это выражение.

Первое имя после SOA (*terminator.movie.edu.*) – это имя первичного мастер-сервера DNS зоны *movie.edu*. Второе имя (*al.robocop.movie.edu.*) – это адрес электронной почты человека, управляющего зоной; чтобы использовать адрес, следует заменить первый символ «.» на символ «@». Часто можно увидеть имена *root*, *postmaster* или *hostmaster* в почтовых адресах. Серверы имен не пользуются этими адресами, поскольку они предназначены для использования людьми. Если возникла проблема, имеющая отношение к зоне, всегда можно отправить сообщение по указанному почтовому адресу. Версии BIND, начиная с 4.9,

предоставляют для указания такого адреса специальный тип RR-записей, RP (responsible person, ответственное лицо). Записи типа RP рассматриваются в главе 7 «Работа с BIND».

Скобки позволяют растягивать SOA-запись на несколько строк. Большинство полей в SOA-записи используются вторичными DNS-серверами, поэтому они будут нами изучены, когда мы дойдем до рассмотрения вторичных серверов в этой главе. Пока просто будем считать, что выбрали разумные значения полей.

Аналогичные SOA-записи мы добавляем в файлы *db.192.249.249* и *db.192.253.253*. В этих файлах первое имя в SOA-записи изменяется с *movie.edu.* на имя соответствующей зоны *in-addr.arpa*: *249.249.192.in-addr.arpa.* и *253. 253.192.in-addr.arpa.*

NS–записи

Следующие части, которые мы добавляем к каждому из файлов, – это NS-записи (name server, DNS-сервер). По одной NS-записи на каждый DNS-сервер, который является авторитативным для нашей зоны. Вот NS-записи из файла *db.movie.edu*:

```
movie.edu.  IN NS  terminator.movie.edu.
movie.edu.  IN NS  wormhole.movie.edu.
```

Эти записи указывают, что существует два DNS-сервера для зоны *movie.edu*. Серверы запущены на узлах *terminator.movie.edu* и *wormhole.movie.edu*. Хосты, входящие одновременно в несколько сетей, скажем, *wormhole.movie.edu*, идеально подходят на роли DNS-серверов, поскольку имеют «правильное подключение». Такой узел напрямую доступен узлам более чем одной сети, а если является маршрутизатором, то находится под пристальным наблюдением со стороны администратора. Более подробно размещение DNS-серверов мы рассмотрим в главе 8 «Развитие домена».

Как и в случае с SOA-записью, NS-записи мы добавляем также к файлам *db.192.249.249* и *db.192.253.253*.

RR–записи адресов и псевдонимов

Теперь мы создадим отображения имен в адреса путем добавления следующих RR-записей в файл *db.movie.edu*:

```
;
; Адреса узлов
;
localhost.movie.edu.  IN A     127.0.0.1
robocop.movie.edu.    IN A     192.249.249.2
terminator.movie.edu. IN A     192.249.249.3
diehard.movie.edu.    IN A     192.249.249.4
```

```
misery.movie.edu.     IN A     192.253.253.2
shining.movie.edu.    IN A     192.253.253.3
carrie.movie.edu.     IN A     192.253.253.4
;
; Жители нескольких сетей
;
wormhole.movie.edu.   IN A     192.249.249.1
wormhole.movie.edu.   IN A     192.253.253.1
;
; Псевдонимы
;
bigt.movie.edu.       IN CNAME terminator.movie.edu.
dh.movie.edu.         IN CNAME diehard.movie.edu.
wh.movie.edu.         IN CNAME wormhole.movie.edu.
wh249.movie.edu.      IN A     192.249.249.1
wh253.movie.edu.      IN A     192.253.253.1
```

Первые два блока записей вряд ли кого-то удивят. Буква A обозначает адрес, а каждая RR-запись определяет отображение имени в соответствующий адрес. *wormhole.movie.edu* является маршрутизатором. Этот узел имеет два адреса, привязанных к имени, а следовательно – и две адресные записи. В отличие от поиска по таблице узлов, поиск DNS может возвращать несколько адресов для имени; так, поиск адреса *wormhole.movie.edu* возвращает два результата. Если автор запроса и DNS-сервер расположены в одной сети, некоторые DNS-серверы в целях повышения эффективности сетевого обмена возвращают более «близкий» адрес первым. Эта возможность носит название *сортировки адресов* и описана в главе 10 «Дополнительные возможности». Если сортировка адресов неприменима, адреса подвергаются *круговой перестановке* между запросами, так что в последующих ответах они перечисляются в отличающемся порядке. Эта круговая система («round robin») впервые появилась в BIND версии 4.9.

Третий блок содержит псевдонимы таблицы узлов. Для первых трех псевдонимов мы создали CNAME-RR-записи (canonical names, записи канонических имен). Однако для двух других псевдонимов мы создали адресные записи (почему – расскажем через несколько строк). CNAME-запись определяет отображение псевдонима в каноническое имя узла. Сервер имен работает с записями типа CNAME совершенно не так, как обычно происходит работа с псевдонимами из таблицы узлов. При поиске имени, если DNS-сервер находит CNAME-запись, то имя узла заменяется каноническим именем, после чего поиск продолжается уже для этого имени. К примеру, когда сервер обрабатывает запрос для имени *wh.movie.edu*, то находит CNAME-запись, которая указывает на *wormhole.movie.edu*. Сервер производит поиск *wormhole.movie.edu* и возвращает оба адреса.

Следует запомнить одну вещь, которая касается псевдонимов вроде *bigt.movie.edu* – они никогда не должны появляться в правой части

RR-записей. Иначе говоря, в части данных RR-записи следует всегда использовать канонические имена (скажем, *terminator.movie.edu*). Обратите внимание, что в свежесозданных NS-записях используются именно канонические имена.

Две последних записи призваны решить специфическую проблему. Предположим, что необходимо проверить работу одного из интерфейсов маршрутизатора, например, *wormhole.movie.edu*. Одним из часто применяемых способов диагностики является использование *ping* в целях проверки рабочего состояния интерфейса. Если попытаться использовать *ping* для имени *wormhole.movie.edu*, DNS-сервер вернет оба адреса узла. *ping* использует первый адрес из списка. Но какой адрес является первым?

Если бы речь шла о таблице узлов, мы бы выбирали между именами *wh249.movie.edu* и *wh253.movie.edu*; и каждому имени соответствовал бы *единственный* адрес этого узла. Чтобы получить эквивалентную возможность в DNS, не следует создавать псевдонимы (CNAME-записи) для *wh249.movie.edu* и *wh253.movie.edu*, т. к. это приведет к тому, что при поиске по псевдониму будут возвращаться оба адреса *wormhole.movie.edu*. Вместо этого следует использовать адресные записи. Таким образом, чтобы проверить работу интерфейса 192.253.253.1 на узле *wormhole.movie.edu*, мы выполняем команду *ping wh253.movie.edu*, поскольку это имя соответствует единственному адресу. То же справедливо и для *wh249.movie.edu*.

Сформулируем основное правило: если узел имеет интерфейсы с более чем одной сетью, следует создать по одной адресной (A) записи для каждого уникального псевдонима адреса. CNAME-записи служат для создания псевдонимов, являющихся общими для всех адресов.

При этом не стоит рассказывать пользователям об именах *wh249.movie.edu* и *wh253.movie.edu*. Эти имена предназначены только для служебного использования. Если пользователи привыкнут использовать имена вроде *wh249.movie.edu*, у них возникнут проблемы, поскольку это имя будет работать не во всех контекстах (скажем, не будет работать для файлов *.rhosts*). Происходить это будет потому, что в некоторых контекстах требуется имя, которое является результатом поиска по адресу, то есть каноническое имя *wormhole.movie.edu*.

Мы использовали адресные (A) записи для создания псевдонимов *wh249.movie.edu* и *wh253.movie.edu*, и у читателей может возникнуть вопрос: «Можно ли использовать адресные записи вместо CNAME-записей во *всех* случаях?». У большинства приложений использование адресных записей вместо CNAME-записей затруднений не вызывает, поскольку их интересует только соответствующий IP-адрес. Но есть одно приложение, а именно, *sendmail*, поведение которого изменяется. *Sendmail* обычно заменяет псевдонимы в заголовках почтовых сообщений соответствующими каноническими именами; и канонизация происходит только в том случае, если для имен, указанных в заголов-

ке, существуют CNAME-данные. Если не использовать CNAME-записи для создания псевдонимов, приложению *sendmail* придется рассказать обо всех возможных псевдонимах, под которыми может быть известен узел, а это потребует дополнительных усилий по настройке *sendmail*.

Помимо проблем *sendmail*, проблемы могут возникать и у пользователей, которым понадобится выяснить каноническое имя узла для записи его в файл *.rhosts*. Поиск по псевдониму, для которого существует CNAME-запись, вернет каноническое имя, но если существует только адресная запись, этого не произойдет. В таком случае пользователям для получения канонического имени *придется* производить поиск по IP-адресу, как это делает программа *rlogind*, но такие умные пользователи никогда не встречаются в системах, которые мы администрируем.

PTR-записи

Теперь мы создадим отображения адресов в имена. Файл *db.192.249.249* содержит отображения адресов в имена узлов для сети 192.249.249/24. Для такого отображения используются RR-записи DNS, которые носят название PTR-записей или записей-указателей (pointer records). Для каждого сетевого интерфейса хоста присутствует ровно одна запись. (Вспомним, что поиск по адресам в DNS – это, по сути дела, поиск по именам. Адрес инвертируется и к нему добавляется имя *in-addr.arpa*.)

Вот PTR-записи, которые мы добавили для сети 192.249.249/24:

```
1.249.249.192.in-addr.arpa.  IN PTR wormhole.movie.edu.
2.249.249.192.in-addr.arpa.  IN PTR robocop.movie.edu.
3.249.249.192.in-addr.arpa.  IN PTR terminator.movie.edu.
4.249.249.192.in-addr.arpa.  IN PTR diehard.movie.edu.
```

Здесь есть пара моментов, на которые стоит обратить внимание читателей. Во-первых, каждый адрес должен указывать на единственное имя – каноническое. Так, адрес 192.249.249.1 отображается в *wormhole.movie.edu*, но не в *wh249.movie.edu*. *Допускается* создание двух PTR-записей, одной для *wormhole.movie.edu* и одной для *wh249.movie.edu*, но большинство систем не приспособлены для того, чтобы увидеть более одного имени, соответствующего адресу. Во-вторых, несмотря на то, что узел *wormhole.movie.edu* имеет два адреса, здесь указан только один из них. Это происходит потому, что файл отображает только прямые соединения с сетью 192.249.249/24, и узел *wormhole.movie.edu* имеет только одно такое соединение.

Аналогичные данные мы создаем для сети 192.253.253/24.

Полные файлы данных зоны

Итак, рассказав о различных типах RR-записей в файлах данных зоны, мы покажем, как эти файлы выглядят полностью. Вспомните, что порядок следования записей в действительности не имеет значения.

Вот содержимое файла *db.movie.edu*:

```
$TTL 3h
movie.edu. IN SOA terminator.movie.edu. al.robocop.movie.edu. (
                            1       ; Порядковый номер
                            3h      ; Обновление через 3 часа
                            1h      ; Повторение попытки через 1 час
                            1w      ; Устаревание через 1 неделю
                            1h )    ; Отрицательное TTL в 1 час

;
; Серверы имен
;
movie.edu.  IN NS  terminator.movie.edu.
movie.edu.  IN NS  wormhole.movie.edu.

;
; Адреса для канонических имен
;
localhost.movie.edu.  IN A     127.0.0.1
robocop.movie.edu.    IN A     192.249.249.2
terminator.movie.edu. IN A     192.249.249.3
diehard.movie.edu.    IN A     192.249.249.4
misery.movie.edu.     IN A     192.253.253.2
shining.movie.edu.    IN A     192.253.253.3
carrie.movie.edu.     IN A     192.253.253.4
wormhole.movie.edu.   IN A     192.249.249.1
wormhole.movie.edu.   IN A     192.253.253.1

;
; Псевдонимы
;
bigt.movie.edu.       IN CNAME terminator.movie.edu.
dh.movie.edu.         IN CNAME diehard.movie.edu.
wh.movie.edu.         IN CNAME wormhole.movie.edu.

;
; Специальные имена интерфейсов
;
wh249.movie.edu.      IN A     192.249.249.1
wh253.movie.edu.      IN A     192.253.253.1
```

А вот содержимое файла *db.192.249.249*:

```
$TTL 3h
249.249.192.in-addr.arpa. IN SOA terminator.movie.edu. al.robocop.movie.edu. (
                            1       ; Порядковый номер
```

```
                                3h      ; Обновление через 3 часа
                                1h      ; Повторение попытки через 1 час
                                1w      ; Устаревание через 1 неделю
                                1h )    ; Отрицательное TTL в 1 час

;
; Серверы имен
;
249.249.192.in-addr.arpa.  IN NS  terminator.movie.edu.
249.249.192.in-addr.arpa.  IN NS  wormhole.movie.edu.

;
; Адреса, указывающие на канонические имена
;
1.249.249.192.in-addr.arpa.  IN PTR wormhole.movie.edu.
2.249.249.192.in-addr.arpa.  IN PTR robocop.movie.edu.
3.249.249.192.in-addr.arpa.  IN PTR terminator.movie.edu.
4.249.249.192.in-addr.arpa.  IN PTR diehard.movie.edu.
```

А вот содержимое файла *db.192.253.253*:

```
$TTL 3h
253.253.192.in-addr.arpa. IN SOA terminator.movie.edu. al.robocop.movie.edu. (
                                1        ; Порядковый номер
                                3h       ; Обновление через 3 часа
                                1h       ; Повторение попытки через 1 час
                                1w       ; Устаревание через 1 неделю
                                1h )     ; Отрицательное TTL в 1 час
;
; Серверы имен
;
253.253.192.in-addr.arpa.  IN NS  terminator.movie.edu.
253.253.192.in-addr.arpa.  IN NS  wormhole.movie.edu.

;
; Адреса, указывающие на канонические имена
;
1.253.253.192.in-addr.arpa.  IN PTR wormhole.movie.edu.
2.253.253.192.in-addr.arpa.  IN PTR misery.movie.edu.
3.253.253.192.in-addr.arpa.  IN PTR shining.movie.edu.
4.253.253.192.in-addr.arpa.  IN PTR carrie.movie.edu.
```

Loopback-адрес

Серверу имен требуется еще один дополнительный файл *db.ADDR* для *loopback*-сети и специального адреса, который используется хостом для направления пакетов самому себе. Эта сеть (почти) всегда имеет номер 127.0.0/24, а адрес узла (почти) всегда 127.0.0.1. Следовательно, файл имеет имя *db.127.0.0*. Что неудивительно, поскольку он похож на другие файлы *db.ADDR*.

Вот содержимое файла *db.127.0.0*:

```
$TTL 3h
0.0.127.in-addr.arpa. IN SOA terminator.movie.edu. al.robocop.movie.edu. (
                          1        ; Порядковый номер
                          3h       ; Обновление через 3 часа
                          1h       ; Повторение попытки через 1 час
                          1w       ; Устаревание через 1 неделю
                          1h )     ; Отрицательное TTL в 1 час

0.0.127.in-addr.arpa.  IN NS  terminator.movie.edu.
0.0.127.in-addr.arpa.  IN NS  wormhole.movie.edu.

1.0.0.127.in-addr.arpa.  IN PTR localhost.
```

Зачем DNS-серверам нужен этот глупый маленький файл? Задумаемся на секунду. Никто не отвечает за сеть 127.0.0/24, но она используется многими системами в качестве *loopback*. Поскольку никто не отвечает за эту сеть, каждый отвечает за нее самостоятельно. Можно обойтись без этого файла, и DNS-сервер будет работать. Однако поиск по адресу 127.0.0.1 не даст положительных результатов, поскольку корневой DNS-сервер не был настроен таким образом, чтобы отображать адрес 127.0.0.1 в конкретное имя. Чтобы избежать сюрпризов, это отображение должен обеспечить администратор DNS-сервера.

Указатели корневых серверов

Помимо локальной информации, DNS-серверу также необходимо обладать информацией о DNS-серверах корневого домена. Эту информацию следует получить с интернет-узла *ftp.rs.internic.net* (198.41.0.6). Воспользуйтесь анонимным FTP-доступом, чтобы скопировать файл *named.root* из подкаталога *domain* на этом сервере. (*named.root* – это тот файл, которому мы дали имя *db.cache*. После копирования его просто следует переименовать в *db.cache*.)

```
;       This file holds the information on root name servers needed to
;       initialize cache of Internet domain name servers
;       (e.g. reference this file in the "cache  .  <file>"
;       configuration file of BIND domain name servers).
;
;       This file is made available by InterNIC registration services
;       under anonymous FTP as
;           file                /domain/named.root
;           on server           FTP.RS.INTERNIC.NET
;       -OR- under Gopher at    RS.INTERNIC.NET
;           under menu          InterNIC Registration Services (NSI)
;              submenu          InterNIC Registration Archives
;           file                named.root
;
;       last update:    Aug 22, 1997
;       related version of root zone:   1997082200
;
;
```

```
; formerly NS.INTERNIC.NET
;
.                       3600000  IN  NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.     3600000      A     198.41.0.4
;
; formerly NS1.ISI.EDU
;
.                       3600000      NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.     3600000      A     128.9.0.107
;
; formerly C.PSI.NET
;
.                       3600000      NS    C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.     3600000      A     192.33.4.12
;
; formerly TERP.UMD.EDU
;
.                       3600000      NS    D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET.     3600000      A     128.8.10.90
;
; formerly NS.NASA.GOV
;
.                       3600000      NS    E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET.     3600000      A     192.203.230.10
;
; formerly NS.ISC.ORG
;
.                       3600000      NS    F.ROOT-SERVERS.NET.
F.ROOT-SERVERS.NET.     3600000      A     192.5.5.241
;
; formerly NS.NIC.DDN.MIL
;
.                       3600000      NS    G.ROOT-SERVERS.NET.
G.ROOT-SERVERS.NET.     3600000      A     192.112.36.4
;
; formerly AOS.ARL.ARMY.MIL
;
.                       3600000      NS    H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET.     3600000      A     128.63.2.53
;
; formerly NIC.NORDU.NET
;
.                       3600000      NS    I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET.     3600000      A     192.36.148.17
;
; temporarily housed at NSI (InterNIC)
;
.                       3600000      NS    J.ROOT-SERVERS.NET.
J.ROOT-SERVERS.NET.     3600000      A     198.41.0.10
;
; housed in LINX, operated by RIPE NCC
```

```
;
.                       3600000      NS    K.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET.     3600000      A     193.0.14.129
;
; temporarily housed at ISI (IANA)
;
.                       3600000      NS    L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET.     3600000      A     198.32.64.12
;
; housed in Japan, operated by WIDE
;
.                       3600000      NS    M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.     3600000      A     202.12.27.33
; End of File
```

Доменное имя «.» обозначает корневую зону. Поскольку серверы корневой зоны время от времени меняются, не стоит считать этот список соответствующим действительности. Воспользуйтесь последней версией файла *named.root*.

Каким образом файл идет в ногу со временем? Это задача, которую выполняет сетевой администратор. Некоторые из более старых версий BIND умели периодически обновлять этот файл, но эта возможность в итоге была изъята, поскольку она, очевидно, не работала так, как задумали авторы. Время от времени измененный файл *db.cache* помещается в список рассылки *bind-users* или *namedroppers*. Если вы являетесь участником одного из этих списков, то, скорее всего, услышите об изменениях.

Можно ли помещать в этот файл данные, не относящиеся к корневым DNS-серверам? Можно, но эти данные не будут использоваться. Когда-то DNS-серверы помещали данные из этого файла в кэш. Использование файла изменилось (неуловимым образом), а имя «кэш-файл» осталось. Сервер имен хранит данные этого файла в специальной области памяти, которая известна как область *корневых указателей* (*root hints*). В отличие от кэшированных данных, указатели, после истечения интервала TTL, продолжают использоваться. Корневые указатели используются DNS-сервером, чтобы запрашивать у корневых DNS-серверов текущий перечень корневых DNS-серверов, который уже кэшируется. Когда истекает интервал TTL для этого перечня, DNS-сервер повторяет процедуру и получает новый.

Зачем DNS-серверу посылать запрос DNS-серверу из списка корневых указателей – который, вероятно, сам является корневым DNS-сервером – на предмет получения списка корневых DNS-серверов, если такой список уже есть? Причина проста, DNS-сервер, которому посылается запрос, практически наверняка знает *текущий* список корневых DNS-серверов, тогда как файл может уже не соответствовать действительности.

Для чего нужны цифры 3600000? Это явным образом указанное время жизни записей файла в секундах. В более старых версиях этого файла использовалось число 99999999. Поскольку содержимое файла изначально кэшировалось, DNS-серверу необходимо было знать, как долго можно хранить записи. 99999999 секунд – это просто очень большой интервал времени, который позволял хранить данные в кэше на протяжении всего времени работы сервера. Поскольку DNS-сервер теперь хранит эти данные в специальной области данных и не удаляет их при истечении заданного интервала времени, TTL стали необязательными. Но иметь цифры 3600000 не помешает, и в случае передачи ответственности за сервер следующему администратору это может стать основой BIND-фольклора.

Создание файла настройки BIND

Наконец, создав файлы данных для зоны, мы должны объяснить DNS-серверу, какие именно файлы следует использовать. В BIND для этой цели применяется механизм файла настройки. До сих пор мы обсуждали файлы, формат которых определяется спецификациями DNS. Файл настройки является уникальным для BIND, и его формат не определен RFC-документами по DNS.

Синтаксис файла настройки существенно изменился при переходе от версий 4 к версиям 8. К счастью, он не изменился при переходе от версии 8 к версии 9. Мы начнем с синтаксиса для BIND 4, а затем опишем эквивалентные конструкции для BIND 8 и 9. Сверьтесь со страницами руководства по *named*[1] и уточните, какую версию синтаксиса следует использовать. Если у вас уже есть файл настройки для BIND 4, его можно преобразовать в формат BIND 8 или 9, выполнив программу *named-bootconf*, которая входит в комплект поставки исходных текстов BIND. В BIND 8 эта программа находится в каталоге *src/bin/named-bootconf*. В BIND 9 – в каталоге *contrib/named-bootconf*.

В BIND версии 4, комментарии в файле настройки выглядят так же, как и в файлах данных зоны, они начинаются с символа точки с запятой и заканчиваются концом строки:

```
; это комментарий
```

В BIND 8 и 9 можно пользоваться одним из трех стилей комментариев: С-стилем, С++-стилем или стилем командного интерпретатора:

[1] *named* произносится как «name-dee» («нэйм-ди») и означает «name server daemon» (демон DNS-сервера). BIND произносится аналогично «kind" («кайнд»). Некоторые творческие личности заметили похожесть названий и неправильно прозносят их – «bin-dee» («бин-ди») и «named» (как «tamed», «тэймд»).

```
/* Комментарий в стиле языка C */
// Комментарий в стиле языка C++
# Комментарий  в стиле командного интерпретатора
```

Не пользуйтесь комментариями в стиле BIND 4 в файлах настройки для BIND 8 и 9 – работать это не будет, потому что в последних версиях BIND точка с запятой является признаком конца оператора настройки.

Обычно файлы настройки содержат строку, определяющую каталог, в котором расположены файлы данных зоны. Сервер имен изменяет рабочий каталог на указанный перед чтением файлов данных зоны, что позволяет указывать имена файлов относительного текущего каталога. Вот так выглядит указание каталога для BIND 4:

```
directory /var/named
```

А так – для BIND 8 и 9:

```
options {
        directory "/var/named";
        // здесь указываются дополнительные параметры
};
```

В файле настройки может присутствовать только один оператор *options*, так что любые дополнительные параметры, упомянутые далее по тексту, должны указываться в этом операторе наряду с параметром *directory*.

Файл настройки для первичного мастер-сервера содержит по одной строке для каждого файла данных зоны, который следует учитывать. В BIND 4 такая строка состоит из трех полей – слова *primary* (начинающегося в первой позиции строки), доменного имени зоны и имени файла:

```
primary  movie.edu                db.movie.edu
primary  249.249.192.in-addr.arpa db.192.249.249
primary  253.253.192.in-addr.arpa db.192.253.253
primary  0.0.127.in-addr.arpa     db.127.0.0
```

В BIND 8 и 9 эта строка начинается с ключевого слова *zone*, продолжается доменным именем и именем класса (*in* – класс Интернета). Тип *master* эквивалентен типу *primary* для BIND 4. Последнее поле содержит имя файла:

```
zone "movie.edu" in {
       type master;
       file "db.movie";
};
```

Мы упоминали, что если не указывать класс данных в RR-записи, DNS-сервер определит класс, исходя из данных в файле настройки.

Указание *in* в операторе *zone* устанавливает класс данных Интернета. Для оператора *zone* в BIND 8 и 9 класс *in* является классом по умолчанию, так что его можно не указывать для зон класса Интернета. Поскольку в синтаксисе BIND 4 не предусмотрено указание класса зоны, по умолчанию также принимается значение *in*.

Вот строка файла настройки BIND 4, предписывающая чтение файла корневых указателей:

```
cache . db.cache
```

а вот эквивалентная строка для файла настройки BIND 8 или 9[1]:

```
zone "." in {
        type hint;
        file "db.cache";
};
```

Как ранее уже говорилось, этот файл не содержит данные кэша, а только *указатели* (*hints*) корневых DNS-серверов.

По умолчанию BIND 4 читает файл настройки с именем */etc/named.boot*, но этот параметр можно изменить с помощью ключа командной строки. BIND 8 и 9 ожидают найти файл настройки */etc/named.conf* вместо */etc/named.boot*. Файлы данных зоны в нашем примере расположены в каталоге */var/named*. В каком именно каталоге они будут расположены в той или иной системе – особого значения не имеет. Следует избегать лишь помещения этого каталога в корневую файловую систему, если ощущается нехватка места в этой системе, а также следить за тем, чтобы файловая система, содержащая этот каталог, была смонтирована до запуска DNS-сервера. Вот полностью файл */etc/named.boot* для BIND 4:

```
; Файл настройки BIND

directory /var/named

primary  movie.edu                db.movie.edu
primary  249.249.192.in-addr.arpa db.192.249.249
primary  253.253.192.in-addr.arpa db.192.253.253
primary  0.0.127.in-addr.arpa     db.127.0.0
cache    .                        db.cache
```

А вот полный файл */etc/named.conf* для BIND 8 и 9:

```
// Файл настройки BIND

options {
```

[1] В действительности, в BIND 9 существует встроенная зона *указателей*, поэтому нет необходимости включать оператор *zone* для зоны указателей в файл *named.conf*. Однако ее включение не повредит, поскольку отсутствие этой зоны в файле настройки может стать причиной нервной дрожи.

```
        directory "/var/named";
        // Здесь указываются дополнительные параметры
};
zone "movie.edu" in {
        type master;
        file "db.movie.edu";
};

zone "249.249.192.in-addr.arpa" in {
        type master;
        file "db.192.249.249";
};

zone "253.253.192.in-addr.arpa" in {
        type master;
        file "db.192.253.253";
};

zone "0.0.127.in-addr.arpa" in {
        type master;
        file "db.127.0.0";
};

zone "." in {
        type hint;
        file "db.cache";
};
```

Сокращения

Итак, мы создали все файлы, необходимые для работы первичного мастер-сервера DNS. Теперь взглянем еще раз на файлы данных зоны, некоторые углы мы не стали срезать, и сейчас самое время это сделать. Но, не зная, как выглядит полная форма записи, можно совершенно запутаться в краткой форме. Поскольку читатели уже знают, как выглядит полная форма файла настройки BIND, переходим к сокращениям.

Добавление доменных имен

Второе поле директивы *primary* (BIND 4) или оператора *zone* (BIND 8 и 9) определяет доменное имя. Это доменное имя является ключом к наиболее полезному типу сокращений. Оно является *суффиксом по умолчанию* (*origin*) для всей информации в файле данных зоны. Суффикс по умолчанию добавляется ко всем именам в файле данных зоны, которые не заканчиваются точкой, и поскольку каждый файл описывает отдельную зону, суффиксы по умолчанию в разных файлах различны.

Поскольку имя суффикса по умолчанию добавляется в конец остальных имен, для адреса *robocop.movie.edu* в файле *db.movie.edu* можно указать вместо:

```
robocop.movie.edu.    IN A     192.249.249.2
```

вот такую строку:

```
robocop    IN A     192.249.249.2
```

В файле *db.192.24.249* присутствует строка:

```
2.249.249.192.in-addr.arpa.  IN PTR robocop.movie.edu.
```

Поскольку *249.249.192.in-addr.arpa* является суффиксом по умолчанию, можно заменить ее на следующую:

```
2  IN PTR robocop.movie.edu.
```

Если помните, мы предупреждали, что не следует забывать ставить точку в конце полных доменных имен. Предположим, вы забыли поставить последнюю точку. И запись:

```
robocop.movie.edu    IN A     192.249.249.2
```

превратится в запись для *robocop.movie.edu.movie.edu*, то есть мы получим совершенно нежелательный результат.

Запись через @

Если доменное имя *совпадает* с суффиксом по умолчанию, его можно указывать в виде «@». Чаще всего такая запись встречается в SOA-записях в файлах данных зоны. Выглядит это следующим образом:

```
@ IN SOA terminator.movie.edu. al.robocop.movie.edu. (
                    1        ; Порядковый номер
                    3h       ; Обновление через 3 часа
                    1h       ; Повторение попытки через 1 час
                    1w       ; Устаревание через 1 неделю
                    1h )     ; Отрицательное TTL в 1 час
```

Повтор последнего имени

Если имя RR-записи (начинающееся в первой позиции строки) состоит из пробелов или символа табуляции, то автоматически подставляется имя из предыдущей записи. Это можно использовать при необходимости создать несколько записей для одного имени. Приведем пример использования одного имени для создания двух записей:

```
wormhole   IN A     192.249.249.1
           IN A     192.253.253.1
```

Во второй адресной записи неявно указывается имя *wormhole*. Этим сокращением можно пользоваться даже в случаях, когда RR-записи имеют разные типы.

Сокращенные файлы данных зоны

Теперь, когда читатели ознакомились с сокращениями, мы повторим файлы данных для зоны, применив эти сокращения на деле.

Содержимое файла *db.movie.edu*:

```
$TTL 3h
;
; Суффикс по умолчанию, добавляемый ко всем именам,
; не заканчивающимся точкой: movie.edu
;

@ IN SOA terminator.movie.edu. al.robocop.movie.edu. (
                    1        ; Порядковый номер
                    3h       ; Обновление через 3 часа
                    1h       ; Повторение попытки через 1 час
                    1w       ; Устаревание через 1 неделю
                    1h )     ; Отрицательное TTL в 1 час

;
; Серверы имен (неявно указано имя '@')
;
            IN NS  terminator.movie.edu.
            IN NS  wormhole.movie.edu.

;
; Адреса для канонических имен
;
localhost  IN A     127.0.0.1
robocop    IN A     192.249.249.2
terminator IN A     192.249.249.3
diehard    IN A     192.249.249.4
misery     IN A     192.253.253.2
shining    IN A     192.253.253.3
carrie     IN A     192.253.253.4

wormhole   IN A     192.249.249.1
           IN A     192.253.253.1

;
; Псевдонимы
;
bigt       IN CNAME terminator
dh         IN CNAME diehard
wh         IN CNAME wormhole

;
; Специальные имена интерфейсов
;
wh249      IN A     192.249.249.1
wh253      IN A     192.253.253.1
```

А вот содержимое файла *db.192.249.249*:

```
$TTL 3h
```

```
;
; Суффикс по умолчанию, добавляемый ко всем именам,
; не заканчивающимся точкой: 249.249.192.in-addr.arpa
;

@ IN SOA terminator.movie.edu. al.robocop.movie.edu. (
                    1        ; Порядковый номер
                    3h       ; Обновление через 3 часа
                    1h       ; Повторение попытки через 1 час
                    1w       ; Устаревание через 1 неделю
                    1h )     ; Отрицательное TTL в 1 час

;
; Серверы имен (неявно указано имя '@')
;
    IN NS  terminator.movie.edu.
    IN NS  wormhole.movie.edu.

;
; Адреса, указывающие на канонические имена
;
1  IN PTR wormhole.movie.edu.
2  IN PTR robocop.movie.edu.
3  IN PTR terminator.movie.edu.
4  IN PTR diehard.movie.edu.
```

И содержимое файла *db.192.253.253*:

```
$TTL 3h
;
; Суффикс по умолчанию, добавляемый ко всем именам,
; не заканчивающимся точкой: 253.253.192.in-addr.arpa
;

@ IN SOA terminator.movie.edu. al.robocop.movie.edu. (
                    1        ; Порядковый номер
                    3h       ; Обновление через 3 часа
                    1h       ; Повторение попытки чере 1 час
                    1w       ; Устаревание через 1 неделю
                    1h )     ; Отрицательное TTL в 1 час

;
; Серверы имен (неявно указано имя '@')
;
    IN NS  terminator.movie.edu.
    IN NS  wormhole.movie.edu.

;
; Адреса, указывающие на канонические имена
;
1  IN PTR wormhole.movie.edu.
2  IN PTR misery.movie.edu.
3  IN PTR shining.movie.edu.
4  IN PTR carrie.movie.edu.
```

Содержимое файла *db.127.0.0*:

```
$TTL 3h
@ IN SOA terminator.movie.edu. al.robocop.movie.edu. (
                    1        ; Порядковый номер
                    3h       ; Обновление через 3 часа
                    1h       ; Повторение попытки через 1 час
                    1w       ; Устаревание через 1 неделю
                    1h )     ; Отрицательное TTL в 1 час

    IN NS  terminator.movie.edu.
    IN NS  wormhole.movie.edu.

1  IN PTR localhost.
```

Читатели могли заметить, что в новой версии файла *db.movie.edu* можно было бы удалить *movie.edu* из имен узлов в записях SOA и NS следующим образом:

```
@ IN SOA terminator al.robocop (
                    1        ; Порядковый номер
                    3h       ; Обновление через 3 часа
                    1h       ; Повторение попытки через 1 час
                    1w       ; Устаревание через 1 неделю
                    1h )     ; Отрицательное TTL в 1 час

IN NS  terminator
IN NS  wormhole
```

Но так нельзя делать в прочих файлах данных зоны, поскольку они имеют отличные суффикса по умолчанию. В файле *db.movie.edu* мы оставили полные доменные имена, чтобы записи SOA и NS были абсолютно одинаковыми во *всех* файлах данных зоны

Проверка имени узла (BIND 4.9.4 и более поздние версии)

Если используемый вами DNS-сервер имеет версию более раннюю, чем 4.9.4, или же используются версии от 9 до 9.1.0[1], переходите к следующему разделу.

Если DNS-сервер имеет версию 4.9.4 или более позднюю, следует обратить пристальное внимание на имена узлов. Начиная с версии 4.9.4, BIND проверяет имена узлов на соответствие документу RFC 952. Несоответствие имени узла этому документу считается синтаксической ошибкой.

[1] Проверка имен не реализована в версиях BIND с 9 по 9.1.0. Возможно, она будет реализована в последующих версиях BIND 9, поэтому имеет смысл все же прочесть этот раздел.

Прежде чем начать паниковать, следует осознать, что проверка применяется только к именам, которые считаются именами узлов. Вспомните, что RR-запись содержит поле имени и поле данных. Например:

```
<имя>       <класс>  <тип>   <данные>
terminator  IN       A       192.249.249.3
```

Имена узлов встречаются в поле имени адресных (A) записей и MX-записей (которые рассмотрены в главе 5 «DNS и электронная почта»). Имена узлов также встречаются в поле данных записей типа SOA и NS. CNAME-записи не подчиняются правилам именования узлов, поскольку могут указывать на имена, не являющиеся именами узлов.

Рассмотрим правила именования узлов. Имена узлов могут содержать буквы и цифры в каждой из меток. Следующие имена узлов являются допустимыми:

```
ID4            IN A 192.249.249.10
postmanring2x  IN A 192.249.249.11
```

Дефис внутри метки разрешен:

```
fx-gateway     IN A 192.249.249.12
```

Недопустимо использование подчеркивания в именах узлов.

Имена, не являющиеся именами узлов, могут состоять из любых отображаемых ASCII-символов.

Если в поле данных RR-записи необходимо указать адрес электронной почты (как в SOA-записях), первая метка, которая не является именем узла, может содержать любые отображаемые символы, но все остальные метки должны соответствовать описанному синтаксису имен узлов. Так, почтовый адрес имеет следующий вид:

```
<ASCII-символы>.<символы-допустимые-в-имени-узла>
```

Почтовый адрес *key_grip@movie.edu* можно без проблем использовать в SOA-записи, несмотря на подчеркивание. Не забывайте, что в почтовых адресах символ «@» следует заменять символом «.», следующим образом:

```
movie.edu. IN SOA terminator.movie.edu. key_grip.movie.edu. (
                  1        ; Порядковый номер
                  3h       ; Обновление через 3 часа
                  1h       ; Повторение попытки через 1 час
                  1w       ; Устаревание через 1 неделю
                  1h )     ; Отрицательное TTL в 1 час
```

Этот вторичный этап проверки может привести к большим проблемам в случае обновления более старой, либеральной версии BIND до новой, консервативной, особенно в тех случаях, когда администраторами было стандартизировано использование подчеркиваний в именах узлов. Если необходимо отложить смену имен (вы ведь не забудете все равно их поменять?), этот момент можно упростить до выдачи простых предупреждающих сообщений вместо ошибок либо просто игнорирования неправильных имен. Следующий оператор в файле настройки BIND 4 превращает ошибки в предупреждающие сообщения:

```
check-names primary warn
```

Эквивалентная строчка для BIND 8 и BIND 9:

```
options {
        check-names master warn;
};
```

Предупреждающие сообщения заносятся в log-файл посредством *syslog*, инструмента, который мы затронем чуть позже. Следующий оператор в файле настройки BIND 4 позволяет полностью проигнорировать ошибки проверки имен:

```
check-names primary ignore
```

Эквивалент для BIND 8 и BIND 9:

```
options {
        check-names master ignore;
};
```

Если неправильные имена принадлежат зоне, для которой ваш сервер является вторичным (и над которой у вас нет контроля), добавьте аналогичный оператор с ключевым словом *secondary* вместо *primary*:

```
check-names secondary ignore
```

В случае BIND 8 или 9, используйте *slave* вместо *secondary*:

```
options {
        check-names slave ignore;
};
```

А для имен, получаемых в качестве ответов на запросы, можно указать:

```
check-names response ignore
```

Для BIND 8:

```
options {
        check-names response ignore;
};
```

Установки по умолчанию для BIND 4.9.4:

```
check-names primary fail
check-names secondary warn
check-names response ignore
```

Установки по умолчанию для BIND 8:

```
options {
         check-names master fail;
         check-names slave warn;
         check-names response ignore;
};
```

В BIND 8 проверку имен можно настраивать отдельно для каждой зоны, и в случае указания значения для конкретной зоны это значение имеет более высокий приоритет, чем определенное оператором *options*:

```
zone "movie.edu" in {
         type master;
         file "db.movie.edu";
        check-names fail;
};
```

Строка *options* содержит три поля (*check-names master fail*), тогда как строка проверки для зоны – только два поля (*check-names fail*). Это происходит потому, что строка оператора *zone* уже определяет контекст (зону, указанную этим оператором).

Инструменты

Не правда ли, было бы очень удобно иметь инструмент для преобразования таблицы узлов в формат мастер-файла? Существует такая зверушка, написанная на языке Perl: *h2n*. *h2n* можно использовать для создания начальных файлов данных и последующего самостоятельного их сопровождения. Как вариант, можно пользоваться программой *h2n* на постоянной основе. Как мы видели, формат таблицы узлов гораздо проще для понимания и корректного редактирования, чем формат мастер-файла. Поэтому существует возможность сопровождать файл */etc/hosts* и повторно выполнять *h2n* при необходимости обновить зону после внесения в файл изменений.

Если вы планируете использовать *h2n*, то можно и начать с этого инструмента, поскольку для создания новых файлов данных зоны он использует файл */etc/hosts*, а не созданные вручную зональные данные. Мы могли бы сэкономить время и силы, сгенерировав пример зональных данных в этой главе посредством следующей команды:

```
% h2n -d movie.edu -s terminator -s robocop \
```

```
-n 192.249.249 -n 192.253.253 \
-u al.robocop.movie.edu
```

(Для BIND 8 и 9 следует добавить *-v 8* к списку ключей команды.)

Ключи *-d* и *-n* позволяют указать доменное имя для зоны прямого отображения и номер сети. Обратите внимание, что имена файлов данных зоны создаются на основе параметров этих ключей. Ключи *-s* позволяют перечислить авторитативные DNS-серверы зон, которые будут использованы при создании NS-записей. Ключ *-u* (user, пользователь) позволяет указать адрес электронной почты для SOA-записи. Программа *h2n* более подробно рассмотрена в главе 7, после изучения того, как DNS влияет на электронную почту.

Запуск первичного мастер-сервера DNS

Итак, создав файлы данных для зоны, мы готовы к запуску пары DNS-серверов. Речь идет об основном контрольном сервере имен и вторичном сервере имен. Однако прежде чем запускать DNS-сервер, следует убедиться, что запущен демон *syslog*. Если DNS-сервер при чтении файла настройки или зональных данных находит ошибку, информация об этой ошибке заносится в log-файл с помощью демона *syslog*. Если ошибка критическая, DNS-сервер прекращает работу.

Запуск DNS-сервера

Мы предполагаем, что к данному моменту на машине уже установлен DNS-сервер BIND и программа *nslookup*. Сверьтесь с руководством по *named* в поисках каталога, который содержит исполняемый файл сервера, и убедитесь, что этот исполняемый файл присутствует в системе. В системах BSD DNS-сервер начинал свое существование в каталоге */etc*, но вполне мог переместиться в */usr/sbin*. Также *named* можно поискать в */usr/etc/in.named* и */usr/sbin/in.named*. Последующие описания подразумевают, что файл расположен в каталоге */usr/sbin*.

Чтобы запустить сервер, следует получить полномочия суперпользователя (root). Сервер имен принимает запросы через привилегированный порт, а для этого требуются права пользователя root. На первый раз запустите DNS-сервер из командной строки, чтобы убедиться в его корректной работе. Позже мы объясним, как автоматически запускать сервер при загрузке системы.

Следующая команда запускает DNS-сервер. Мы выполнили ее на узле *terminator.movie.edu*:

```
# /usr/sbin/named
```

Такая команда подразумевает, что файлом настройки является */etc/named.boot* (BIND 4) или */etc/named.conf* (BIND 8 или 9). Файл на-

стройки может находиться в другом месте, но в этом случае следует указать DNS-серверу – где именно, используя ключ *-c*:

```
# /usr/sbin/named -c conf-file
```

Ошибки в log-файле syslog

Первое, что следует сделать после запуска DNS-сервера, – заглянуть в log-файл демона *syslog* в поисках сообщений об ошибках. Те, кто не знаком с демоном *syslog*, могут взглянуть на страницы руководства по файлу *syslog.conf* и ознакомиться с описанием файла настройки *syslog* либо изучить документацию по программе *syslogd* (которая и содержит описание демона *syslog*). Сервер имен заносит сообщения в log-файл как *daemon* (демон) под именем *named*. Узнать, в какой файл записываются сообщения демона *syslog*, можно, найдя слово *daemon* в файле */etc/syslog.conf*:

```
% grep daemon /etc/syslog.conf
*.err;kern.debug;daemon,auth.notice /var/adm/messages
```

На этом узле *syslog*-сообщения от DNS-сервера заносятся в log-файл, хранимый в файле */var/adm/messages*, и при этом *syslog* пропускает только сообщения, которые имеют приоритет LOG_NOTICE или более высокий. Некоторые полезные сообщения имеют приоритет LOG_INFO – и вы можете захотеть их прочитать. Следует ли изменять этот уровень, читатели смогут решить, прочтя главу 7, в которой мы рассмотрим сообщения *syslog* более подробно.

При запуске DNS-сервера в log-файл заносится стартовое сообщение:

```
% grep named /var/adm/messages
Jan 10 20:48:32 terminator named[3221]: starting.
```

Стартовое сообщение не является сообщением об ошибке, но за ним могут следовать другие сообщения, обладающие этим свойством. (Если сервер сообщает *restarted* (*перезапуск*) вместо *starting*, это тоже вполне нормально. Это сообщение изменилось в BIND версии 4.9.3.) Наиболее часто встречаются синтаксические ошибки в файлах данных зоны и файле настройки. К примеру, если мы забудем указать тип записи в адресной записи:

```
robocop  IN  192.249.249.2
```

сервер отреагирует следующими *syslog*-сообщениями:

```
Jan 10 20:48:32 terminator named[3221]: Line 24: Unknown type:
                192.249.249.2
Jan 10 20:48:32 terminator named[3221]: db.movie.edu Line 24:
                Database error near (192.249.249.2)
Jan 10 20:48:32 terminator named[3221]: master zone "movie.edu" (IN) rejected
due to errors (serial 1)
```

В случае, если сделать орфографическую ошибку в слове «zone» в файле */etc/named.conf*:

```
zne "movie.edu" in {
```

будет получено следующее сообщение об ошибке:

```
Mar 22 20:14:21 terminator named[1477]: /etc/named.conf:10:
                 syntax error near `zne'
```

Если BIND версии 4.9.4 или более поздней находит имя, которое не соответствует стандарту, установленному документом RFC 952, в log-файл демона *syslog* попадут такие сообщения:

```
Jul 24 20:56:26 terminator named[1496]: owner name "ID_4.movie.edu IN"
                                        (primary) is invalid - rejecting
Jul 24 20:56:26 terminator named[1496]: db.movie.edu:33: owner name error
Jul 24 20:56:26 terminator named[1496]: db.movie.edu:33: Database error near (A)
Jul 24 20:56:26 terminator named[1496]: master zone "movie.edu" (IN) rejected
due to errors (serial 1)
```

Если речь идет о синтаксической ошибке, следует проверить строки, которые упоминаются в сообщениях *syslog*, и попытаться понять, в чем проблема. Читатели уже представляют себе, как должны выглядеть файлы данных зоны; этого представления должно быть достаточно, чтобы разобраться с самыми простыми ошибками. В сложных случаях им придется обращаться к приложению А «Формат сообщений DNS и RR-записей», а значит – кровавым подробностям синтаксиса всех RR-записей. Если синтаксическая ошибка поддается исправлению, следует исправить ее, после чего перезагрузить сервер командой *ndc* :

```
# ndc reload
```

Эта команда предписывает серверу прочитать файлы данных повторно.[1] Использование *ndc* для управления DNS-сервером более подробно описано в главе 7.

Тестирование системы с помощью nslookup

В случае корректного создания локальных зон и действующего подключения к сети Интернет, не должно возникать никаких сложностей при выполнении запросов по локальному или удаленному доменному имени. Сейчас мы произведем несколько операций поиска с помощью *nslookup*. В этой книге программе *nslookup* посвящена целая глава (глава 12, *nslookup и dig*), но мы достаточно подробно рассмотрим ее здесь, чтобы произвести базовое тестирование DNS-сервера.

---

[1] В случае сервера имен BIND 9 потребуется использовать *rndc*, но мы еще не рассказывали о настройке этой программы. Информация об этом содержится в главе 7. А *ndc* работает практически без настройки.

Установка локального доменного имени

Прежде чем начать работу с *nslookup*, следует установить локальное доменное имя узла. После этого можно будет производить поиск по имени *carrie* без необходимости полностью произносить *carrie.movie.edu* – доменное имя *movie.edu* будет добавляться системой автоматически.

Существуют два способа установки локального доменного имени: с помощью программы *hostname(1)* либо указанием в файле */etc/resolv.conf*. Некоторые утверждают, что на практике в большинстве случаев применяется указание в файле */etc/resolv.conf*. Используйте любой из способов. В этой книге мы предполагаем, что локальное доменное имя получается посредством *hostname(1)*.

Создайте файл */etc/resolv.conf* и добавьте в него следующую строку, начав ее с первой позиции строки (вместо *movie.edu* используйте локальное доменное имя):

```
domain movie.edu
```

Либо с помощью *hostname(1)* задайте доменное имя. Для узла *terminator* мы использовали *hostname(1)* и доменное имя *terminator.movie.edu*. Точку к имени добавлять не следует.

Поиск для локального доменного имени

nslookup можно использовать для поиска RR-записей любого типа, через любой DNS-сервер. По умолчанию происходит поиск адресных (A) записей, а запросы посылаются первому из DNS-серверов, определенных в файле *resolv.conf*. (Если имя DNS-сервера не указано в *resolv.conf*, DNS-клиент посылает запросы локальному DNS-серверу.) Чтобы найти адрес узла с помощью *nslookup*, следует выполнить *nslookup* с единственным аргументом – доменным именем узла. Поиск для локального доменного имени должен вернуть результаты практически мгновенно.

Мы выполнили *nslookup* для узла *carrie*:

```
% nslookup carrie
Server: terminator.movie.edu
Address: 192.249.249.3

Name:    carrie.movie.edu
Address: 192.253.253.4
```

Если поиск для локального доменного имени работает, локальный DNS-сервер был настроен правильно для зоны прямого отображения. В случае, если поиск возвращает ошибку, пользователь получает сообщение вроде этого:

```
*** terminator.movie.edu can't find carrie: Non-existent domain
```

Такое сообщение означает, что узел *carrie* не входит в зону (проверьте файл данных зоны), либо не было установлено локальное доменное имя (*hostname(1)*), либо присутствовали иные ошибки в работе DNS-сервера (хотя их следовало отловить при проверке сообщений *syslog*).

Поиск для локального адреса

Если *nslookup* в качестве аргумента получает адрес, то выполняется PTR-запрос вместо адресного. Мы выполнили *nslookup* для адреса узла *carrie*:

```
% nslookup 192.253.253.4
Server: terminator.movie.edu
Address: 192.249.249.3

Name:    carrie.movie.edu
Address: 192.253.253.4
```

Если поиск по адресу работает, локальный DNS-сервер был настроен правильно для зоны *in-addr.arpa* (зоны обратного отображения). Если поиск не сработает, будет отображено сообщение об ошибке, похожее на то, что было бы получено при поиске по доменному имени.

Поиск для внешнего доменного имени

Следующий шаг – попытаться использовать локальный DNS-сервер для поиска по внешнему доменному имени, скажем *ftp.uu.net*, или имени любой другой системы, которая нам известна и расположена в сети Интернет. Эта команда возвращает результат не столь быстро, как предыдущие. Если *nslookup* не получает ответ от локального DNS-сервера, пройдет чуть больше минуты, прежде чем работа завершится с соответствующим сообщением.

```
% nslookup ftp.uu.net.
Server: terminator.movie.edu
Address: 192.249.249.3

Name:      ftp.uu.net
Addresses: 192.48.96.9
```

Если получен положительный результат, можно сделать вывод, что локальному DNS-серверу известны координаты корневых DNS-серверов и способы связи с ними, которые позволяют получать информацию по доменным именам, расположенным во внешних зонах. В случае отрицательного ответа причиной может быть отсутствие файла корневых указателей (соответствующее сообщение *syslog* будет занесено в log-файл), либо присутствуют неполадки в сети, которые не позволяют установить связь с DNS-серверами внешней зоны. Имеет смысл повторить попытку для другого доменного имени.

Если уже первые попытки поиска увенчались успехом, можете принимать поздравления! Первичный мастер-сервер DNS запущен и функ-

ционирует. С этого момента можно начинать настройку вторичного DNS-сервера.

Еще один тест

Но раз уж мы начали тестировать, выполним и еще одну проверку. Попытаемся сделать так, чтобы удаленный сервер нашел информацию по доменному имени в одной из наших зон. Это сработает только в том случае, если DNS-серверы родительской зоны уже делегировали наши зоны только что установленному DNS-серверу. Если предки потребовали наличия двух работающих DNS-серверов для делегирования, пропустите этот раздел и переходите к следующему.

Для поиска по локальному доменному имени через удаленный DNS-сервер с помощью *nslookup*, следует указать доменное имя в качестве первого аргумента команды, а доменное имя удаленного DNS-сервера в качестве второго. Опять же, если присутствуют какие-то проблемы, может пройти чуть больше минуты, прежде чем *nslookup* выдаст сообщение об ошибке. В следующем примере DNS-сервер *gatekeeper.dec.com* производит поиск адреса для *carrie.movie.edu*:

```
% nslookup carrie gatekeeper.dec.com.
Server: gatekeeper.dec.com.
Address: 204.123.2.2

Name:    carrie.movie.edu
Address: 192.253.253.4
```

Если первые два поиска работали, а поиск локального имени через удаленный DNS-сервер – нет, то, возможно, локальная зона не была зарегистрирована в настройках DNS-серверов родительской зоны. Поначалу это не вызовет особых проблем, поскольку системы в пределах локальных зон могут успешно находить данные для локальных доменных имен и имен, принадлежащих внешним зонам. Вы сможете посылать сообщения электронной почты, использовать FTP для работы с локальными и удаленными системами, хотя некоторые системы запрещают FTP-соединения для случаев, когда невозможно отобразить адреса узлов в их адреса. Через некоторое время отсутствие такой регистрации станет проблемой. Узлы, не принадлежащие локальным зонам, не смогут находить доменные имена в ваших зонах, так что вы сможете посылать почту друзьям, обитающих во внешних зонах, но не будете получать их ответов. Чтобы решить эту проблемы, следует связаться с администратором родительской зоны и попросить их проверить делегирование ваших зон.

Редактирование загрузочных файлов

Убедившись, что DNS-сервер работает правильно и может использоваться в дальнейшем, следует настроить его автоматическую загрузку

и установку доменного имени в загрузочных файлах. Возможно, поставщик операционной системы уже позаботился о том, чтобы DNS-сервер стартовал при загрузке. Возможно, понадобится раскомментировать соответствующие строки в загрузочных файлах, в других случаях в этих файлах может происходить проверка существования файла */etc/named.conf* или */etc/named.boot*. Найти строки для автоматического запуска сервера можно с помощью следующей команды:[1]

```
% grep named /etc/*rc*
```

либо, если речь идет о загрузочных файлах System V:

```
% grep named /etc/rc*/S*
```

В случае если поиск не принес результатов, добавьте примерно такие строки в соответствующий инициализационный файл, с учетом того, что они должны выполняться после того, как сетевые интерфейсы будут инициализированы с помощью *ifconfig*:

```
if test -x /etc/named -a -f /etc/named.conf
then
        echo "Starting named"
        /etc/named
fi
```

Возможно, вы захотите запустить DNS-сервер после того, как хосту будет указан роутер по умолчанию, или будет запущен демон маршрутизации (*routed* или *gated*), в зависимости от того, нужен ли этим службам DNS-сервер, или они могут обойтись файлом */etc/hosts*.

Узнайте, в каком из загрузочных файлов происходит инициализация имени узла. Измените имя хоста (*hostname(1)*) на доменное имя. К примеру, мы изменили:

```
hostname terminator
```

на:

```
hostname terminator.movie.edu
```

Запуск вторичного DNS-сервера

Для надежности потребуется установить еще один DNS-сервер. Возможно (рано или поздно многие так и поступают) установить более двух авторитативных DNS-серверов для локальных зон. Два DNS-сервера – это минимум, поскольку в случае, если есть только один сервер, и он перестает работать, служба доменных имен для зоны перестает функционировать. Второй DNS-сервер делит нагрузку с первым серве-

[1] Для Linux данная команда будет выглядеть: *grep named /etc/rc.d/*/S*. – *Примеч. науч. ред.*

ром или принимает на себя всю нагрузку в случае аварии на первом сервере. *Возможно* просто установить еще один основной DNS-сервер, но мы не рекомендуем этого делать. Вместо этого следует создать вторичный DNS-сервер. Позже, если вы не испугаетесь дополнительных усилий, которые нужно затратить, чтобы несколько первичных серверов могли дружно сосуществовать, то всегда сможете превратить вторичный сервер в первичный.

Откуда DNS-сервер знает, является он первичным для зоны или вторичным? Эта информация содержится в файле *named.conf* – для каждой зоны. NS-записи не содержат такую информацию о серверах имен – они лишь идентифицируют серверы. (Вообще говоря, DNS нет разницы: если происходит разрешение имен, вторичные DNS-серверы ничем не хуже первичных.)

В чем разница между первичным DNS-сервером и вторичным? Коренное различие в том, откуда сервер получает свои данные. Первичный DNS-сервер читает данные из файлов данных зоны. Вторичный сервер загружает данные по сети, получая их от другого DNS-сервера. Этот процесс носит название *передачи зоны*.

Вторичный DNS-сервер может получать свои данные не только от первичного DNS-сервера, но и от другого вторичного сервера.

Большим преимуществом в использовании вторичных DNS-серверов является тот факт, что необходимо сопровождать единственный набор файлов данных для зоны, а именно – набор файлов для первичного сервера. Нет необходимости беспокоиться о синхронизации файлов нескольких DNS-серверов; вторичные DNS-серверы синхронизируются автоматически. Минус в том, что вторичный сервер не синхронизируется каждое мгновение, но проверяет актуальность своих данных через определенные интервалы времени. Интервал опроса – одно из тех чисел в SOA-записи, про которые мы еще ничего не рассказывали. (BIND версий 8 и 9 реализует механизм ускорения распределения зональных данных, который будет описан позже.)

Вторичный DNS-сервер не должен получать по сети все свои данные: «лишние» файлы *db.cache* и *db.127.0.0* точно такие же, как на основном сервере, так что имеет смысл хранить их локальную копию. Это означает, что вторичный DNS-сервер является первичным сервером для *0.0.127.in-addr.arpa*. Конечно, *возможно* сделать вторичный сервер вторичным и для *0.0.127.in-addr.arpa*, но данные этой зоны никогда не изменяются, так что особой разницы нет.

Установка

Чтобы установить вторичный DNS-сервер, потребуется создать каталог для файлов данных зоны на узле, который будет являться вторичным сервером (к примеру, */var/named*) и скопировать в него файлы */etc/named.conf*, *db.cache* и *db.127.0.0*:

```
# rcp /etc/named.conf host:/etc
# rcp db.cache db.127.0.0 host:db-file-directory
```

Потребуется отредактировать файл */etc/named.conf* на узле вторичного DNS-сервера. Если используется BIND 4, следует заменить каждое вхождение ключевого слова *primary* на *secondary*; это изменение не затрагивает зону *0.0.127.in-addr.arpa*. Перед именем файла в каждой из этих строк необходимо добавить IP-адрес первичного мастер-сервера DNS, который мы уже установили. К примеру, если исходная строка для BIND 4 выглядела так:

```
primary  movie.edu      db.movie.edu
```

измененная будет иметь следующий вид:

```
secondary  movie.edu      192.249.249.3 db.movie.edu
```

Если исходная строка для BIND 8 или 9 выглядела так:

```
zone "movie.edu" in {
       type master;
       file "db.movie.edu";
};
```

следует заменить ключевое слово *master* на *slave* и добавить строку *masters* с указанием IP-адреса основного сервера:

```
zone "movie.edu" in {
       type slave;
       file "bak.movie.edu";
       masters { 192.249.249.3; };
};
```

Так мы говорим DNS-серверу, что он является вторичным для зоны *movie.edu* и что он должен реагировать на изменения этой зоны, хранимой DNS-сервером с IP-адресом 192.249.249.3. Вторичный DNS-сервер будет хранить резервную копию этой зоны в локальном файле *bak.movie.edu*.

Вторичный DNS-сервер Кинематографического университета будет размещен на узле *wormhole.movie.edu*. Файл настройки на узле *terminator.movie.edu* (на котором размещается основной сервер) выглядит так:

```
directory /var/named

primary  movie.edu                db.movie.edu
primary  249.249.192.in-addr.arpa db.192.249.249
primary  253.253.192.in-addr.arpa db.192.253.253
primary  0.0.127.in-addr.arpa     db.127.0.0
cache    .                        db.cache
```

Мы скопировали файлы */etc/named.conf*, *db.cache* и *db.127.0.0* на узел *wormhole.movie.edu* и отредактировали файл настройки в соответствие

с приведенными выше инструкциями. Файл настройки для BIND 4 на узле *wormhole.movie.edu* выглядит следующим образом:

```
directory /var/named

secondary  movie.edu                192.249.249.3 bak.movie.edu
secondary  249.249.192.in-addr.arpa 192.249.249.3 bak.192.249.249
secondary  253.253.192.in-addr.arpa 192.249.249.3 bak.192.253.253
primary    0.0.127.in-addr.arpa     db.127.0.0
cache      .                        db.cache
```

Эквивалентный файл настройки BIND 8 или 9 выглядит так:

```
options {
        directory "/var/named";
};

zone "movie.edu" in {
         type slave;
         file "bak.movie.edu";
         masters { 192.249.249.3; };
};

zone "249.249.192.in-addr.arpa" in {
         type slave;
         file "bak.192.249.249";
         masters { 192.249.249.3; };
};

zone "253.253.192.in-addr.arpa" in {
         type slave;
         file "db.192.253.253";
         masters { 192.249.249.3; };
};

zone "0.0.127.in-addr.arpa" in {
         type master;
         file "db.127.0.0";
};

zone "." in {
         type hint;
         file "db.cache";
};
```

Он инструктирует DNS-сервер, работающий на узле *wormhole.movie.edu*, загружать *movie.edu*, *249.249.192.in-addr.arpa* и *253.253.192.in-addr.arpa* по сети, получая информацию от DNS-сервера с адресом 192.249.249.3 (*terminator.movie.edu*). Помимо этого, сервер сохраняет резервную копию этих файлов в каталоге */var/named*. Возможно, кому-то будет удобнее размещать резервные копии файлов в отдельном подкаталоге. Мы добавляем к файлам уникальный префикс (*bak*), поскольку иногда появлется необходимость вручную удалить все резервные копии. Удобно, когда уже по имени видно, что файлы являются

резервными копиями, и редактировать их не имеет смысла. Позже мы обсудим резервное копирование файлов подробнее.

Теперь запустим вторичный DNS-сервер. Следует проверить наличие сообщений об ошибках в log-файле демона *syslog*, точно так же, как это делалось для основного сервера. Как и для первичного мастер-сервера, команда запуска:

```
# /usr/sbin/named
```

Помимо тестов, уже описанных для первичного DNS-сервера, вторичный следует подвергнуть еще одному. Необходимо проверить, были ли созданы резервные копии файлов. Вскоре после того, как вторичный сервер был запущен на узле *wormhole.movie.edu*, мы обнаружили в каталоге *var/named* файлы *bak.movie.edu*, *bak.192.249.249* и *bak.192.253.253*. Это означает, что вторичный сервер успешно получил зону от мастер-сервера и сохранил ее резервную копию.

Чтобы завершить установку вторичного DNS-сервера, попробуйте произвести поиск для тех же доменных имен, которые использовались при тестировании первичного сервера. На этот раз следует выполнять *nslookup* на узле, на котором работает вторичный DNS-сервер, чтобы именно этот сервер получал запросы. Если вторичный DNS-сервер работает как должен, добавьте соответствующие строки в загрузочные файлы системы, чтобы при загрузке системы DNS-сервер стартовал автоматически, а также воспользуйтесь командой *hostname(1)* для установки доменного имени.

Резервные копии файлов

Дополнительные DNS-серверы *не обязаны* сохранять резервную копию зональных данных. Если существует резервная копия, вторичный DNS-сервер читает ее при запуске, а позже проверяет первичный мастер-сервер DNS на наличие более свежей копии вместо того, чтобы сразу загружать копию зоны с основного сервера. Если на основном контрольном сервере имен доступна более свежая копия, вторичный сервер получает ее и сохраняет в качестве резервной копии.

Зачем нужна резервная копия? Предположим, первичный DNS-мастер-сервер недоступен на момент запуска вторичного сервера. В таком случае, вторичный сервер не сможет произвести передачу зоны и не будет функционировать в качестве DNS-сервера для зоны, пока основной сервер не станет доступен. В случае наличия резервной копии, вторичный сервер обладает информацией, хотя она может быть немного устаревшей. И поскольку при использовании резервного копирования вторичный DNS-сервер становится менее зависим от основного, он более надежен в работе.

Чтобы избежать создания резервных копий, следует удалить имя файла в конце строк с ключевым словом *secondary* (файл настройки BIND 4).

Из файлов настройки для BIND 8 или 9 следует удалить строку *file*. При этом мы советуем настраивать дополнительные DNS-серверы таким образом, чтобы они создавали резервные копии. Сохранять резервные копии файлов данных зоны – ничего не стоит, зато недешево обойдется ситуация, когда резервная копия спасла бы положение, но ее нет.

Значения SOA

Помните вот эту SOA-запись?

```
movie.edu. IN SOA terminator.movie.edu. al.robocop.movie.edu. (
                          1         ; Порядковый номер
                          3h        ; Обновление через 3 часа
                          1h        ; Повторение попытки через 1 час
                          1w        ; Устаревание через 1 неделю
                          1h )      ; Отрицательное TTL в 1 час
```

Мы так и не объяснили, для чего нужны значения в скобках.

Порядковый номер относится ко всем данным в пределах зоны. Мы начали с единицы, что вполне логично. Но многие люди находят более удобным использовать в качестве порядкового номера даты, к примеру 1997102301. Это дата в формате ГГГГММДДNN, где ГГГГ – год, ММ – месяц года, ДД – день месяца, а NN – счетчик числа изменений зональных данных в этот день. Порядок полей менять нельзя, поскольку только этот порядок приводит к увеличению значения порядкового номера при смене даты. Это очень важно: какой бы формат не использовался, порядковый номер должен увеличиваться при обновлении данных зоны.

Когда вторичный DNS-сервер устанавливает соединение со своим мастером на предмет получения информации о зоне, прежде всего он запрашивает порядковый номер данных. Если порядковый номер данных зоны у вторичного DNS-сервера меньше, чем порядковый номер данных мастера, считается, что зональные данные вторичного сервера устарели. В этом случае вторичный сервер получает новую копию зоны. Если при запуске вторичного сервера не существует резервной копии, происходит безусловная загрузка зоны. Как можно догадаться, при изменении файлов данных зоны на первичном мастер-сервере следует увеличивать порядковый номер. Изменение файлов данных зоны описано в главе 7.

Следующие четыре поля определяют различные временные интервалы, причем по умолчанию значения указываются в секундах:

Обновление (refresh)

Интервал обновления инструктирует вторичный DNS-сервер, с какой частотой следует проверять актуальность информации для зоны. Чтобы читатели получили представление о нагрузке, которую

создает это значение, сообщаем, что вторичный сервер при каждом обновлении делает один запрос SOA-записи для зоны. Выбранное значение, три часа, умеренно агрессивно. Большинство пользователей смирятся с задержкой в половину рабочего дня, ожидая, когда их рабочие станции станут частью сети. Если речь идет о ежедневных процедурах, касающихся DNS, вполне можно увеличить значение до восьми часов. Если же данные зоны меняются не очень часто, а все вторичные DNS-серверы удалены на большие расстояния (как корневые DNS-серверы), имеет смысл задуматься о бо́льшем значении, скажем, интервале в 24 часа.

Повторение попытки (retry)

Если по истечении интервала обновления вторичный сервер не может достучаться до своего мастера (который, вполне возможно, не работает на этот момент), он повторяет попытки через равные интервалы времени, определяемые данным значением. В обычной ситуации интервал повторения попытки короче, чем интервал обновления, но это необязательно.

Устаревание (expire)

Если вторичный DNS-сервер не может соединиться с основным в течение указанного числа секунд, данные зоны на вторичном сервере устаревают. Устаревание зоны означает, что вторичный сервер перестает отвечать на запросы по этой зоне, поскольку зональные данные настолько утратили актуальность, что не могут быть полезными. По сути дела, это поле определяет момент, когда данные становятся настолько старыми, что лучше не использовать их вовсе. Интервалы устаревания порядка недели – обычное дело, они могут быть длиннее (до месяца) в случаях, если существуют проблемы связи с первичным источником информации. Интервал устаревания всегда должен быть гораздо длиннее, чем интервалы обновления и повторения попытки; в противном случае, данные зоны будут устаревать еще до попытки их обновления.

Отрицательное TTL

TTL – это время жизни (*time to live*). Это значение относится ко всем отрицательным ответам DNS-серверов, авторитативных для данной зоны.

Для версий BIND более старых, чем 8.2, последнее поле SOA-записи определяет *оба* значения – стандартное (по умолчанию) и отрицательное значение времени жизни информации для зоны.

Те из читателей, кто знаком с предыдущими изданиями этой книги, могут заметить, что изменился формат значений полей в SOA-записях. Когда-то BIND понимал только значения, определяемые в секундах, – для всех четырех описанных полей. (Как следствие, выросло це-

лое поколение администраторов, которые знают, что в неделе 608400 секунд.) Теперь, при работе со всеми серверами, кроме самого старого (BIND 4.8.3), можно указывать значения в других единицах измерения, причем не только в SOA-записи, но и в качестве аргумента управляющего оператора TTL, что мы видели выше по тексту. К примеру, задать трехчасовой интервал обновления можно значениями *3h*, *180m* и даже *2h60m*. Дни обозначаются буквой *d*, а недели – буквой *w*.

Корректные значения SOA-записи зависят от конкретного случая. В целом, более длительные интервалы времени снижают нагрузку на DNS-серверы и замедляют распространение изменений, более короткие увеличивают нагрузку и ускоряют распространение изменений. Значения, которые мы используем в этой книге, вполне подойдут для большинства установок. Документ RFC 1537 рекомендует следующие значения для DNS-серверов высшего уровня:

```
Обновление                24 hours
Повторение попытки        2 hours
Устаревание               30 days
Стандартное TTL           4 days
```

Существует одна особенность реализации, о которой следует знать. Версии BIND, которые предшествовали версии 4.8.3, перестают отвечать на запросы в процессе загрузки зоны. В результате пакет BIND был модифицирован с целью распределения загрузок зоны и сокращения интервалов недоступности. Поэтому, даже в случае, когда установлены короткие интервалы обновления, вторичные DNS-серверы могут производить синхронизацию реже, чем предписано этими интервалами. BIND пытается произвести загрузку зоны определенное число раз, а затем делает 15-минутный перерыв, прежде чем повторить попытки.

Итак, мы рассказали о том, как DNS-серверы обновляют свои данные... но в BIND 8 и 9 механизм распространения данных зоны другой! Опрос основных серверов все еще доступен, но в BIND 8 и 9 существует механизм уведомления об изменениях в зональных данных. Если первичный мастер-сервер и все вторичные являются серверами пакета BIND версии 8 или 9, первичный мастер-сервер DNS уведомляет вторичные серверы об изменениях зоны в течение пятнадцати минут после загрузки новой копии этой зоны. Уведомление заставляет вторичный сервер сократить интервал обновления и попытаться загрузить зону немедленно. Более подробно мы обсудим этот механизм в главе 10.

Несколько мастер-серверов

Существуют ли другие способы сделать работу вторичных DNS-серверов более надежной? Разумеется: можно указать до десяти IP-адресов мастер-серверов. В файле настройки BIND 4 следует добавить эти адреса после первого IP-адреса, но до имени файла резервной копии. В фай-

ле настройки BIND версий 8 или 9 следует добавить эти адреса после первого IP-адреса, используя точку с запятой в качестве разделителя:

```
masters { 192.249.249.3; 192.249.249.4; };
```

Вторичный сервер будет перебирать мастер-серверы из списка, пока не получит ответ. Вплоть до BIND версии 8.1.2 вторичный DNS-сервер всегда производил передачу зоны с первого ответившего мастер-сервера, если данные на этом мастере имели больший порядковый номер. Последующие DNS-серверы опрашивались только в случае отсутствия ответов от предшествующих. Начиная с BIND версии 8.2, вторичный сервер опрашивает все перечисленные мастер-серверы DNS и производит передачу зоны с сервера, данные которого имеют наибольший порядковый номер. Если таких серверов несколько, вторичный сервер получает данные зоны от первого (в порядке чтения списка) из этих серверов.

Изначально эта возможность предназначалась для перечисления всех IP-адресов узла, на котором работает первичный мастер-сервер DNS зоны, если хост был подключен к нескольким сетям. Но поскольку невозможно определить, является ли сервер, с которым установлена связь, первичным мастером или вторичным, можно перечислить IP-адреса узлов, на которых работают вторичные DNS-серверы зоны, если это имеет какой-либо смысл в существующей сети. В этом случае, если первый первичный мастер-сервер DNS не работает или недоступен, вторичный DNS-сервер может получить зону от другого мастер-сервера DNS.

Добавление зон

Теперь, когда у вас есть работающие DNS-серверы, можно подумать о поддержке нескольких зон. Что следует для этого сделать? Ничего особенного. Все, что нужно сделать, – добавить операторы *primary* или *secondary* (BIND 4), или *zone* (BIND 8 и 9) к файлу настройки. Можно даже добавлять операторы *secondary* в файл настройки первичного мастер-сервера DNS, а *primary* – в файл настройки вторичного DNS-сервера. (Возможно, читатели уже заметили, что вторичный DNS-сервер является первичным мастером для зоны *0.0.127.in-addr.arpa*.)

А теперь будет полезно повторить то, что мы уже говорили ранее. Несколько глупо называть *конкретный* DNS-сервер первичным мастер-сервером DNS или вторичным DNS-сервером. Серверы имен могут быть – и обычно бывают – авторитативными для нескольких зон. Сервер имен может являться первичным мастером для одной зоны и вторичным для другой. Однако большинство DNS-серверов для большинства загружаемых ими зон являются либо первичными мастерами, либо вторичными. Поэтому, если мы называем отдельный DNS-сервер первичным мастером или вторичным, то имеем в виду, что он является первичным или вторичным мастером для *большинства* загружаемых им зон.

Что дальше?

В этой главе мы рассказали о том, как можно создать файлы данных для зоны путем конвертирования файла */etc/hosts* в эквивалентные данные для DNS-сервера, а также как установить первичный и вторичный DNS-серверы. Чтобы закончить с созданием локальных зон, предстоит еще выполнить кое-какую работу: необходимо модифицировать данные зоны для работы с электронной почтой и настроить прочие узлы зоны на использование новых DNS-серверов. Возможно, понадобится также организовать работу еще нескольких DNS-серверов. Все эти темы мы рассматриваем в последующих главах.

- *MX-записи*
- *И все-таки, что такое почтовый ретранслятор?*
- *MX-алгоритм*

5

DNS и электронная почта

Тут Алиса почувствовала, что глаза у нее слипаются. Она сонно бормотала:
– Едят ли кошки мошек? Едят ли кошки мошек?
Иногда у нее получалось:
– Едят ли мошки кошек?
Алиса не знала ответа ни на первый, ни на второй вопрос, и потому ей было все равно, как их ни задать.

Вероятно, на вас уже тоже напала сонливость, вызванная длинной предыдущей главой. К счастью, эта глава посвящена теме, которая должна заинтересовать системных и почтовых администраторов: взаимодействию DNS и электронной почты. И даже если эта тема вас не интересует, времени на ее обсуждение потребуется гораздо меньше, чем на предыдущую главу.

Одним из преимуществ DNS перед таблицами узлов является усовершенствованная поддержка маршрутизации почты. В те времена, когда почтовым программам был доступен только файл *HOSTS.TXT* (и его наследник, */etc/hosts*), они в лучшем случае могли попытаться доставить почту по IP-адресу узла. В случае неудачи оставалось только продолжать периодические попытки отправить письмо или вернуть его отправителю.

В DNS предусмотрены способы указания резервных узлов, через которые может производиться доставка почтовых сообщений, а также передачи вопросов, связанных с доставкой почты другим узлам. Это, к примеру, позволяет возлагать обработку почты для узлов, являющихся бездисковыми станциями, на обслуживающий их сервер.

В отличие от таблиц узлов, DNS позволяет использовать в адресах электронной почты произвольные имена. Возможно – что и делает большинство организаций в сети Интернет – использовать доменное

имя основной зоны прямого отображения в адресе электронной почты. Также можно добавлять в произвольную зону доменные имена, которые будут использоваться только в адресах электронной почты и не связаны с какими-либо конкретными узлами сети. Кроме того, один логический адрес электронной почты может представлять несколько почтовых серверов. Напротив, при использовании таблиц узлов в почтовых адресах можно было употреблять только имена хостов, и все.

В целом, перечисленные возможности предоставляют администраторам гораздо больше свободы в настройке маршрутизации электронной почты в сетях.

MX-записи

Для реализации усовершенствованной маршрутизации электронной почты в DNS используется единственный тип RR-записи: MX-запись. Изначально функции MX-записи были поделены между двумя записями: MD-записью (mail destination) и MF-записью (mail forwarder). MD-запись содержала конечный адрес, по которому следует доставить почтовое сообщение, адресованное определенному домену; MF-запись содержала информацию об узле, который передаст почту конечному адресату, если этот адресат не может получить почту обычным маршрутом.

Ранние опыты с использованием DNS в сети ARPAnet показали, что разделение этих функций на практике не очень эффективно. Почтовой программе требовалось наличие как MD-, так и MF-записей для каждого доменного имени, чтобы принять решение о том, куда посылать почтовые сообщения – одной из них было бы недостаточно. Явный же поиск информации какого-либо одного типа (будь то MD или MF) приводил к кэшированию DNS-сервером записей только этого типа. Так что почтовым программам приходилось либо делать два запроса (по одному на каждый тип записи), либо просто отвергать кэшированные результаты. В результате затраты на маршрутизацию почты значительно превышали затраты на работу остальных служб, что, в конечном итоге, было сочтено недопустимым.

Для решения проблемы эти типы записей были объединены в один тип – MX. Почтовые программы стали обходиться набором MX-записей для конкретного доменного имени в целях принятия решения по маршрутизации почтовых сообщений. Использование кэшированных MX-записей стало абсолютно допустимым, при условии соблюдения значений TTL.

MX-записи определяют *почтовый ретранслятор* (*mail exchanger*) для доменного имени; то есть узел, который обработает *или* передаст дальше почтовые сообщения, предназначенные адресату в указанном домене (например, через брандмауэр). «Обработка» почты относится либо к

доставке почтовых сообщений конечному адресату, либо к передаче их через шлюз другому почтовому транспорту, например X.400 или Microsoft Exchange. «Ретрансляция» означает передачу почты конечному домену либо другому почтовому ретранслятору, расположенному «ближе» к конечному адресату в мерах расстояний STMP (Simple Mail Transfer Protocol, интернет-протокол доставки почтовых сообщений). Иногда при ретрансляции почтовые сообщения на некоторое время ставятся в очередь почтового ретранслятора.

Чтобы предотвратить появление петель маршрутизации почты, в записях типа MX, помимо доменного имени почтового ретранслятора, содержится вторичный параметр: *приоритет* (*preference value*). Приоритет – это шестнадцатибитное положительное целое (может принимать значения от 0 до 65535), которое определяет приоритет для почтового ретранслятора. Скажем, вот такая MX-запись:

```
peets.mpk.ca.us.    IN    MX    10 relay.hp.com.
```

идентифицирует *relay.hp.com* в качестве почтового ретранслятора для *peets.mpk.ca.us* с приоритетом 10.

При выборе приоритета почтовых ретрансляторов для конкретного адреса определяют, в каком порядке они будут использоваться почтовой программой. Само по себе значение не особенно важно, важно его отношение к приоритетам, определенным для остальных почтовых ретрансляторов: больше оно или меньше прочих значений? В случае отсутствия других записей

```
plange.puntacana.dr.  IN  MX  1 listo.puntacana.dr.
plange.puntacana.dr.  IN  MX  2 hep.puntacana.dr.
```

абсолютно эквивалентно:

```
plange.puntacana.dr.  IN  MX  50  listo.puntacana.dr.
plange.puntacana.dr.  IN  MX  100 hep.puntacana.dr.
```

Почтовые программы должны пытаться доставить почту почтовым ретрансляторам, начиная с тех, которые имеют *наименьшие* значения приоритета. Может показаться не очень естественным, что наиболее предпочтительный ретранслятор имеет наименьший приоритет. Но, поскольку приоритет суть беззнаковая величина, это позволяет присвоить «лучшему» почтовому ретранслятору приоритет 0.

В случае невозможности доставки почты через наиболее предпочтительные почтовые ретрансляторы почтовой программе следует попытаться произвести доставку через менее предпочтительные ретрансляторы (имеющие более высокие приоритеты), выбирая их в порядке повышения приоритетов. Короче говоря, почтовые программы должны опрашивать более предпочтительные ретрансляторы прежде менее предпочтительных. Приоритеты могут совпадать для двух или нескольких ретрансляторов, что позволяет почтовым программам де-

лать собственный выбор предпочтительных ретрансляторов.[1] Почтовая программа должна опросить все почтовые ретрансляторы с определенным приоритетом, прежде чем переходить к следующему, более «высокому» значению.

Так, к примеру, могут выглядеть MX-записи для *oreilly.com*:

```
oreilly.com.    IN    MX    0  ora.oreilly.com.
oreilly.com.    IN    MX    10 ruby.oreilly.com.
oreilly.com.    IN    MX    10 opal.oreilly.com.
```

Эти MX-записи дают почтовым программам указания пытаться доставить почту для *oreilly.com* путем передачи:

1. Сначала *ora.oreilly.com*
2. Затем один из *ruby.oreilly.com* или *opal.oreilly.com*, и наконец
3. Оставшийся ретранслятор с приоритетом 10 (тот, который не использовался на шаге 2)

Разумеется, после доставки почты одному из почтовых ретрансляторов *oreilly.com* процесс опроса прекращается. После успешной доставки почты через *ora.oreilly.com* нет смысла отправлять ее через *ruby.oreilly.com* или *opal.oreilly.com*.

Обратите внимание, что *oreilly.com* – не узел сети; это доменное имя основной зоны прямого отображения компании O'Reilly & Associates. Компания O'Reilly & Associates использует это доменное имя в качестве пункта назначения почты для всех, кто в ней работает. Гораздо легче запомнить единственный e-mail «адрес», *oreilly.com*, чем помнить на каком из узлов – *ruby.oreilly.com* или *amber.oreilly.com* – в действительности создана учетная запись электронной почты для каждого конкретного сотрудника.

Разумеется, такая система требует от почтовой программы на *ora.oreilly.com* определенных действий по отслеживанию учетных записей электронной почты сотрудников и узлов, на которых эти учетные записи созданы. Обычно это достигается созданием на *ora.oreilly.com* специального файла *псевдонимов* (*aliases*), который используется для передачи почтовых сообщений конечным адресатам.

Что произойдет, если ни одной MX-записи для пункта назначения не существует, но присутствует хотя бы одна A-запись? Неужели почтовая программа попросту не доставит почту в пункт назначения? Разумеется, более поздние версии программы *sendmail* можно собрать именно с таким алгоритмом работы. Тем не менее большинство поставщиков собирает *sendmail* с более мягким алгоритмом: если не существуют MX-записи, но существует хотя бы одна A-запись, будут делать-

[1] Последняя версия (Version 8) программы *sendmail* на деле выбирает ретранслятор из имеющих одинаковые приоритеты случайным образом.

ся попытки доставить почту для указанного адреса. Версия 8 программы *sendmail*, собранная «из коробки», пытается доставить почту по указанным адресам в отсутствие MX-записей. Сверьтесь с документацией, включенной в поставку системы, если необходимо уточнить, посылает ли почтовый сервер сообщения по указанным адресам в случае наличия одной лишь адресной записи.

Несмотря на тот факт, что практически все почтовые программы доставляют сообщения по соответствующим адресам даже в случае присутствия только адресной записи, имеет смысл создать хотя бы по одной MX-записи для каждого допустимого пункта назначения. При необходимости отправить почтовые сообщения *sendmail прежде всего* всегда проверяет наличие MX-записей для указанных пунктов назначения. Если таковые отсутствуют, DNS-сервер – обычно один из авторитативных – все равно должен среагировать на этот запрос, после чего *sendmail* переходит к поиску A-записей.

Это занимает дополнительное время, замедляет процесс доставки почтовых сообщений и создает дополнительную нагрузку на авторитативные DNS-серверы вашей зоны. Если же просто добавить MX-запись для каждого пункта назначения, указав доменное имя, которое отображается в тот же самый адрес, что и возвращаемый поиском по адресным записях, программе *sendmail* останется отправить единственный запрос, после чего локальный DNS-сервер почтовой программы кэширует эту MX-запись для последующего использования.

И все–таки, что такое почтовый ретранслятор?

Концепция почтового ретранслятора, вероятно, является новой для большинства читателей, поэтому давайте изучим ее чуть более подробно. Воспользуемся простейшей аналогией. Представьте себе, что почтовый ретранслятор – это аэропорт. Вместо того чтобы создавать MX-записи, объясняющие почтовым программам, куда следует отправлять сообщения, вы можете порекомендовать вашим друзьям аэропорт, в который удобнее всего прилететь, если они соберутся навестить вас.

Предположим, вы живете в Лос-Гатосе, в Калифорнии. Ближайший аэропорт, в котором могут совершить посадку ваши друзья, – в Сан-Хосе, второй по удаленности – в Сан-Франциско, а третий – в Окленде. (Не будем сейчас заострять внимание на прочих факторах, как то цены на билеты, трафик в районе залива и т. д.) Все еще не понимаете? Тогда представьте себе вот такую схему:

```
los-gatos.ca.us.    IN    MX    1 san-jose.ca.us.
los-gatos.ca.us.    IN    MX    2 san-francisco.ca.us.
los-gatos.ca.us.    IN    MX    3 oakland.ca.us.
```

Данный MX-список – это просто упорядоченный перечень пунктов назначения, который предписывает почтовым программам (вашим

друзьям), куда отправлять сообщения (лететь), если они хотят доставить сообщения по нужному адресу (посетить вас). Значения предпочтения показывают, насколько желательно пользоваться тем или иным маршрутом, их можно считать логическими «расстояниями» до конечного пункта путешествия (причем измеряемыми в произвольных единицах) либо просто участниками хит-парада приближенности почтовых ретрансляторов к конечному адресу.

Этим списком вы говорите: «Попытайтесь прилететь в Сан-Хосе, а если туда попасть невозможно, попробуйте в Сан-Франциско или Окленд, именно в таком порядке». Помимо прочего, в этой фразе содержится *еще* и указание в случае прилета в Сан-Франциско воспользоваться самолетом местной авиалинии, чтобы добраться до Сан-Хосе. Если же речь идет об Окленде, следует попытаться аналогичным образом попасть в Сан-Хосе, либо в Сан-Франциско.

Итак, каковы же признаки хорошего почтового ретранслятора? Они соответствуют признакам хорошего аэропорта:

Масштаб

Если вы собрались в Лос-Гатос, то вряд ли захотите совершить посадку в крохотном аэропорту Райд-Хиллвью, не приспособленном для приема больших самолетов и большого числа пассажиров. (Вероятно, вы предпочтете, чтобы ваш большой реактивный самолет совершил посадку в аэропорту международного класса.) Точно так же не следует использовать чахлый, не способный справиться с нагрузкой узел сети в качестве почтового ретранслятора.

Время непрерывной работы

Вам ведь вряд ли придет в голову лететь до международного аэропорта Денвера зимой? Значит, вы не станете пользоваться и узлом, который редко находится в рабочем состоянии или редко выполняет функции почтового ретранслятора.

Доступность

Если ваши родственники летят издалека, следует убедиться, что им будет доступен прямой рейс хотя бы до одного из аэропортов в списке. Если они летят из Хельсинки, нельзя ограничивать выбор только Сан-Хосе и Оклендом. Точно так же следует убедиться, что по меньшей мере один почтовый ретранслятор для вашего узла доступен для всех ваших потенциальных корреспондентов.

Управление и администрирование

От того, насколько качественно управление аэропортом, зависит и ваша безопасность, и общее впечатление от его посещения. Об этих факторах стоит задуматься при выборе ретранслятора. Конфиденциальность почты, скорость ее доставки в обычной ситуации, порядок работы с ней в случае недоступности ваших узлов зависят ис-

ключительно от квалификации администраторов почтовых ретрансляторов.

Запомните этот пример, мы к нему еще вернемся.

MX-алгоритм

Такова основная идея MX-записей и почтовых ретрансляторов, но есть и кое-что еще, о чем следует знать. Чтобы избежать петель маршрутизации, почтовые программы должны использовать чуть более сложный алгоритм, чем в случае принятия решения о том, куда направлять почтовые сообщения.[1]

Представим, что может произойти, если почтовые программы не будут учитывать появление петель маршрутизации. Допустим, необходимо отправить на адрес *nuts@oreilly.com* почтовое сообщение, содержащее негодующий отзыв об этой книге. К сожалению, узел *ora.oreilly.com* в данный момент недоступен. Никаких проблем! Что написано в MX-записях для oreilly.com?

```
oreilly.com.    IN    MX    0  ora.oreilly.com.
oreilly.com.    IN    MX    10 ruby.oreilly.com.
oreilly.com.    IN    MX    10 opal.oreilly.com.
```

Почтовая программа принимает решение отправить почту через узел *ruby.oreilly.com*, который исправно функционирует. Почтовая программа на *ruby.oreilly.com* пытается передать почту узлу *ora.reilly.com*, естественно, безрезультатно, поскольку этот узел пребывает в нерабочем состоянии. И что дальше? Если на *ruby.oreilly.com* не предусмотрена проверка разумности действий, будет предпринята попытка передать почту узлу *opal.oreilly.com*, либо самому узлу *ruby.oreilly.com*. Разумеется, это не даст никакого результата в плане доставки почтовых сообщений. Если *ruby.oreilly.com* отправит сообщение себе, получится петля маршрутизации. Если *ruby.oreilly.com* отправит сообщение узлу *opal.oreilly.com*, *opal.oreilly.com* либо вернет это сообщение узлу *ruby.oreilly.com*, либо пошлет себе, и в этом случае снова получится петля маршрутизации.

Чтобы предотвратить подобные вещи, почтовые программы исключают из рассмотрения некоторые MX-записи перед принятием решения о том, куда посылать сообщения. Почтовая программа сортирует список MX-записей по приоритетам и производит поиск канонического доменного имени узла, на котором запущена сама программа. Если текущий узел является почтовым ретранслятором, программа исключает эту MX-запись, а также все MX-записи с приоритетами бо́льшими

[1] Этот алгоритм основан на документе RFC 974, в котором описана почтовая маршрутизация в сети Интернет.

или равными значению из первой исключаемой записи (то есть исключает менее предпочтительные и столь же предпочтительные почтовые ретрансляторы). Таким образом, предотвращается отправка почты узлом самому себе либо узлам, которые расположены «дальше» от конечного пункта назначения.

Вернемся к аналогии с аэропортом. На этот раз представьте себе, что вы – пассажир самолета (почтовое сообщение), который пытается попасть в Грили, штат Колорадо. Прямого рейса до Грили нет, но можно воспользоваться рейсом до города Форт-Коллинз или до Денвера (двумя из более предпочтительных почтовых ретрансляторов). Поскольку Форт-Коллинз ближе к Грили, вы выбираете именно этот рейс.

Итак, очутившись в Форт-Коллинзе, вы понимаете, что смысла лететь в Денвер нет, поскольку это отдалит вас от пункта назначения (и будет выбором менее предпочтительного почтового маршрутизатора). А лететь из Форт-Коллинза в Форт-Коллинз и вовсе глупо. Так что единственным приемлемым рейсом остается прямой рейс из Форт-Коллинза в Грили. Таким образом, вы можете не рассматривать рейсы до менее предпочтительных пунктов, чтобы предотвратить рейсовые петли и не терять лишнее время на дорогу.

Одно предостережение: большинство почтовых программ производят поиск *исключительно канонического* доменного имени текущего узла в списке MX-записей. Псевдонимы (доменные имена в левой части CNAME-записей) не рассматриваются. Нет никаких гарантий, что почтовая программа сможет обнаружить свой узел в списке MX-записей, если в этих записях не были использованы канонические доменные имена; в таком случае существует риск появления петель маршрутизации.

В том случае, если почтовый ретранслятор был определен через псевдоним и наивно пытается доставить почту самому себе, будет обнаружена ошибка и почта вернется со следующим сообщением об ошибке:

```
554 MX list for movie.edu points back to relay.isp.com
554 <root@movie.edu>... Local configuration error
```

Это сообщение заменило замысловатое «I refuse to talk to myself» («Отказываюсь разговаривать с собой») в более новых версиях программы *sendmail*. Мораль: всегда используйте каноническое доменное имя почтового ретранслятора в MX-записях.

И еще одно предостережение: для узлов, перечисленных в качестве почтовых ретрансляторов, *должны* существовать адресные записи. Почтовая программа должна быть в состоянии определить адрес почтового ретранслятора, чтобы попытаться доставить через него сообщения.

Вспомним пример с *oreilly.com*. Узел *ruby.oreilly.com* получает сообщение от читателя, почтовая программа сверяется со списком MX-записей:

```
oreilly.com.    IN    MX    0  ora.oreilly.com.
oreilly.com.    IN    MX    10 ruby.oreilly.com.
oreilly.com.    IN    MX    10 opal.oreilly.com.
```

Обнаружив доменное имя текущего узла в списке, почтовая программа на *ruby.oreilly.com* исключает из рассмотрения записи с приоритетом 10 или выше (записи выделены жирным шрифтом):

```
oreilly.com.    IN    MX    0  ora.oreilly.com.
oreilly.com.    IN    MX    10 ruby.oreilly.com.
oreilly.com.    IN    MX    10 opal.oreilly.com.
```

после чего остается только:

```
oreilly.com.    IN    MX    0 ora.oreilly.com.
```

Поскольку узел *ora.oreilly.com* неработоспособен, *ruby.oreilly.com* откладывает доставку сообщения, помещая его в очередь.

А что происходит в случае, когда почтовая программа обнаруживает, что текущий узел имеет наивысший приоритет (наименьшее число в MX-записи) и что в связи с этим следует исключить все MX-записи из списка? Некоторые почтовые программы пытаются произвести прямую доставку по IP-адресу конечного узла в качестве последнего средства. Однако в большинстве почтовых программ такая ситуация считается ошибкой. Она может возникать в том случае, если DNS считает, что почтовая программа должна обрабатывать (а не просто передавать дальше) почту для определенного узла, но почтовая программа не была соответствующим образом настроена. Или же, если администратор некорректно расположил MX-записи, используя неверные приоритеты.

Скажем, ребята, которые управляют доменом *acme.com*, добавили MX-запись, чтобы передавать почту, адресованную *acme.com*, почтовой программе своего интернет-провайдера:

```
acme.com.    IN    MX    10 mail.isp.net.
```

Обычно почтовая программа должна быть настроена так, чтобы распознавать собственные псевдонимы и имена узлов, для которых она производит обработку почты. Если почтовая программа узла *mail.isp.net* не была сконфигурирована так, чтобы распознавать почтовые адреса домена *acme.com* в качестве локальных, она будет считать, что пришел запрос на передачу почты, и попытается отправить сообщения почтовому ретранслятору, который находится ближе к пункту назначения.[1] После изучения MX-записей для *acme.com* программа обнару-

[1] Разумеется, речь идет о том, что почтовая программа *mail.isp.net* вообще позволяет передачу почты для неизвестных ей доменных имен. В противном случае в доставке почтовых сообщений просто будет отказано.

жит, что текущий узел является наиболее предпочтительным почтовым ретранслятором, после чего вернет почтовое сообщение отправителю с уже знакомой нам ошибкой:

```
554 MX list for acme.com points back to mail.isp.com
554 <root@acme.com>... Local configuration error
```

Во многих версиях программы *sendmail* используется класс *w* или файловый класс *w* для определения списка «местных» пунктов доставки. В зависимости от существующего файла *sendmail.cf*, добавление псевдонима может сводиться просто к добавлению к этому файлу строки:

```
Cw acme.com
```

Если почтовая программа использует иной почтовый транспорт (скажем, UUCP) для доставки почты узлам, для которых является почтовым ретранслятором, вероятно, потребуется более сложная настройка.

Внимательные читатели, наверно, заметили, что для приоритетов мы используем числа, кратные 10. Это удобный подход, поскольку позволяет временно добавлять MX-записи с промежуточными значениями, не меняя значений всех остальных записей, и больше никакого волшебства здесь нет. Мы могли бы с тем же успехом использовать приращение 1 или 100.

- *DNS-клиент*
- *Пример настройки клиента*
- *Как упростить себе жизнь*
- *Специфика настройки различных систем*

6

Конфигурирование узлов

Общество, собравшееся на берегу, имело весьма неприглядный вид: перья у птиц взъерошены, шерстка у зверьков промокла насквозь. Вода текла с них ручьями, всем было холодно и неуютно.

Итак, DNS-серверы для ваших зон заработали, и теперь необходимо настроить узлы сети таким образом, чтобы они при работе пользовались этими серверами. В частности, необходимо настроить DNS-клиенты этих узлов. Следует также проверить такие файлы, как *hosts.equiv* и *.rhosts* с целью внесения изменений, продиктованных использованием DNS; возможно, понадобится преобразовать некоторые имена узлов в этих файлах в доменные. Можно воспользоваться псевдонимами – для удобства пользователей и минимизации шока от перехода на DNS.

Все темы, включая настройку клиента в распространенных вариантах системы Unix, а также в Microsoft Windows 95, Windows NT и Windows 2000, рассмотрены в этой главе.

DNS-клиент

Мы рассказывали про DNS-клиенты в главе 2 «Как работает DNS», но без особых подробностей. Они отвечают за перевод запроса программы в запрос к DNS-серверу и за перевод ответа сервера в ответ для программы.

Пока еще мы не затрагивали настройку DNS-клиентов, поскольку не было случая. Когда мы устанавливали наши DNS-серверы в главе 4 «Установка BIND», для работы было более чем достаточно стандартного поведения клиента. Но если бы мы хотели заставить DNS-клиент выполнять какие-то действия помимо стандартных либо вести себя несколько иначе, нам пришлось бы произвести его настройку.

Есть одно обстоятельство, о котором следует упомянуть прямо сейчас: в следующих разделах мы будем описывать поведение обычного DNS-клиента BIND версии 8.2.3 в отсутствие других служб имен. Не все клиенты ведут себя так же; некоторые поставщики систем включают клиент, основанный на более ранних версиях кода DNS, а некоторые реализовали дополнительную функциональность, позволяющую изменить алгоритм работы клиента. В случаях, когда по нашему мнению это имеет значение, мы будем описывать разницу в поведении клиента BIND 8.2.3 и более ранних версий системы, в частности 4.8.3 и 4.9, которые все еще включались во многие системы на момент последнего обновления этой книги. Различные расширения мы рассмотрим позже в этой главе.

Что же именно можно настроить в DNS-клиенте? Большинство клиентов позволяют изменять по меньшей мере три аспекта поведения: локальное доменное имя, список поиска и сервер (серверы) имен, который (которые) опрашивает клиент. Во многих Unix-системах доступны для настройки дополнительные аспекты поведения, что связано с наличием нестандартных расширений DNS. Иногда эти расширения необходимы, чтобы иметь дело с некоторыми программами, скажем, Сетевой информационной службой Sun (NIS), а иногда они добавляются просто для увеличения стоимости системы.[1]

Настройка DNS-клиента практически полностью ограничивается редактированием файла */etc/resolv.conf* (как вариант – */usr/etc/resolv.conf* или нечто вроде, более подробная информация содержится в руководстве по *программе-клиенту* (*resolver*), как правило, в разделах 4 или 5). Существует пять основных *инструкций*, которые можно использовать в пределах *resolv.conf*: *domain*, *search*, *nameserver*, *sortlist* и *options*. Эти инструкции контролируют поведение DNS-клиента. В некоторых реализациях Unix существуют и другие инструкции, мы рассмотрим их в конце главы.

Локальное доменное имя

Локальное доменное имя – это доменное имя, в пределах которого существует DNS-клиент. В большинстве случаев это имя совпадает с доменным именем зоны, в которой расположен хост клиента. К примеру, для клиента на узле *terminator.movie.edu*, вероятно, используется *movie.edu* в качестве локального доменного имени.

Клиент использует локальное доменное имя для интерпретации имен, не являющихся абсолютными. Например, при добавлении строки:

```
relay bernie
```

[1] Служба NIS раньше носила имя Yellow Pages (Желтые страницы) или YP, но была переименована, поскольку оказалось, что права владения именем Yellow Pages принадлежат телефонной компании Великобритании.

к файлу *.rhosts* имя *relay* считается расположенным в локальном домене. Это гораздо более логично, чем давать пользователю *bernie* доступ к каждому узлу сети Интернет, доменное имя которого начинается с *relay*. Прочие файлы авторизации, такие как *hosts.equiv* и *hosts.lpd*, также следуют этому правилу.

В обычной ситуации локальное доменное имя определяется по имени узла; локальным доменным именем считается часть имени, следующая за первой точкой «.». Если имя не содержит точки, то в качестве локального домена принимается корневой. Таким образом, имя узла (*hostname*) *asylum.sf.ca.us* подразумевает локальное доменное имя *sf.ca.us*, а имя узла *dogbert* – корневой локальный домен, что, скорее всего, неверно, если учесть, насколько мало количество узлов с доменным именем из одной метки.[1]

Локальное доменное имя можно также установить с помощью инструкции *domain* в файле *resolv.conf*. Инструкция *domain* имеет для клиента более высокий приоритет, чем извлечение локального доменного имени из имени узла.

У инструкции *domain* очень простой синтаксис, но следует использовать ее правильно, поскольку клиент не сообщает об ошибках. Ключевое слово *domain* начинается в первой колонке строки, за ним может следовать один или несколько пробелов или символов табуляции. Строка завершается локальным доменным именем. Локальное доменное имя не должно заканчиваться точкой. Вот пример правильной инструкции:

```
domain colospgs.co.us
```

В более старых версиях клиента BIND (более ранних, чем BIND 4.8.3) *не допустимо* использование пробелов в конце строки, поскольку это приводит к установке локального доменного имени, заканчивающегося пробелами, а это, в большинстве случаев, эффект совершенно нежелательный. Существует и еще один способ установки локального доменного имени – посредством переменной окружения LOCALDOMAIN. Использование LOCALDOMAIN удобно в том смысле, что эта переменная может устанавливаться в разные значения для разных пользователей. Допустим, речь идет об огромном суперкомпьютере в корпоративном вычислительном центре и о служащих со всего мира, имеющих к этому компьютеру доступ. Каждый из служащих может в процессе выполнения работы пользоваться произвольным поддоменом, принадлежащим компании. С помощью LOCALDOMAIN каждый из пользователей может задать нужное локальное доменное имя в файлах настройки командного интерпретатора.

---

[1] На самом деле, существуют отдельные имена доменов, связанные с адресами, например *cc*.

Какой из трех методов следует использовать? Мы изначально предпочитаем автоматическое определение локального доменного имени на основе имени узла прежде всего потому, что так принято в Беркли и этот способ более правильный – в том смысле, что минимизирует дополнительные явные настройки. Кроме того, некоторые из программ Беркли, в особенности программы, использующие библиотечный вызов *ruserok( )* для идентификации пользователей, допускают использование кратких имен узлов в файлах типа *hosts.equiv* только в том случае, если полное доменное имя узла было определено посредством *hostname*.

Если же речь идет о программах, которые не могут работать с длинными именами узлов (*hostnames*), можно воспользоваться инструкцией *domain*. Команда *hostname* будет по-прежнему возвращать краткое имя, а DNS-клиент будет подставлять домен из файла *resolv.conf*. Использование же переменной LOCALDOMAIN может понадобиться для узла с большим числом пользователей.

Список поиска

Локальное доменное имя – производное от имени узла или заданное в файле *resolv.conf* – также определяет *список поиска* по умолчанию. Список поиска был придуман, чтобы немного облегчить жизнь пользователям путем сокращения объема набираемого текста. Идея состоит в том, чтобы производить поиск по вводимым в командной строке неполным именам (именам, не являющимся абсолютными) в одном или нескольких доменах.

Большинство сетевых команд Unix, принимающих доменное имя в качестве одного из аргументов (например, *telnet*, *ftp*, *rlogin*, *rsh*), используют для этого аргумента список поиска.

При переходе от BIND 4.8.3 к BIND 4.9 изменился как способ задания стандартного списка поиска, так и способ его применения. Если применяемый клиент – старого образца, то мы столкнемся с поведением версии 4.8.3, а если речь идет о более новой модели, включая BIND 8.2.3[1], то мы увидим изменения, которые появились в версии 4.9.

Независимо от версии BIND, пользователь может показать, что имя является абсолютным, добавив к нему точку.[2] Так, последняя точка в команде:

---

[1] Несмотря на добавление консорциумом ISC многочисленных новых функций к серверной части BIND 8, анализатор остался практически таким же, как в BIND 4.9.

[2] Как мы уже говорили, анализатор правильно интерпретирует точку в конце имени. Однако некоторые программы, в частности отдельные пользовательские почтовые программы Unix, некорректно работают с почтовыми адресами, которые заканчиваются точкой. Причем ошибки начинаются раньше, чем доменное имя из поля адреса добирается до анализатора.

```
% telnet ftp.ora.com.
```

означает «нет смысла искать в других доменах, это доменное имя является абсолютным». Аналогичным образом работает первый слэш в полных именах файловых систем Unix или MS-DOS. Если путевое имя начинается с символа слэша, оно интерпретируется как абсолютное и вычисляется от корня файловой системы, в противном случае оно является относительным (вычисляется от текущего каталога).

Список поиска BIND 4.8.3

В клиентах BIND версии 4.8.3 стандартный список поиска включает локальное доменное имя и доменные имена всех родительских доменов, состоящие не менее чем из двух меток. Таким образом, для узла с клиентом версии 4.8.3 и инструкцией:

```
domain cv.hp.com
```

стандартный список поиска будет содержать, во-первых, *cv.hp.com* – локальное доменное имя, а во-вторых, *hp.com* – имя родительского домена. Но не *com*, поскольку в этом имени только одна метка.[1] Производится поиск по указанному имени после поочередного добавления к имени элементов списка, и только в том случае, если имя содержит по меньшей мере одну точку. Поэтому команда:

```
% telnet pronto.cv.hp.com
```

приводит к поиску *pronto.cv.hp.com.cv.hp.com* и *pronto.cv.hp.com.hp.com*, и лишь затем собственно имени *pronto.cv.hp.com*. Команда:

```
% telnet asap
```

выполненная на том же узле, приводит к поиску клиентом имен *asap.cv.hp.com* и *asap.hp.com*, но не *asap*, поскольку это имя («asap») не содержит точек.

Заметим, что перебор имен из списка поиска прекращается, как только очередное доменное имя возвращает данные, которые являлись предметом поиска. В примере с именем *asap* никогда бы не произошел поиск с добавлением к имени элемента *hp.com*, если бы процесс разрешения имени *asap.cv.hp.com* закончился получением адреса.

---

[1] Одна из причин, по которой более старые анализаторы BIND не считали нужным добавлять только доменное имя домена высшего уровня, это то, что тогда – как, впрочем, и теперь – существовало крайне мало узлов во втором уровне пространства имен сети Интернет. То есть маловероятно, что добавление *com* или *edu* к имени *foo* приведет к получению имени реального узла. Кроме того, поиск адреса *foo.com* или *foo.edu* может потребовать выполнения запроса к корневому серверу имен, а это создает дополнительную нагрузку и отнимает драгоценное время.

Список поиска BIND 4.9 и более поздних версий

В клиентах BIND 4.9 и более поздних версий пакета стандартный список поиска включает только локальное доменное имя. Поэтому после использования инструкции:

```
domain cv.hp.com
```

стандартный список поиска будет содержать единственный элемент – *cv.hp.com*. Помимо этого, есть и еще одно отличие от предыдущих версий – элементы списка добавляются к имени *после* того, как для этого имени был произведен поиск. Если имя-аргумент содержит хотя бы одну точку, производится поиск для этого имени *перед* поиском с добавлением элементов списка. Поиск по списку производится только в том случае, если поиск для собственно имени не дал результатов. Даже в случае, когда имя-аргумент не содержит точек (то есть, является именем из одной метки), для него и тогда выполняется поиск, но уже после того, как будут перепробованы все элементы списка.

Почему лучше производить поиск по *буквальному* аргументу сначала? Разработчики DNS из опыта сделали вывод, что в большинстве случаев, если пользователь вводил имя хотя бы с одной точкой, это было абсолютное доменное имя без последней точки. При использовании более старого варианта поиска по списку клиент посылал несколько бесполезных запросов, прежде чем даже попробовать произвести поиск для указанного имени.

Поэтому если при использовании клиента версии 4.9 или более новой пользователь вводит:

```
% telnet pronto.cv.hp.com
```

прежде всего производится поиск по имени *pronto.cv.hp.com* (в аргументе целых три точки). Если поиск заканчивается безрезультатно, клиент производит поиск по имени *pronto.cv.hp.com.cv.hp.com*. Команда:

```
% telnet asap
```

выполненная на том же узле, приводит к поиску по имени *asap.cv.hp.com*, поскольку исходное имя не содержит точки, а затем просто по имени *asap*.

Инструкция search

Что делать, если стандартный список поиска нас не устраивает? В BIND версии 4.8.3 и более поздних версиях клиента список поиска может быть задан явным образом, перечислением доменных имен в предпочтительном порядке поиска. Задать список поиска позволяет инструкция *search*.

Синтаксис инструкции *search* весьма схож с синтаксисом инструкции *domain*, с той разницей, что можно указывать несколько доменных имен. Ключевое слово *search* должно начинаться в первой колонке строки, за ним может следовать пробел или символ табуляции. Далее может присутствовать от одного до шести доменных имен в порядке предпочтения.[1] Первое доменное имя в списке интерпретируется как локальное доменное имя, поэтому инструкции *search* и *domain* являются взаимоисключающими. Если использовать обе инструкции в файле *resolv.conf*, силу имеет та, что написана последней.

К примеру, инструкция:

```
search corp.hp.com paloalto.hp.com hp.com
```

является предписанием клиенту производить поиск сначала в домене *corp.hp.com*, затем в домене *paloalto.hp.com*, а затем в родительском домене *hp.com*.

Эта инструкция может быть полезной для узла, пользователи которого часто работают с узлами доменов *corp.hp.com* и *paloalto.hp.com*. С другой стороны, если используется клиент BIND версии 4.8.3, инструкция:

```
search corp.hp.com
```

является предписанием клиенту пропустить поиск в родительском домене локального доменного имени при использовании списка поиска. (В версиях клиента 4.9 и более поздних имя родительского домена обычно не входит в список поиска, так что такой вариант не отличается от стандартного.) Такая настройка полезна, если пользователи работают только с узлами локального домена, либо если существуют проблемы связи с DNS-серверами родительского домена (в этом случае минимизируется число запросов к родительским DNS-серверам).

В случае применения инструкции *domain* и последующего обновления клиента в версии 4.9 или более поздней, пользователи, которые полагались на тот факт, что родитель локального домена находится в списке поиска, могут подумать, что клиент внезапно «сломался». Прежнее поведение клиента можно восстановить с помощью инструкции *search* для настройки клиента на работу с тем же списком поиска, что и раньше. К примеру, для версий BIND 4.9, 8 и 9 можно заменить *domain nsr.hp.com* на *search nsr.hp.com hp.com* и получить ту же функциональность.

[1] DNS-клиенты BIND 9 поддерживают до восьми элементов в списке поиска.

Инструкция nameserver

В главе 4 мы рассказывали о двух типах DNS-серверов: первичных мастер-серверах и вторичных DNS-серверах. А что делать в случае, когда существует необходимость работать со службой DNS, но не устанавливать при этом DNS-сервер на каждый узел? Что делать в случае, когда *невозможно* установить DNS-сервер на узле (допустим, операционная система не позволяет этого сделать)? Вы же не обязаны держать DNS-сервер на *каждом* узле?

Разумеется, не обязаны. По умолчанию клиент ищет DNS-сервер, работающий на том же узле, и именно поэтому мы смогли воспользоваться инструментом *nslookup* на узлах *terminator.movie.edu* и *wormhole.movie.edu* сразу после настройки DNS-серверов. Однако существует возможность перенаправить клиент на другой узел в поисках службы имен.

Инструкция *nameserver* (да-да, пишется в одно слово) передает клиенту IP-адрес сервера, которому следует посылать запросы. К примеру, строка:

```
nameserver 15.32.17.2
```

является предписанием клиенту посылать запросы DNS-серверу, который работает на хосте с IP-адресом 15.32.17.2, а не DNS-серверу локального хоста. Это значит, что хосты, на которых не работают DNS-сервера, могут пользоваться инструкцией *nameserver* для указания клиенту удаленных DNS-серверов. Как правило, клиенты на хостах настраиваются так, чтобы они посылали запросы вашим собственным DNS-серверам.

Более старые, чем BIND 4.9, DNS-серверы не имеют понятия о разграничении доступа, а многие администраторы более новых серверов не налагают ограничений на поступающие запросы, поэтому можно настроить клиент на использование произвольного DNS-сервера. Разумеется, направление клиента на чужой DNS-сервер без предварительного разрешения – действие бесцеремонное, если не просто грубое, а работа с собственными серверами дает более высокую производительность, так что будем считать описанную возможность аварийной.

Помимо этого, существует возможность объяснить клиенту, что следует посылать запросы локальному DNS-серверу, используя либо IP-адрес локального узла, либо нулевой адрес. Нулевой адрес, 0.0.0.0, интерпретируется большинством реализаций TCP/IP в качестве адреса «этого узла». Разумеется, реальный IP-адрес узла также интерпретируется как локальный адрес. На узлах, которые не интерпретируют нулевой адрес таким образом, можно использовать адрес loopback-интерфейса – 127.0.0.1.

А что если DNS-сервер, которому посылает запросы клиент, не работает? Разве нет способа указать запасной сервер? Неужели придется вернуться к использованию таблицы узлов?

Клиент позволяет указать до трех (посчитайте-ка до трех) DNS-серверов с помощью нескольких инструкций *nameserver*. Клиент опрашивает эти DNS-серверы в порядке их перечисления, пока не будет получен ответ от одного из них, или не истечет интервал ожидания. К примеру, строки:

```
nameserver 15.32.17.2
nameserver 15.32.17.4
```

являются инструкций клиенту сначала послать запрос DNS-серверу по адресу 15.32.17.2, а в случае отсутствия ответа – DNS-серверу по адресу 15.32.17.4. Следует помнить, что число перечисленных DNS-серверов влияет и на другие аспекты поведения DNS-клиента.

При использовании нескольких инструкций *nameserver* **ни в коем случае** не применяйте адрес loopback-интерфейса! В некоторых реализациях TCP/IP, основанных на реализации Беркли, существует ошибка, которая вызывает сбои в работе BIND в случаях, когда локальный DNS-сервер не работает. Используемый клиентом сокет дейтаграммы не переключается на новый локальный адрес, если локальный DNS-сервер не запущен, и как следствие клиент посылает пакеты с запросами резервным удаленным DNS-серверам с адресом отправителя 127.0.0.1. Когда удаленный DNS-сервер пытается ответить, то посылает пакеты с ответами самому себе.

В списке один DNS–сервер

В случае с единственным DNS-сервером[1] клиент посылает запросы этому серверу с интервалом ожидания в пять секунд. Интервал ожидания определяет время, в течение которого клиент будет ожидать ответа от DNS-сервера, прежде чем послать еще один запрос. Если клиент сталкивается с ошибкой, это значит, что DNS-сервер недоступен или не работает; если истекает интервал ожидания, этот интервал удваивается, а запрос повторяется. Перечислим ошибки, которые могут приводить к такой ситуации:

- Получение ICMP-сообщения о *недоступности порта* (*port unreachable*), которое означает, что никакой DNS-сервер не принимает запросы через порт DNS-сервера

[1] Когда мы говорим «единственный сервер имен», то имеем в виду либо наличие всего одной инструкции *nameserver* в файле *resolv.conf*, либо полное отсутствие инструкций *nameserver* – для случая использования локального сервера имен.

- Получение ICMP-сообщения о *недоступности узла* (*host unreachable*) или о *недоступности сети* (*network unreachable*), которые означают, что запросы не могут быть отправлены по указанному IP-адресу

Если доменное имя или запрашиваемые данные не существуют, клиент не повторяет запрос. Все DNS-серверы, по крайней мере теоретически, обладают одинаковым «видом» пространства имен, и нет причин одному из них верить, а другому нет. Поэтому если один из DNS-серверов сообщает, что указанное доменное имя не существует или тип искомых данных не существует для указанного доменного имени, то любой другой DNS-сервер должен возвращать точно такой же ответ.[1] Если клиент получает сетевую ошибку каждый раз при посылке запроса (не более четырех раз[2]), то он возвращается к использованию таблицы узлов. Обратите внимание, что речь идет именно об *ошибках*, а не истечении интервала ожидания. Если истекает интервал ожидания хотя бы для одного запроса, клиент возвращает пустой ответ и не переходит к использованию файла */etc/hosts*.

В списке несколько DNS-серверов

Если DNS-серверов несколько, поведение клиента несколько изменяется. Происходит следующее: клиент начинает с посылки запроса первому DNS-серверу из списка, с интервалом ожидания в пять секунд, как и в случае с единственным DNS-сервером. Если время ожидания истекает либо получена сетевая ошибка, клиент посылает запрос следующему DNS-серверу и ожидает ответ те же пять секунд. К сожалению, клиент не получает многих из возможных ошибок; сокет, используемый клиентом, «не подключен» (unconnected), поскольку он должен принимать ответы от опрашиваемых DNS-серверов, а неподключенные сокеты не могут принимать ICMP-сообщения об ошибках. Если клиент опросил все перечисленные DNS-серверы и не получил ответов, интервал ожидания изменяется, а цикл опроса повторяется с начала.

В следующей серии запросов интервал ожидания клиентом ответа основывается на количестве DNS-серверов, указанных в файле *resolv.conf*. Интервал ожидания для второй серии запросов – 10 секунд,

---

[1] Существующая в DNS задержка может служить причиной маленького расхождения: первичный мастер-сервер имен может быть авторитативным для зоны и предоставлять сведения, отличающиеся от сведений вторичного узла, который также является авторитативным для этой зоны. Мастер-сервер имен только что загрузил новые данные зоны из файлов, а вторичный сервер еще не успел получить новую зону от первичного. Оба сервера возвращают авторитативные ответы для этой зоны, но первичный мастер может знать о только что появившемся узле, о котором еще ничего не известно вторичному серверу.

[2] Не более двух раз для анализаторов BIND 8.2.1 и более поздних версий.

которые делятся на число DNS-серверов с отбрасыванием дробной части результата. Для каждой последующей серии интервал ожидания удваивается. После трех наборов переключений (для каждого сервера из списка истекли четыре интервала ожидания), клиент прекращает попытки получить от DNS-серверов ответ.

В BIND версии 8.2.1 консорциум ISC внес в DNS-клиент изменения, сократив число повторных серий до одной, то есть до двух попыток для каждого из DNS-серверов, перечисленных в файле *resolv.conf*. Это должно было сократить время ожидания для случаев, когда ни один из DNS-серверов не отвечает.

Специально для тех, кто не любит арифметику, мы приводим табл. 6.1, в которой рассчитаны интервалы ожидания для случаев одного, двух или трех DNS-серверов.

Таблица 6.1. Интервалы ожидания в BIND версий с 4.9 по 8.2

Повторы	**Количество DNS-серверов**		
	1	**2**	**3**
0	5 с	(дважды) 5 с	(трижды) 5 с
1	10 с	(дважды) 5 с	(трижды) 3 с
2	20 с	(дважды) 10 с	(трижды) 6 с
3	40 с	(дважды) 20 с	(трижды) 13 с
Всего	75 с	80 с	81 с

Поведение по умолчанию для клиента BIND 8.2 и более поздних версий показано в табл. 6.2.

Таблица 6.2. Интервалы ожидания в BIND версий с 8.2.1

Повторы	**Количество DNS-серверов**		
	1	**2**	**3**
0	5 с	(дважды) 5 с	(трижды) 5 с
1	10 с	(дважды) 5 с	(трижды) 3 с
Всего	15 с	20 с	24 с

Итак, если мы перечислили три DNS-сервера, клиент посылает запрос первому из них с интервалом ожидания в пять секунд. Если интервал ожидания истекает, клиент посылает запрос второму серверу с таким же интервалом ожидания, и точно так же операция повторяется с третьим сервером. Если клиент перебрал все три DNS-сервера, он удваивает интервал ожидания и делит его на три (10 поделить на три с отбрасыванием дробной части – получается три секунды) и снова посылает запрос первому серверу.

Пугают показатели времени? Давайте вспомним, что речь идет о худшем варианте развития событий. При нормально работающих серверах имен, расположенных на достаточно быстрых узлах, клиенты будут получать ответы в пределах одной секунды. Лишь в случае, когда все перечисленные DNS-серверы сильно загружены либо существуют проблемы доступа к ним, клиент будет вынужден пройти через все переключения и сдаться, не получив ответа.

Что делает клиент, потеряв всякие надежды получить ответ? Возвращает ошибку. Обычно в таком случае отображается примерно следующее сообщение:

```
% telnet tootsie
tootsie: Host name lookup failure
```

Разумеется, ожидание этого сообщения может занять порядка 75 секунд, так что наберитесь терпения.

Инструкция sortlist

Инструкция *sortlist* в версиях BIND 4.9 и более поздних позволяет указывать подсети и сети, которым должен отдавать предпочтение клиент, получая в ответе на запрос несколько адресов. В некоторых случаях существует необходимость работать с конкретными узлами через определенную сеть. Возьмем для примера рабочую станцию и NFS-сервер; обе машины имеют по два сетевых интерфейса: один Ethernet-интерфейс в подсети 128.32.1/24; один в кольце FDDI, в подсети 128.32.42/24. Если предоставить DNS-клиент на рабочей станции самому себе, можно только гадать, какой из IP-адресов NFS-сервера будет использован при монтировании файловой системы (предположительно, первый из перечисленных в пакете с ответом сервера). Чтобы знать наверняка, что первым будет использован адрес интерфейса в кольце FDDI, можно добавить в файл *resolv.conf* инструкцию *sortlist*, которая сортирует адреса подсети 128.32.42/24 в нужном порядке, что принимается во внимание программами, работающими с клиентом:

```
sortlist 128.32.42.0/255.255.255.0
```

После символа «слэш» следует маска описанной подсети. Если есть необходимость отдать предпочтение целой сети, можно опустить слэш и маску подсети:

```
sortlist 128.32.0.0
```

Клиент будет считать, что речь идет о целой сети 128.32/16. (Клиент генерирует стандартную маску полной сети на основе первых двух битов IP-адреса.)

Разумеется, можно указать несколько (до десяти) подсетей или сетей, которые более предпочтительны:

```
sortlist 128.32.42.0/255.255.255.0 15.0.0.0
```

DNS-клиент располагает адреса из ответа в порядке перечисления их сетей и подсетей в инструкции *sortlist*, а все прочие адреса добавляет в конец списка.

Инструкция options

Инструкция *options* появилась в BIND версии 4.9, она позволяет менять внутренние настройки DNS-клиента. Первая такая настройка – это флаг отладки RES_DEBUG. Инструкция:

```
options debug
```

устанавливает флаг RES_DEBUG, что приводит к печати огромного объема отладочной информации на стандартный вывод, разумеется, если клиент был собран с ключом DEBUG. (Не стоит на это полагаться, поскольку большинство поставщиков систем поставляют клиенты, собранные без этого ключа.) Такая возможность весьма полезна, если необходимо продиагностировать проблему, связанную с клиентом, либо службой имен в целом, но во всех остальных случаях она просто мешает жить.

Второй доступный параметр – значение *ndots*, определяющее минимальное число точек в доменном имени-аргументе, при котором поиск по этому имени осуществляется до применения списка поиска. По умолчанию это происходит для имен, которые содержат одну или более точек, что эквивалентно настройке *ndots:1*. Если в имени есть хотя бы одна точка, клиент попытается найти введенное имя. Если можно предполагать, что пользователи будут часто вводить частичные доменные имена, для которых требуется применение списка поиска, порог можно приподнять. К примеру, у нас есть локальное доменное имя *mit.edu*, и пользователи привыкли набирать:

```
% ftp prep.ai
```

с автоматическим добавлением *mit.edu* и результатом *prep.ai.mit.edu*, можно увеличить значение *ndots* до двух, чтобы пользователи не дергали попусту корневые DNS-серверы в поиске имен в домене высшего уровня *ai*. Этого можно добиться командой:

```
options ndots:2
```

В BIND версии 8.2 появились четыре новых параметра настройки клиента: *attempts*, *timeout*, *rotate* и *no-check-names*. *attempts* позволяет определить число запросов, посылаемых клиентом каждому из DNS-серверов, перечисленных в файле *resolv.conf*, перед «капитуляцией». Если вам кажется, что новое значение по умолчанию – два раза – слишком мало для ваших DNS-серверов, смените его обратно на старое значение, которое было стандартным до версии 8.2.1:

```
options attempts:4
```

Максимальное допустимое значение – пять.

timeout позволяет задать начальный интервал ожидания ответа на запрос. Значение по умолчанию – пять секунд. Если существует необходимость ускорить переключение, можно уменьшить значение до двух секунд:

```
options timeout:2
```

Максимальное допустимое значение – 30 секунд. Для второй и последующих серий запросов начальный интервал ожидания удваивается и делится на число DNS-серверов, указанных в файле *resolv.conf*.

rotate позволяет предписать клиенту использовать все DNS-серверы, перечисленные в файле *resolv.conf*, а не только первый. Если первый DNS-сервер из списка работоспособен, он будет использоваться для разрешения всех запросов клиента. До тех пор пока этот DNS-сервер не окажется перегруженным либо не перестанет работать, второй или третий DNS-серверы использоваться не будут. Распределить нагрузку между серверами можно с помощью следующей инструкции:

```
options rotate
```

При каждом новом запросе порядок использования указанных DNS-серверов будет изменяться. Иными словами, клиент по-прежнему будет начинать работу с первого DNS-сервера, но для следующего доменного имени будет использован второй сервер в качестве первого и т. д.

Следует помнить, что многие программы при работе не используют этот механизм, поскольку в большинстве случаев происходит инициализация клиента, поиск данных для имени и завершение работы. Так, перестановка серверов не влияет на повторяемые команды *ping*, поскольку каждый из процессов программы *ping* инициализирует клиент, посылает запрос первому из серверов, перечисленных в *resolv.conf*, и затем завершается, прежде чем клиент будет вызван еще раз. Каждый новый экземпляр программы *ping* понятия не имеет о том, какой DNS-сервер работал с предыдущим экземпляром. Но процессы с большим временем жизни, которые посылают много запросов, например демон *sendmail*, без труда пользуются преимуществами перестановки серверов.

Перестановка также затрудняет отладку. С ней никогда нельзя знать наверняка, какому из DNS-серверов демон *sendmail* послал запрос, получив впоследствии неправильный ответ.

И, наконец, параметр *no-check-names* позволяет отключать проверку клиентом доменных имен, которая по умолчанию включена.[1] Клиент проверяет имена, полученные в ответах, на соответствие интернет-

[1] Во всех DNS-клиентах, поддерживающих проверку имен, то есть начиная с BIND версии 4.9.4.

стандартам именования узлов: в именах допустимы только буквы, цифры и дефисы. Этот параметр необходимо установить, если у пользователей существует потребность в разрешении доменных имен, включающих подчеркивания или другие недопустимые символы.

Можно установить значения сразу нескольких параметров, скомбинировав их в одной строке файла *resolv.conf* следующим образом:

```
options attempts:4 timeout:2 ndots:2
```

Комментарии

В клиенте BIND начиная с версии 4.9 (самое время, как мы считаем) появилась возможность включать комментарии в файл *resolv.conf*. Строки, начинающиеся с символа решетки или точки с запятой в первой позиции, интерпретируются как комментарии и игнорируются клиентом.

Замечание по поводу инструкции клиента версии 4.9

Если вы только переходите к использованию клиента BIND 4.9, будьте осторожны при работе с новыми инструкциями. Код более старого клиента может входить в состав программ, существующих на узле. Большую часть времени это не представляет проблемы, поскольку клиенты в Unix-системах игнорируют инструкции, которых не понимают. Но при этом не следует ожидать, что все программы на узле поймут смысл новых инструкций.

Если речь идет об узле, программы которого работают с очень старым кодом клиента (из версии более ранней, чем 4.8.3), и существует необходимость использовать инструкцию *search* для программ, которые ее понимают, следует поступить следующим образом: использовать в файле *resolv.conf* и инструкцию *domain* и инструкцию *search*, причем инструкция *domain* должна предшествовать инструкции *search*. Старые клиенты будут выполнять инструкцию *domain*, игнорируя *search*, поскольку не распознают эту инструкцию. Новые клиенты будут выполнять инструкцию *domain*, но инструкция *search* будет иметь более высокий приоритет.

Примеры настройки DNS-клиента

С теорией покончено, перейдем к содержанию файлов *resolv.conf*, существующих на реальных хостах. Настройка клиента зависит от наличия локального DNS-сервера, поэтому мы рассмотрим оба случая – настройку для локальных DNS-серверов и для удаленных.

Собственно клиент

Нас, администраторов *movie.edu*, попросили настроить рабочую станцию одного преподавателя, на которой отсутствует DNS-сервер. Решить, к какому домену принадлежит станция, очень просто, поскольку единственный выбор – *movie.edu*. При этом преподаватель работает с исследователями компании Pixar – над новым алгоритмом расчета освещенности, поэтому, видимо, имеет смысл добавить *pixar.com* в список поиска на этой рабочей станции. Следующая инструкция *search*:

```
search movie.edu pixar.com
```

предписывает использовать *movie.edu* в качестве локального доменного имени рабочей станции и производить в пределах *pixar.com* поиск имен, не найденных в *movie.edu*.

Новая рабочая станция находится в сети 192.249.249/24, ближайшие DNS-серверы – *wormhole.movie.edu* (192.249.249.1) и *terminator.movie.edu* (192.249.249.3). Обычно следует настраивать узлы на использование ближайшего из доступных DNS-серверов. (Ближе всего DNS-сервер расположен, если работает на том же узле, следующий по удаленности – DNS-сервер в той же подсети или сети.) В данном случае оба сервера расположены одинаково близко, но нам известно, что узел *wormhole.movie.edu* более мощный в плане производительности. Поэтому первая инструкция *nameserver* в файле *resolv.conf*:

```
nameserver 192.249.249.1
```

Про этого преподавателя известно, что он начинает ужасно кричать, если возникают проблемы с компьютером, поэтому мы добавим *terminator.movie.edu* (192.249.249.3) в качестве резервного DNS-сервера. В этом случае, если по какой-либо причине не будет работать *wormhole.movie.edu*, рабочая станция нашего преподавателя по-прежнему сможет использовать службу имен (разумеется, если действует *terminator.movie.edu* и сеть в целом).

В итоге, файл *resolv.conf* выглядит следующим образом:

```
search movie.edu pixar.com
nameserver 192.249.249.1
nameserver 192.249.249.3
```

Локальный DNS-сервер

Теперь необходимо настроить почтовый концентратор Университета, *postmanrings2x.movie.edu*, на работу со службой доменных имен. *postmanrings2x.movie.edu* используется всеми группами *movie.edu*. Недавно мы настроили DNS-сервер на этом узле, чтобы сократить нагрузку на другие узлы, поэтому следует убедиться, что клиент посылает запросы в первую очередь DNS-серверу локального узла.

Самый простой вариант настройки DNS-клиента в этом случае – полное отсутствие настройки: просто не будем создавать файл *resolv.conf* и позволим клиенту автоматически использовать локальный DNS-сервер. Имя узла (*hostname*) следует установить в полное доменное имя узла, чтобы клиент мог определить локальное доменное имя.

Если мы предусмотрительно решим, что нужен резервный DNS-сервер, то можем создать *resolv.conf* для произведения настройки. Необходимость в настройке для резервного сервера зависит, в основном, от надежности локального DNS-сервера. Качественные реализации DNS-сервера BIND способны работать дольше, чем некоторые операционные системы, а значит, можно обойтись без резервного сервера. Если же в послужном списке локального сервера встречаются проблемы – скажем, неожиданные сбои или прекращение корректной работы, добавление резервного DNS-сервера может себя оправдать.

Чтобы добавить резервный DNS-сервер, следует указать локальный DNS-сервер первым в файле *resolv.conf* (IP-адрес локального узла или нулевой адрес 0.0.0.0 – на усмотрение администратора), затем один или два дополнительных. Помните, что не следует пользоваться адресом loopback-интерфейса, если нет уверенности, что TCP/IP-стек системы не страдает от ошибки, которую мы описывали ранее.

Поскольку лучше перестраховаться, чем потом пожинать плоды непрофессионализма, мы добавим два резервных сервера. *postman-rings2x.movie.edu* также находится в сети 192.249.249/24, поэтому *terminator.movie.edu* и *wormhole.movie.edu* – наиболее близко расположенные к нему DNS-серверы (если не считать локального). Не забывая о необходимости распределения нагрузки, мы изменим порядок опроса этих серверов относительно предыдущего примера, в котором мы настраивали только DNS-клиент. И поскольку нам не хочется ждать полных пять секунд, пока клиент перейдет к работе со вторым DNS-сервером, мы сократим интервал ожидания до двух секунд. Вот так, в итоге, выглядит файл *resolv.conf*:

```
domain movie.edu
nameserver 0.0.0.0
nameserver 192.249.249.3
nameserver 192.249.249.1
options timeout:2
```

Как упростить себе жизнь

Итак, произведя настройку узла на использование DNS, спросим себя, что изменилось. Придется ли пользователям набирать длинные доменные имена? Придется ли им изменять свои почтовые адреса и адреса списков рассылки?

Благодаря существованию списка поиска, многие вещи будут продолжать работать как раньше. Однако есть исключения и существенные отличия в поведении программ, которые используют DNS. Мы постараемся рассмотреть самые распространенные случаи.

Различия в поведении служб

Как мы уже знаем, приложения вроде *telnet*, *ftp*, *rlogin* и *rsh* применяют список поиска для работы с именами, которые не заканчиваются точкой. Это значит, что если мы находимся в *movie.edu* (то есть если *movie.edu* является локальным доменным именем, а список поиска содержит *movie.edu*), то можем набрать:

```
% telnet misery
```

или:

```
% telnet misery.movie.edu
```

или даже:

```
% telnet misery.movie.edu.
```

и получить одинаковые результаты. То же правило действует и для других служб. Существует одно отличие, которое может оказаться полезным: поскольку DNS-сервер может возвращать несколько IP-адресов при поиске адресных записей, современные версии Telnet, FTP и веб-броузеров пытаются связаться с первым из полученных адресов, а в случае, если соединение не может быть установлено по какой-либо причине, пробуют следующий адрес, и так далее:

```
% ftp tootsie
ftp: connect to address 192.249.249.244: Connection timed out
Trying 192.253.253.244...
Connected to tootsie.movie.edu.
220 tootsie.movie.edu FTP server (Version 16.2 Fri Apr 26
     18:20:43 GMT 1991) ready.
Name (tootsie: guest):
```

Помните, что с помощью инструкции *sortlist* в файле *resolv.conf* можно контролировать порядок перебора приложениями полученных адресов.

Нетипичной в этом смысле является служба NFS. Команда *mount* замечательно работает с доменными именами, и доменные имена можно использовать в файле */etc/fstab* (в отдельных системах – */etc/checklist*). Но в том, что касается файлов */etc/exports* и */etc/netgroup*, следует проявлять осторожность. */etc/exports* определяет, какие файловые системы доступны для NFS-монтирования различным NFS-клиентам. В файле *netgroup* можно определить имя для группы узлов, а затем использовать его в файле *exports* для распределения групповых полномочий.

К сожалению, более старые версии NFS не используют DNS в целях проверки *exports* или *netgroup* – сервер NFS получает информацию о клиенте в виде пакета RPC (Remote Procedure Call). Как следствие, клиент идентифицируется теми данными, которые предоставил, а в качестве идентификатора узла в Sun RPC используется его имя (*hostname*). Поэтому имя, используемое в любом из файлов, должно совпадать с именем узла-клиента, а это имя далеко не всегда совпадает с доменным.

Электронная почта

Некоторые из программ, связанных с электронной почтой (к примеру, *sendmail*), также работают не так, как ожидалось. *sendmail* использует список поиска не так, как другие программы. Будучи настроенной на использование DNS-сервера, *sendmail* применяет операцию, называемую *канонизацией*, для преобразования имен в адресах электронной почты в полные, канонические доменные имена.

В процессе канонизации *sendmail* производит сочетание элементов списка поиска с именем и поиск данных типа ANY, то есть записей любого типа. *sendmail* использует то же правило, что и более новые DNS-клиенты: если канонизируемое имя содержит хотя бы одну точку, то оно, прежде всего, проверяется буквально. Если DNS-сервер, которому был направлен запрос, находит CNAME-запись (псевдоним), *sendmail* заменяет имя каноническим, на которое ссылается псевдоним, а затем производит канонизацию *полученного* имени (на случай, если правая часть записи псевдонима также является псевдонимом). Если DNS-сервер находит адресную запись, *sendmail* использует доменное имя, разрешенное в адрес, в качестве канонического. Если DNS-сервер не нашел адресных записей, но нашел одну или несколько MX-записей, выполняется одно из следующих действий:

- Если список поиска еще не использовался, *sendmail* использует доменное имя, для которого найдены MX-записи, в качестве канонического.
- Если результаты были получены в процессе использования одного из элементов списка поиска, *sendmail* отмечает, что доменное имя потенциально является каноническим, и продолжает перебор элементов списка. Если в дальнейшем для одного из комбинированных имен найдена адресная запись, именно это доменное имя принимается за каноническое. В противном случае, в качестве канонического используется доменное имя, для которого раньше всего были найдены MX-записи.[1]

[1] Все эти сложности необходимы для работы с MX-записями, определяемыми с помощью маски, о них речь пойдет в главе 16 «Обо всем понемногу».

При обработке SMTP-сообщения *sendmail* применяет канонизацию многократно – для адреса получателя и нескольких полей заголовка SMTP.[1]

Помимо этого, *sendmail* присваивает макросу *$w* канонизированное имя *hostname* при запуске демона *sendmail*. Так что даже при использовании короткого имени узла, состоящего из одной метки, *sendmail* производит канонизацию с помощью списка поиска, определенного в файле *resolv.conf*. Затем *sendmail* добавляет макрос *$w* и все псевдонимы для *$w*, найденные в процессе канонизации, к классу *$=w*, то есть списку прочих имен почтового сервера.

И это важно, потому что только имена из класса *$=w* по умолчанию опознаются программой *sendmail* в качестве имен локального узла. *sendmail* будет пытаться передать почту, адресованную доменному имени, которое не является локальным, почтовому ретранслятору. Поэтому если не настроить программу *sendmail* таким образом, чтобы она распознавала все псевдонимы узла (путем добавления их к классу *w* или файловому классу *w*, как мы показывали в главе 5 «DNS и электронная почта»), узел будет пытаться передать дальше сообщения, которые адресованы любому другому имени, кроме канонического.

Существует еще одна особенность класса *$=w*, она заключается в том, что в MX-записях *sendmail* опознает в качестве имени локального узла только имена, входящие в класс *$=w*. Как следствие, если в правой части MX-записи использовать имя, про которое не известно точно, что оно входит в класс *$=w*, существует риск, что узел не опознает имя в качестве своего. Это может приводить к созданию петли маршрутизации и последующей отправке сообщений их автору.

И еще одно замечание по программе *sendmail*: если DNS-сервер начинает использоваться совместно со старой версией *sendmail* (до версии 8), следует установить параметр *I* в файле *sendmail.cf*. Параметр *I* определяет действия *sendmail* в случае отрицательных результатов поиска данных для узла-адресата. При использовании */etc/hosts* отрицательные результаты поиска являются неисправимой ошибкой. Если имя не было найдено в таблице узлов, маловероятно, что позже оно там появится каким-то волшебным образом, поэтому почтовая программа может просто вернуть сообщение отправителю. Однако при использовании DNS, отрицательный результат может быть временным явлением, вызванным, к примеру, спорадическими сетевыми проблемами. Установка параметра *I* предписывает программе *sendmail* поместить почтовые сообщения в очередь, но не возвращать их отправителю. Для

[1] Некоторые более старые версии *sendmail* применяют другой алгоритм канонизации: список поиска используется при создании запросов к серверам имен – на предмет получения CNAME-записей для исходного имени. При поиске по типу CNAME возвращаются только CNAME-записи. Если такая запись найдена, исходное имя заменяется именем из правой части записи.

установки параметра *I* просто добавьте *OI* к содержимому файла *sendmail.cf*.

Обновление файлов .rhosts, hosts.equiv и других

При использовании DNS вы можете столкнуться с необходимостью избавиться от неоднозначностей, связанных с именами узлов в файлах авторизации. Строки, в которых указаны простые имена узлов из одной метки, будут считаться принадлежащими локальному домену. К примеру, файл *lpd.allow* на узле *wormhole.movie.edu* может содержать строки:

```
wormhole
terminator
diehard
robocop
mash
twins
```

Но если мы перенесем *mash* и *twins* в зону *comedy.movie.edu*, у этих узлов уже не будет доступа к службе *lpd*; записи в *lpd.allow* разрешают доступ только узлам *mash.movie.edu* и *twins.movie.edu*. Поэтому придется добавить соответствующие доменные имена, которые не принадлежат локальному домену сервера *lpd*:

```
wormhole
terminator
diehard
robocop
mash.comedy.movie.edu
twins.comedy.movie.edu
```

Несколько других файлов, которые следует проверить на предмет корректировки имен узлов:

```
hosts.equiv
.rhosts
X0.hosts
sendmail.cf
```

Иногда для исключения двусмысленностей бывает достаточно прогнать эти файлы через фильтр канонизации – программу, которая преобразует имена узлов в доменные имена, применяя список поиска. Вот очень короткий фильтр канонизации на языке Perl, который отлично справится с задачей:

```
#!/usr/bin/perl -ap
# Ожидается одно имя узла на строку - в первом поле (а-ля .rhosts,
# X0.hosts)

s/$F[0]/$d/ if ($d)=gethostbyname $F[0];
```

Создание псевдонимов

Позаботившись обо всем и преобразовав все свои файлы *.rhosts*, *hosts.equiv* и *sendmail.cf* после настройки узла на работу с DNS, мы должны подумать и о том, что пользователям нужно привыкнуть к использованию доменных имен. Хотелось бы, чтобы процесс проистекал с минимальными неудобствами и был более чем компенсирован преимуществами DNS.

Один из способов облегчить жизнь пользователей после настройки DNS – создать псевдонимы для известных узлов, которые более недоступны по существовавшим ранее именам. К примеру, наши пользователи привыкли к тому, что можно набрать *telnet doofy* или *rlogin doofy* и получить доступ к системе электронных объявлений, которая работает в киностудии на другом конце города. Теперь им придется набирать полное доменное имя узла *doofy* – *doofy.maroon.com*. Но большинству пользователей полное доменное имя неизвестно, и пройдет какое-то время, прежде чем все будут в курсе и привыкнут к нововведениям.

К счастью, BIND позволяет создавать для нужд пользователей псевдонимы. Достаточно просто установить значение переменной среды HOSTALIASES в полное имя файла, которые содержит отображение псевдонимов в доменные имена. К примеру, чтобы создать общий псевдоним для *doofy*, мы можем присвоить переменной HOSTALIASES значение */etc/host.aliases* (в одном из загрузочных файлов системы), и добавить в этот файл строку:

```
doofy    doofy.maroon.com
```

Формат файла псевдонимов очень простой: псевдоним начинается с первой позиции строки, за ним следуют пробелы и доменное имя, соответствующее псевдониму. Доменное имя записывается без точки в конце, а псевдоним не должен содержать точек вообще.

Теперь, если наши пользователи набирают *telnet doofy* или *rlogin doofy*, DNS-клиент прозрачным образом подставляет вместо *doofy* – *doofy.maroon.com* в запросе к DNS-серверу. Пользователи видят примерно следующий вывод:

```
Trying...
Connected to doofy.maroon.com.
Escape character is '^]'.
IRIX System V.3 (sgi)
login:
```

Однако если клиент возвращается к использованию файла */etc/hosts*, переменная HOSTALIASES перестает иметь значение. Поэтому мы включаем аналогичный псевдоним в файл */etc/hosts*.

Со временем и, вероятно, после нескольких лекций, пользователи начнут ассоциировать полное доменное имя, которое видят в сообщениях программы *telnet* с используемой системой электронных объявлений.

Если известны имена, с которыми могут испытывать сложности пользователи, можно уменьшить их переживания с помощью HOSTALIASES. Если неизвестно заранее, с какими узлами работают пользователи, можно разрешить им создавать собственные файлы псевдонимов. В этом варианте пользователь должен устанавливать значение переменной HOSTALIASES в загрузочных файлах командного интерпретатора.

Специфика настройки различных систем

Существует мнение, что Unix – стандартная операционная система, но существует почти столько же стандартов Unix, сколько и вариантов системы. Поэтому стилей настройки DNS-клиента существует практически столько же, сколько существует разновидностей Unix. Практически все реализации клиентов поддерживают изначальный синтаксис Беркли, но во многих их них присутствуют дополнения или вариации. Здесь мы постарались как можно более полно рассмотреть основные стили настройки DNS-клиента в различных системах.

SunOS 4.x от Sun

Настройка узла, работающего под управлением SunOS, может стать настоящим испытанием. Можно сказать, что поведение DNS-клиента в SunOS наиболее сильно отличается от стандартного BIND, если говорить о крупных поставщиках Unix-систем. Это происходит, прежде всего потому, что DNS-клиент в SunOS интегрирован с сетевой информационной службой Sun (Network Information Service, или NIS, в девичестве Yellow Pages).

В двух словах, NIS реализует механизм синхронизации важных файлов на различных узлах сети. Речь идет не только о */etc/hosts*, но также о */etc/services*, */etc/passwd* и других файлах. Sun позиционирует DNS в качестве резервного варианта, используемого с NIS; в случае, когда клиент NIS не может найти имя узла (или IP-адрес) в карте NIS (*hosts*), существует возможность настроить его на использование DNS-сервера.

Обратите внимание, что функциональность DNS-клиента реализована в составе программы *ypserv*, которая работает также и с другими типами NIS-запросов. Так что, если не запущена программа *ypserv*, не запущен и клиент! (К счастью, клиент в системе Solaris 2 можно использовать и не выполняя *ypserv*.) Преимущество *ypserv* для разрешения всех запросов заключается в том, что нет необходимости настраивать DNS-клиенты на клиентах NIS, достаточно сделать это на серверах

NIS.[1] Клиенты NIS запрашивают у NIS-сервера данные для узла, а сервер NIS, при необходимости, запрашивает эти данные у DNS.

Если установлена SunOS 4.x (Solaris 1), можно: (1) следовать генеральной линии партии и настроить клиент на использование DNS в качестве резервного варианта для NIS, (2) работать с NIS без карты *hosts* либо (3) попрать обычаи и перекомпилировать клиент в целях использования исключительно DNS; свободно доступные копии измененных клиентов распространяются по сети Интернет. Однако мы должны предупредить, что, по нашим сведениям, Sun *не будет* обеспечивать поддержку измененных клиентов.

Если используется Solaris 2, можно просто по-человечески настроить DNS-клиент и, с помощью файла *nsswitch.conf*, указать, что для разрешения имен следует применять DNS.

Измененные клиенты

Мы не будем подробно останавливаться на этом варианте, прежде всего потому, что он достаточно хорошо документирован и практически автоматизирован в дистрибутивах BIND 4. Собственно процесс заключается в создании новой стандартной разделяемой библиотеки *libc.so*, путем удаления функций, использующих NIS, и заменой их DNS-вариантами. Хотя Sun великодушно предоставляет все необходимые функции для замены, поддержки этих функций не существует. Что еще хуже, функции, поставляемые в составе SunOS 4.x, основаны на коде из BIND 4.8.1.

Дистрибутивы исходных текстов BIND 4 содержали инструкции по установке функций клиента для системы SunOS 4.x – в каталоге *shres/sunos* (файл назывался *INSTALL*). Что касается BIND 8, эти инструкции не работают (а в BIND 8.2.2 они вообще не включены). Старые дистрибутивы исходных текстов BIND по-прежнему можно получить путем анонимного FTP-доступа к узлу *ftp.isc.org*, каталог называется */isc/bind/src*. Если вы намереваетесь собрать замену клиента SunOS 4.x из исходных текстов, мы рекомендуем использовать исходные тексты BIND 4.9.7, доступные по адресу *ftp.isc.org/isc/bind/src/4.9.7/bind-4.9.7-REL.tar.gz*.

Если вы предпочитаете обойтись без потенциально поучительного опыта создания собственной разделяемой библиотеки на языке C и воспользоваться результатами чужого труда, то можете взглянуть на библиотеку *resolv+*, которая разработана на основе клиента BIND 4.8.3. *resolv+* является усовершенствованной версией функций клиента версии 4.8.3 для SunOS. Библиотека была доработана Биллом Уизнером

[1] В действительности необходимо также настроить DNS-клиент на узлах, использующих *sendmail.mx*, усовершенствованную в плане работы с MX-записями версию *sendmail* от Sun.

(Bill Wisner), она позволяет администраторам выбирать, в каком порядке производятся запросы к NIS и DNS (аналогично расширениям, которые были добавлены к различным Unix-системам другими компаниями и о которых мы поговорим чуть позже). Новые функции и инструкции по сборке их в файл *libc.so* доступны для копирования с сервера *ftp.uu.net*, файл называется */networking/ip/dns/resolv+2.1.1.tar.Z*. Более подробно функциональность библиотеки *resolv+* описана далее в этой главе, в разделе, посвященном системам Linux.

Совместное использование DNS и NIS

Однако если речь идет о социально приемлемом варианте, то NIS и DNS должны мирно сосуществовать. Это несколько нетривиальная задача, поэтому мы рассмотрим ее чуть более подробно. Установку NIS мы описывать не будем, эти кровавые подробности можно найти в книге Хэла Стерна (Hal Stern) «Работа с NFS и NIS» (Managing NFS and NIS, O'Reilly). Помните, что приводимые инструкции действительны только для версий SunOS более поздних, чем 4.1. Если используется более старая версия SunOS, следует задуматься о применении измененных библиотек с *ftp.uu.net* либо рассмотреть вариант обновления системы.

Во-первых, необходимо изменить файл *Makefile*, который используется NIS для создания карт – файлов, которые распространяются на все прочие узлы сети. Изменения следует вносить на основном сервере NIS, а не на дополнительных.

На узле SunOS файл сборки для NIS называется */var/yp/Makefile*. Изменения, которые следует внести, очень просты: одну строку следует раскомментировать, а вторую закомментировать. Найдите эти строки:

```
#B=-b
B=
```

и измените их следующим образом:

```
B=-b
#B=
```

Теперь следует произвести сборку карты узлов NIS:

```
# cd /var/yp
# rm hosts.time
# make hosts.time
updated hosts
pushed hosts
```

Теперь в карте *hosts* присутствует «волшебный токен» (magic cookie), предписывающий NIS выполнять запросы к DNS в случаях, когда имя узла не может быть найдено в карте *hosts*. Теперь при поиске имени программа *ypserv* сверяется с соответствующей картой *hosts* для локального домена NIS и, в случае отсутствия имени, посылает запрос

DNS-серверу. Список поиска, используемый *ypserv* при создании запросов к DNS-серверу, получается на основе локального доменного имени NIS (*domainname*) либо инструкции *domain* из файла *resolv.conf*.

Теперь, если существует необходимость, следует создать файл *resolv.conf*. Правила настройки DNS-клиента в SunOS немного отличаются:

- Нельзя установить имя узла (*hostname*) в доменное имя с целью намекнуть клиенту, какой локальный домен следует использовать.
- Также нельзя использовать инструкцию *search* в файле *resolv.conf*, поскольку клиент SunOS 4.x основан на коде из BIND 4.8.1. Клиент молча проигнорирует эту инструкцию.
- *Можно* установить имя домена NIS (*domainname*) в доменное имя (если используется NIS, то это должно быть имя домена NIS), и клиент самостоятельно определит имя локального домена DNS. Однако в данном случае существуют отличия от стандартного поведения BIND; к примеру, если установить *domainname* в значение *fx.movie.edu*, список поиска будет содержать только имя *movie.edu*. Почему в списке поиска не будет имени *fx.movie.edu*? Поскольку NIS полагает, что авторитативный источник данных для *fx.movie.edu* уже проверен – таковым источником является карта *hosts* для *fx.movie.edu*.
- Если локальное доменное имя должно совпадать с именем домена NIS (*domainname*), можно добавить точку или плюс (+) в начало имени *domainname*. Чтобы получить локальное доменное имя *fx.movie.edu*, можно присвоить *domainname* значение *+fx.movie.edu* или *.fx.movie.edu*.
- Можно также изменить стандартное поведение NIS, установив локальное доменное имя с помощью инструкции *domain* в файле *resolv.conf*. Если мы хотим принудительно включить *fx.movie.edu* в список поиска клиента, то можем добавить инструкцию *domain fx.movie.edu* к содержимому файла *resolv.conf*.
- Более того, с помощью инструкции *domain* в файле *resolv.conf* можно задать доменное имя DNS, никак не связанное с именем домена NIS (domainname). В некоторых неудачных вариантах локальное доменное имя NIS (domainname) не идентично, а иногда совершенно не похоже на локальное доменное имя DNS. К примеру, факультет информационных технологий Университета кинематографии изначально создал NIS-домен с именем *it.dept.movieu*, и домен используется по сей день. Чтобы избежать паразитных DNS-запросов, касающихся несуществующего домена *dept.movieu*, узлы в этом домене NIS должны быть настроены с помощью инструкции – например *domain movie.edu* в файле *resolv.conf*.

И наконец, *resolv.conf* от Sun обращается с инструкцией *nameserver* ничуть не хуже, чем лучшие реализации BIND. Поэтому, закончив со-

здавать волшебные токены и выбирать имена доменов NIS и DNS, мы можем перейти к добавлению информации о серверах имен в файл *resolv.conf* и на этом закончить настройку.

Обходимся без NIS

Если необходимо сохранить поддержку от Sun, но при этом обойтись без противной системы NIS, предлагается следующий вариант: можно использовать NIS с пустой картой *hosts*. Создайте файл *resolv.conf*, измените файл сборки NIS, как описано в предыдущем разделе, и создайте пустую карту *hosts*. Для создания пустой карты *hosts* следует всего лишь временно «спрятать» файл */etc/hosts* основного сервера NIS, создать карту NIS, а затем вернуть файл */etc/hosts* на место:

```
% mv /etc/hosts /etc/hosts.tmp
% touch /etc/hosts # чтобы программа make не жаловалась
% cd /var/yp
% make hosts.time
updated hosts
pushed hosts
% mv /etc/hosts.tmp /etc/hosts
```

Теперь, выполняя запросы к NIS, клиент не будет получать результатов, и станет посылать запросы серверу имен.

Если пересборка карт NIS происходит периодически, следует проследить, чтобы карта *hosts* не была по случайности собрана на основе существующего файла */etc/hosts*. Лучший способ это сделать – удалить цель *hosts* из файла сборки NIS (*Makefile*). Можно просто закомментировать строки в файле *Makefile*, начиная с этой:

```
hosts.time: $(DIR)/hosts
```

и заканчивая следующей пустой строкой.

Solaris 2.x от Sun

Клиент в Solaris 2 до версии 2.5.1 основан на коде клиента BIND 4.8.3. В системах Solaris 2.6, 7 и 8 клиенты основаны на коде BIND 4.9.4-P1. Что интересно, Sun не последовала совету документа RFC 1535 – сокращать список поиска до локального доменного имени, поэтому даже в клиентах 2.6 и более поздних версий системы имена всех родительских доменов, содержащие по крайней мере две метки, включаются в список поиска. Существуют «заплаты»-обновления клиентов Solaris 2.5 и 2.5.1 до клиентов BIND 4.9.3.[1]

Все DNS-клиенты в системах Solaris 2.x поддерживают расширения, предоставляющие возможность выбрать порядок, в котором клиент

[1] Информация о версиях текущих обновлений доступна по адресу *http://sunsolve.sun.com/pub-cgi/show.pl?target=patches/patch-access*.

будет консультироваться с различными источниками информации об узлах, включая DNS, NIS, NIS+ и */etc/hosts*. Порядок опроса служб задается в файле *nsswitch.conf*, который находится в каталоге */etc*.

Вообще говоря, файл *nsswitch.conf* нужен для определения порядка, в котором происходит использование ряда различных ресурсов. Следует выбрать базу данных, для которой производится настройка, указав соответствующее ключевое слово. Для DNS-служб имя базы данных – *hosts*. Для базы *hosts* существуют следующие источники: *dns*, *nis*, *nisplus* и *files* (последний, в данном случае, подразумевает */etc/hosts*). Чтобы задать порядок, в котором происходит опрос источников, нужно перечислить их в этом порядке после имени базы данных. К примеру, строка:

```
hosts:  dns files
```

предписывает клиенту использовать сначала DNS (то есть посылать запрос DNS-серверу), а затем файл */etc/hosts*. По умолчанию переход к следующему источнику совершается, если предыдущий источник недоступен либо не найдено искомое имя (то есть, в данном случае, произойдет переход от DNS к */etc/hosts*). Стиль работы можно изменить, создав *условие* и *действие* в квадратных скобках между ресурсами. Существуют следующие условия:

UNAVAIL

Источник не был настроен (в случае DNS – отсутствует файл *resolv.conf* и DNS-сервер не работает на локальном узле).

NOTFOUND

Источник информации не может найти имя, о котором идет речь (в случае DNS – не существует имя или тип записей, для которого производится поиск).

TRYAGAIN

Источник информации занят, но может ответить на следующий запрос (к примеру, истек интервал ожидания клиента при попытке произвести поиск для имени).

SUCCESS

Имя было найдено указанным источником информации.

Для каждого случая можно выбрать связанное действие: *continue* (приводит к использованию следующего источника информации) либо *return* (завершение). Стандартное действие для условия *SUCCESS* – *return*, для всех остальных – *continue*.

К примеру, клиент должен прекращать поиск для доменного имени при получении ошибки NXDOMAIN (доменное имя не существует), но сверяться с файлом */etc/hosts* в случае недоступности DNS:

```
hosts:  dns [NOTFOUND=return] files
```

Кстати говоря, стандартное содержимое файла *nsswitch.conf* в системе Solaris определяется ответами, данными программе *SunInstall*. Верите или нет, но ни один из стандартных вариантов *nsswitch.conf* не содержит в качестве источника *dns*. И такое – от компании с доменом в *.com*?

nscd

В системе Solaris 2.x появился кэширующий демон службы имен *nscd*. *nscd* кэширует результаты поисковых запросов для источников *passwd*, *group* и *hosts*. Можно считать *nscd* аналогом DNS-сервера, специализирующегося на кэшировании, с той разницей, что дополнительно производится кэширование информации из источников *passwd* и *group*. Идея Sun при разработке *nscd* заключалась в повышении производительности путем кэширования часто используемых имен. При этом ходят слухи, что *nscd* в отдельных случаях замедляет поиск в DNS, поэтому многие просто не используют *nscd*. Более того, *nscd* интерферирует с механизмом round robin (*nscd* кэширует записи в одном порядке и не производит их перестановку).

По умолчанию *nscd* запускается в процессе создания многопользовательской рабочей среды при загрузке системы и производит чтение файла настройки – */etc/nscd.conf*. Администраторам доступна настройка некоторых параметров в *nscd.conf*. Наиболее важны следующие:

enable-cache hosts (yes | no)

Параметр определяет необходимость кэширования результатов поиска для узлов

positive-time-to-live hosts value

Параметр определяет длительность хранения положительных результатов (например, адресов), значение задается в секундах

negative-time-to-live hosts value

Параметр определяет длительность хранения отрицательных результатов (например, NXDOMAIN), значение задается в секундах

Если вы не убеждены в полезности *nscd* по меньшей мере в смысле поиска в DNS, воспользуйтесь такой строкой:

```
enable-cache hosts no
```

чтобы отключить кэширование для источника *hosts*.

HP-UX от HP

Реализация клиента от HP – практически обычный клиент BIND. Клиенты в составе систем HP-UX версий с 8.0 по 10.00 основаны на коде BIND 4.8.3, поддерживают стандартные инструкции *domain*, *nameserver* и *search*. Порядок, в котором узел использует DNS, NIS и таблицу узлов, жестко фиксирован. Узел использует DNS, если система настроена (то есть, существует файл *resolv.conf* или работающий локаль-

но DNS-сервер). Если не работает служба DNS, но работает NIS, узел использует NIS. Если не работает ни DNS, ни NIS, используется таблица узлов. Переход к работе с менее предпочтительными источниками информации выполняется в обстоятельствах, описанных выше по тексту (то есть клиент использует только один DNS-сервер – указанный в файле *resolv.conf* либо работающий на локальном узле – и получает четыре ошибки в процессе взаимодействия с этим DNS-сервером).

Фиксированный алгоритм, разумеется, не настолько гибок, как в случае других систем, но легко поддается диагностике и отладке. Если DNS, NIS и таблица узлов могут использоваться в любом порядке, диагностика проблем, возникающих у пользователей, может вызывать огромные затруднения.

Клиенты в составе систем HP-UX версий с 10.10 по 11.00 основаны на коде BIND 4.9.x. Как следствие они работают со списком поиска так же, как BIND 4.9.x, и распознают инструкцию *options ndots*.

Существуют «заплатки» для версий HP-UX 10.x и более поздних, которые обновляют DNS-сервер и вспомогательные программы до BIND 4.9.7. Чтобы получить доступ к этим обновлениям, следует посетить архив обновлений HP-UX, расположенный по адресу *http://us-support.external.hp.com*, и зарегистрироваться. После этого можно будет заняться поиском самых свежих обновлений.

Клиент HP-UX 11.10 основан на коде BIND 8.1.2. Настройка клиента BIND 8.1.2 практически идентична настройке более ранних клиентов, основанных на коде BIND 4.9.x: он понимает те же инструкции и точно так же работает со списком поиска.

В HP-UX 10.00 появилась функциональность файла *nsswitch.conf* Solaris-систем; то есть можно использовать файл *nsswitch.conf* для определения порядка, в котором клиент консультируется с различными службами имен.[1] Синтаксис точно такой же, как для систем Solaris. Стандартные настройки для базы данных hosts на узле под управлением HP-UX:

```
hosts: dns [NOTFOUND=return] nis [NOTFOUND=return] files
```

Функциональность файла *nsswitch.conf*, как и обновления сервера до BIND 4.9.7, также доступны в виде «заплат» для версий HP-UX вплоть до 9.0. Их можно найти в веб-архиве обновлений HP-UX. «Заплат» может потребоваться довольно много:

- Одна для стандартной разделяемой библиотеки C, *libc.so*, которая содержит функции клиента в HP-UX.

[1] До появления HP-UX 10.10 для настройки порядка опроса информационных источников файл *nsswitch.conf* можно было использовать только для базы *hosts*. Начиная с версии 10.10 стали доступны базы *services*, *networks*, *protocols*, *rpc* и *netgroup*.

- Одна для программы *mount*, которая статически связывается с библиотеками.
- Одна для программы *nslookup*.
- Одна для программ *ifconfig* и *route*.
- Одна для HP Visual User Environment (VUE, графическая среда пользователя) либо Common Desktop Environment (CDE, базовая среда рабочего стола), которые поставляются в виде статически связанных с библиотеками исполняемых файлов.

AIX от IBM

Клиенты в составе более поздних версий системы AIX, включая 4.3 и 4.2.1, также являются относительно стандартными. В основе лежит код BIND 4.9.x, поэтому клиенты распознают инструкции *domain*, *search*, *nameserver*, *options* и *sortlist*; AIX поддерживает до трех инструкций *nameserver*. AIX версий 4 и 4.1 содержит DNS-клиент, основанный на коде BIND 4.8.3, который распознает все те же инструкции, что и клиент AIX 4.2.1, за исключением *options* и *sortlist*.

Различие между поведением клиента AIX и широко распространенного BSD-варианта заключается в том, что AIX считает наличие файла *resolv.conf* предписанием посылать запрос DNS-серверу. Если *resolv.conf* не существует на локальном узле, DNS-клиент использует файл */etc/hosts*. Это означает, что если на узле работает DNS-сервер, следует создать пустой файл */etc/resolv.conf* даже в том случае, если он не будет использоваться для настройки.

AIX 4.3

Клиент AIX 4.3 использует две переменных среды – RES_TIMEOUT и RES_RETRY, которые позволяют изменять начальные интервалы ожидания клиента (а-ля инструкция *options timeout*) и число попыток (а-ля *options attempts*). Эти значения можно устанавливать в загрузочном сценарии командного интерпретатора либо в командной строке, как показано в следующем примере:

```
# RES_TIMEOUT=2 /usr/sbin/sendmail -bd -q1h
```

AIX 4.3 реализует механизм управления порядком разрешения, который связан с файлом *irs.conf* и схож с Solaris-вариантом – *nsswitch.conf*. Однако синтаксис немного отличается. База данных, как и в *nsswitch.conf*, называется *hosts*. Имена источников информации практически такие же (*dns*, *nis* и *local* вместо *files*), но в AIX может использоваться слово *continue* в конце строки, чтобы предписать клиенту продолжить поиск с первого источника в следующей строке. Чтобы показать, что источник информации является авторитативным и что клиент не должен переходить к использованию следующего источника в случае получения от авторитативного источника отрицательного

ответа (как в случае с *[NOTFOUND=return]*), следует добавить метку *=auth* к аргументу. Чтобы предписать клиенту прежде всего обращаться к службе DNS, а затем переходить к использованию */etc/hosts* – только в случае, когда служба DNS не настроена, можно создать следующий файл *irs.conf*:

```
hosts dns=auth continue
hosts local
```

Если необходимо определять порядок для каждого пользователя в отдельности либо замещать настройки по умолчанию, можно воспользоваться переменной среды NSORDER. NSORDER может содержать те же аргументы, что и строки файла *irs.conf*, и имеет формат списка, элементы которого разделяются запятыми:

```
NSORDER=dns,local
```

Как и в файле *irs.conf*, можно отметить авторитативный источник строкой *=auth*:

```
NSORDER=dns=auth,local
```

AIX 4.2.1

Используемый в AIX 4.2.1 механизм управления порядком опроса источников информации при разрешении не сильно отличается, но более ограничен. В AIX 4.2.1 используется файл */etc/netsvc.conf*. В пределах этого файла база данных также называется *hosts*, но отделяется от перечня источников символом равенства, а не двоеточием, при этом источник DNS обозначается именем *bind*, а файл */etc/hosts* именем *local*. Строка:

```
hosts = local,nis,bind
```

предписывает клиенту AIX прежде всего сверяться с локальным файлом */etc/hosts*, затем с картой NIS (*hosts*), и наконец – использовать DNS. Как и в AIX 4.3, отдельные пользователи или процессы имеют возможность изменять порядок разрешения, заданный в файле *netsvc.conf*, путем установки значения переменной среды *NSORDER*.

Следует также отметить, что клиент можно настраивать с помощью инструмента управления системой AIX – System Management Interface Tool (SMIT).

Tru64 Unix и Digital Unix от Compaq

Клиент, поставляемый в составе системы Tru64 Unix 5.0, основан на коде клиента BIND 8.1.2. Клиент, поставляемый в составе системы Digital Unix 4.0, основан на коде клиента BIND 4.9.x. Соответственно, обе реализации распознают пять основных инструкций настройки,

описанных в этой главе, за исключением появившихся в BIND 8.2, таких как *options timeout*.

Клиент системы Tru64 Unix 5.0 производит проверку имен, но также позволяет использовать в доменных именах символы, традиционно недопустимые. Необходимо просто перечислить символы, маскируя их обратным слэшем, после инструкции *allow_special*. Так, сделать допустимым использование подчеркиваний можно с помощью следующей инструкции:

```
allow_special \_
```

Аргумент *all* является предписанием разрешить использование любых символов, но навряд ли это хорошая мысль.

Оба варианта Unix от Compaq предоставляют возможность определения порядка опроса клиентом служб NIS, DNS и таблицы файлов. Для этого используется файл *svc.conf* (руководство по *svc.conf(4)* рекомендуется к прочтению).[1] *svc.conf* позволяет задавать порядок опроса служб для различных баз данных, включая почтовые псевдонимы, записи идентификации (отображения IP-адресов в имена узлов или доменные), пароли и информацию о группах, а также уйму других настроек.

Настройка клиента с помощью файла *svc.conf* связана с использованием имени базы данных *hosts*. За именем следует знак равенства, а затем ключевые слова, идентифицирующие службы, которые следует опрашивать. Ключевые слова разделяются запятыми, порядок перечисления определяет порядок использования. Для базы данных *hosts* допустимы ключевые слова *local* (*/etc/hosts*), *yp* («Yellow Pages», прежнее название службы NIS) и *bind* (DNS). Ключевое слово *local* должно быть первым из перечисленных для *hosts*. Пробелы в строке допустимы только после запятых и в конце строки. К примеру, строка

```
hosts=local,bind
```

является предписанием клиенту сверяться прежде всего с файлом */etc/hosts* при поиске имен узлов и, в случае отсутствия результата, обращаться к службе доменных имен. Это удобно, если таблица узлов имеет небольшой размер и содержит доменное имя локального узла, его IP-адрес, маршрутизатор, используемый по умолчанию, и прочую информацию о различных узлах, которая может понадобиться при загрузке системы. Таблица узлов имеет более высокий приоритет, и это позволяет избежать проблем, связанных со службой доменных имен, в процессе загрузки, когда сетевые службы и *named* могут быть еще не запущены.

В Unix-системах от Compaq существует вспомогательная программа, которая называется *svcsetup* (рекомендуется к прочтению руководство

[1] Бедный старый Ultrix также реализует поддержку *svc.conf*.

по *svcsetup(8)*). Она позволяет создавать файл *svc.conf* в диалоговом режиме, без применения текстового редактора. Набрав *svcsetup*, администратор получает возможность выбрать базу данных, для которой будет производиться настройка. *svcsetup* также запрашивает порядок использования служб, с которыми следует сверяться клиенту.

IRIX от Silicon Graphics

В IRIX версии 6.5 используются клиент и DNS-сервер, созданные на основе кода BIND 4.9.x. Клиент распознает инструкции *domain*, *search*, *nameserver*, *options* и *sortlist*. В предыдущей версии IRIX, 6.4, присутствовал DNS-сервер на основе кода BIND 4.9.x, но клиент – на основе кода BIND 4.8.3. Существуют «заплаты» для версий IRIX вплоть до 5.3, обновляющие DNS-сервер до BIND 4.9.7. Информация о текущих версиях заплат доступна по адресу *http://support.sgi.com/colls/patches/tools/browse*.

В системах IRIX 6.x файл *resolv.conf* переехал из каталога */usr/etc*, в котором проживал раньше, в стандартный */etc*. (В целях сохранения работоспособности программ, собранных в более ранних версиях IRIX, может возникнуть необходимость создания линка из */usr/etc/resolv.conf* на */etc/resolv.conf*.)

IRIX 6.5 так же, как Solaris 2.x и системы HP-UX, поддерживает настройку с помощью файла *nsswitch.conf*. Формат файла *nsswitch.conf* в IRIX точно такой же как в Solaris, с добавлением варианта *noperm* (недостаточно прав для использования службы) к перечню существующих условий. По умолчанию существует следующий порядок для базы данных *hosts*:

```
hosts: nis dns files
```

Демон службы имен IRIX, *nsd*, обращает внимание на файл *nsswitch.conf*. Как и демон *nscd* в исполнении Sun, *nsd* занимается сопровождением общесистемного кэша, в котором хранятся результаты выполненных запросов, включая информацию об узлах, полученную от DNS и NIS. В *nsd* реализована поддержка настройки в *nsswitch.conf* многочисленных атрибутов, которые отсутствуют в системах Solaris и HP-UX. К примеру, можно добавить значение интервала ожидания в скобках, чтобы обозначить для *nsd* длительность хранения полученных от DNS записей:

```
hosts: files dns (timeout=600) # удаление из кэша через 10 минут
```

Время хранения для отрицательных ответов можно задать с помощью *negative_timeout*. Полный перечень атрибутов приводится на страницах руководства *nsd(1m)*.

Более старые клиенты IRIX (до версии 6.4) поддерживают инструкцию *hostresorder* вместо файла *nsswitch.conf*. Как и *nsswitch.conf*, ин-

струкция *hostresorder* позволяет администратору определять порядок, в котором производится поиск в NIS, DNS и локальной таблице узлов. Пользователи могут определить порядок использования с помощью переменной среды *HOSTRESORDER*. DNS-клиент IRIX 6.5 игнорирует инструкцию *hostresorder*.

Аргументами *hostresorder* выступают ключевые слова *nis*, *bind* и *local* (не менее одного). Ключевые слова могут разделяться пробелами либо символом слэша. Пробел означает, что следующий источник информации следует использовать только в том случае, если ответ не был получен от предыдущего (например, имя не найдено в таблице узлов или DNS-сервер сообщает, что имя не существует), либо источник информации был недоступен (скажем, не запущен DNS-сервер). Слэш указывает, что предшествующий источник информации является авторитативным, и если ответ не получен, следует прервать процесс разрешения. Следующий источник информации в этом случае используется, только если авторитативный источник не был доступен в момент выполнения запроса.

Linux

Со времени первого издания этой книги Linux успел взять мир компьютеров приступом. Причины очевидны: Linux бесплатен и гораздо лучше поспевает за последними новшествами мира Unix и сети Интернет, чем любая из коммерческих Unix-систем. В пользу этих слов говорит тот факт, что в Red Hat Linux 7.0, самой последней версии одной из доминирующих разновидностей Linux, присутствует DNS-сервер BIND 8.2.2-P5. Правда, клиент по-прежнему основан на коде BIND 4.9.x. Он поддерживает настройку с помощью файла *nsswitch.conf*.

Однако некоторые из более старых клиентов в составе систем Linux основаны на библиотеке Билла Уизнера *resolv+*, которая, в свою очередь, основана на коде BIND 4.8.3. Как следствие, файл *resolv.conf* может содержать любые допустимые в версии клиента 4.8.3 инструкции (*domain*, *search* и *nameserver*, но не *options* и *sortlist*), а список поиска по умолчанию присутствует в более древнем варианте.

Библиотека *resolv+*, как можно догадаться по названию, содержит некоторые улучшения по сравнению со стандартным клиентом версии 4.8.3. Среди них – способность определять порядок использования DNS, NIS и файла */etc/hosts* (в новых версиях на смену пришел более стандартный файл *nsswitch.conf*), способность к обнаружению определенных случаев подделки пакетов DNS и способность изменять порядок адресных записей в ответах, отдавая предпочтение локальным подсетям.

Все эти нововведения управляются файлом */etc/host.conf*. Вот некоторые из наиболее интересных ключевых слов, допустимых в *host.conf*:

order

Определение порядка использования различных служб имен; допустимые аргументы: *bind*, *hosts* и *nis*, причем должен присутствовать хотя бы один. Разделителем в списке аргументов служит запятая.

nospoof

Единственный аргумент может принимать значение *on* или *off*. *nospoof* предписывает клиенту проверять всю информацию по обратному отображению (PTR), получаемую от удаленных DNS-серверов, путем выполнения прямых (адресных) запросов для доменных имен, содержащихся в ответах. Если адрес, возвращаемый прямым запросом, не совпадает с адресом, для которого клиент исходно производил поиск, PTR-запись игнорируется.

reorder

Единственный аргумент может принимать значение *on* или *off*. Включение предписания *reorder* приводит к сортировке клиентом адресов узлов, расположенных в нескольких сетях, таким образом, что адреса, принадлежащие локальным подсетям, оказываются в начале списка.

Windows 95

Windows 95 реализует свой собственный стек TCP/IP и клиент DNS. Более того, в Windows 95 *два* TCP/IP-стека: один для TCP/IP локальных сетей, а второй – для TCP/IP коммутируемых соединений. Настройка клиента в Windows 95, естественно, производится с помощью графического интерфейса. Чтобы добраться до главной панели настройки DNS, следует выбрать в главном меню пункт Control Panel, затем открыть *Network*, и выбрать *TCP/IP protocol*. Это приведет к появлению диалогового окна, которое выглядит примерно так, как показано на рис. 6.1. Выберите закладку *DNS Configuration*.

Настройка с помощью этой панели достаточно прозрачна: для включения разрешения DNS следует отметить пункт *Enable DNS*, затем указать имя машины (в данном случае, первую метку доменного имени) в поле *Host* и локальное доменное имя (все метки после первой точки) в поле *Domain*. В раздел *DNS Server Search Order* можно добавить до трех DNS-серверов, которые следует опрашивать. И наконец, доменные имена для списка поиска добавляются в раздел *Domain Suffix Search Order* в порядке предпочтения. Если оставить раздел *Domain Suffix Search Order* пустым, клиент Windows 95 создает список поиска на основе локального доменного имени так же, как это сделал бы клиент BIND 4.8.3.

Интересное замечание относительно текущей версии Windows 95: для каждого коммутируемого соединения с каждым провайдером интернет-услуг можно задать отдельный набор DNS-серверов в настройках

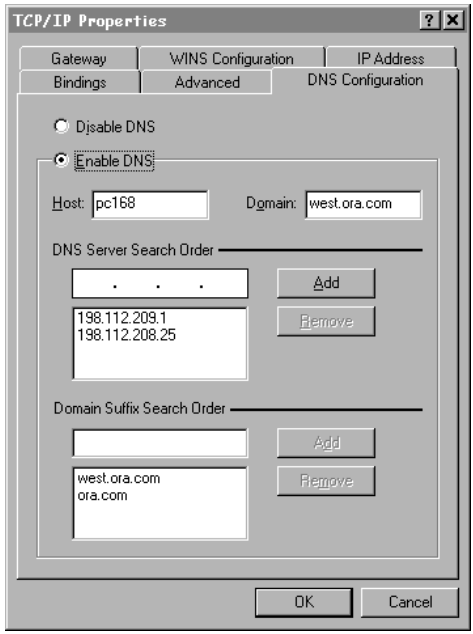

Рис. 6.1. *Настройка клиента в Windows 95*

коммутируемых соединений (удаленного доступа к сети, DUN). Чтобы указать настройки, специфичные для удаленного доступа к сети, следует перейти в папку *My Computer*, расположенную на рабочем столе, затем двойным щелчком в *Dial-up Networking*; правым щелчком мыши вызвать контекстное меню соединения, для которого необходимо изменить настройки клиента, и выбрать пункт *Properties*. Затем необходимо выбрать закладку *Server Types* и щелкнуть по *TCP/IP Settings*. Появится окно, показанное на рис. 6.2.

Если оставить выбранным вариант *Server assigned name server addresses*, клиент будет получать список DNS-серверов, с которыми следует работать, от сервера провайдера, с которым установлено соединение. Если выбрать вариант *Specify name server addresses* и указать адреса одного или двух DNS-серверов, Windows 95 будет пытаться использовать эти адреса при установленном подключении.

Это весьма удобно, если вы пользуетесь услугами нескольких интернет-провайдеров и у каждого из них свои DNS-серверы. При этом указание DNS-серверов в панели *TCP/IP Properties* имеет более высокий приоритет, чем настройки DNS-серверов для отдельных соединений. Чтобы воспользоваться частными настройками для коммутируемых соединений, следует оставить панель *TCP/IP Properties* пустой, если не считать включения работы с DNS и указания имени локального узла. Это ограничение вызвано недостаточной интеграцией TCP/IP-сте-

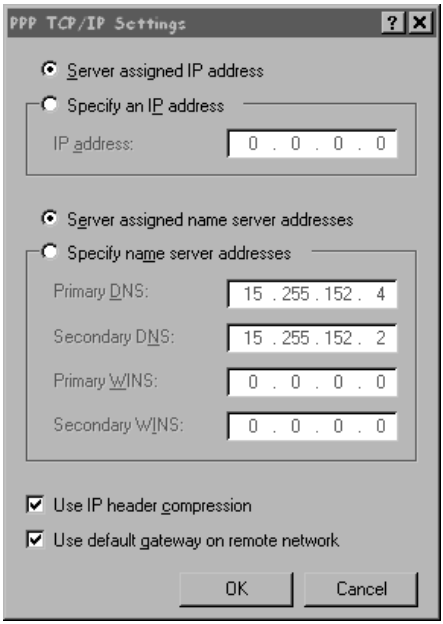

Рис. 6.2. *Настройка клиента для удаленного доступа к сети в Windows 95*

ков Windows 95, в DUN версии 1.3 оно устранено. Более подробно об этом можно прочитать в статье Q191494 базы знаний Microsoft.[1]

Windows 98

Клиент в Windows 98 практически идентичен клиенту Windows 95. (А визуально *абсолютно* идентичен, поэтому обойдемся без картинки для этого случая.) Основное различие между клиентами заключается в том, что в составе Windows 98 поставляется Winsock 2.0.[2]

Winsock 2.0, например, сортирует запросы в соответствие с локальной таблицей маршрутизации. Если DNS-сервер возвращает несколько адресов в ответе, и один из этих адресов принадлежит сети, в которую существует явно определенный (пусть и не стандартный) маршрут, клиент помещает этот адрес в начало ответа. Подробности в статье Q182644 базы знаний Microsoft.

---

[1] Производить поиск статей базы знаний Microsoft по их идентификаторам можно со страницы *http://search.support.microsoft.com/kb*: следует выбрать вариант *Specific article ID number* (поиск по идентификатору конкретной статьи) и ввести идентификационный номер в поле поискового запроса.

[2] Программа Winsock в Windows 95 может быть обновлена до версии 2.0; см. статью Q182108 базы знаний Microsoft.

В Windows 98 также существует возможность частного указания DNS-серверов для отдельных коммутируемых соединений. Клиент работает с серверами, перечисленными на панели *TCP/IP Properties*, и с серверами, заданными для конкретного коммутируемого соединения, – одновременно, и принимает первый положительный ответ, полученный от сервера любого набора. Если клиент получает от всех серверов только отрицательный ответ, то возвращает отрицательный ответ.

Windows NT 4.0

В Windows NT настройка клиента для локальных сетей производится с помощью единственной панели, которая поразительно напоминает соответствующую панель Windows 95, поскольку в системе NT 4.0 была использована «оболочка» Windows 95. По сути дела, если не считать новой кнопки *Edit* и наличия удобных маленьких стрелочек, позволяющих изменять порядок элементов в списках DNS-серверов и списке поиска, семантические различия отсутствуют, что прекрасно видно на рис. 6.3.

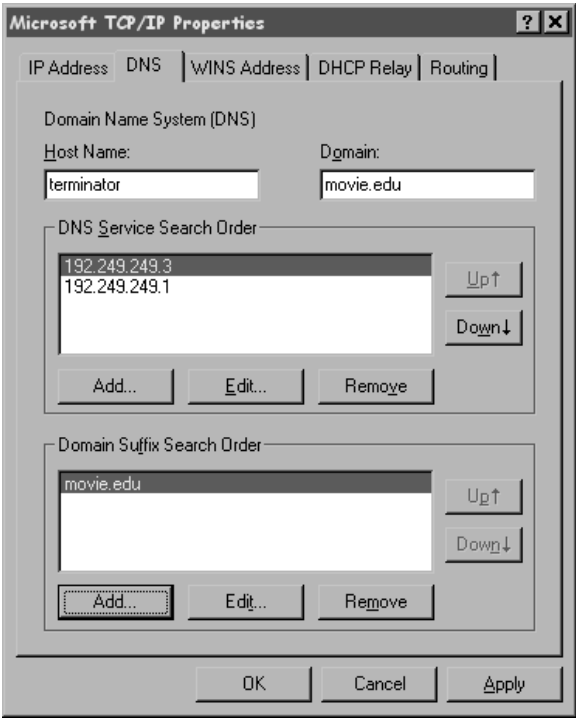

Рис. 6.3. *Настройка клиента в Windows NT*

Чтобы добраться до панели *DNS Configuration*, следует выбрать пункт меню Control Panel, открыть элемент *Network*, и перейти на закладку *Protocols*. Затем двойным щелчком выбрать *TCP/IP Protocol*, закладку для DNS.

Windows NT также позволяет пользователям производить частную настройку клиента для конкретных коммутируемых соединений. Щелкните по иконке *My Computer*, выберите *Dial-Up Networking*, откройте выпадающее меню и выберите имя коммутируемого соединения, для которого необходимо произвести настройку клиента. Затем из меню *More* выберите *Edit Entry* и *Modem Properties*. В полученном окне выберите закладку *Server* и нажмите на кнопку *TCP/IP Settings*. Это приведет к получению того же окна, что мы уже видели в Windows 95 (показано ранее). Если оставить выбранным вариант *Server assigned name server addresses*, клиент будет получать список адресов DNS-серверов от сервера, с которым установлено соединение. Если выбрать вариант *Specify name server addresses* и указать адреса одного или двух DNS-серверов, Windows NT будет использовать эти DNS-серверы, когда коммутируемое соединение установлено. При разрыве коммутируемого соединения, Windows NT возвращается к настройкам клиента для локальных сетей.

Клиент в составе Windows NT 4.0 производит кэширование соответствий имя-адрес для каждого процесса в отдельности, с учетом времени жизни для каждой полученной записи. Браво, Microsoft!

Клиент довольно существенно обновился в Windows NT 4.0, Service Pack 4. Клиент SP4 поддерживает sortlist, как и клиент BIND 4.9.x, но список сортировки не поддается настройке. Вместо этого список сортировки привязан к таблице маршрутизации машины: адреса сетей, в которые существуют прямые маршруты, помещаются в начало списка ответа. Если такое поведение не устраивает – скажем, мешает работе механизма round robin – его можно отменить с помощью нового ключа реестра. Подробности процесса – в статье Q196500 базы знаний Microsoft.

В клиенте SP4 также существует возможность отключать кэширование путем установки (ни за что не угадаете) значения ключа реестра. Подробности процесса – в статье Q187709 базы знаний Microsoft.

Еще одна новость SP4 – новый алгоритм переключений. Клиент попрежнему посылает первый запрос первому DNS-серверу из списка *DNS Server Search Order*. Но при этом интервал ожидания ответа установлен в одну секунду, по истечению которой происходит передача запроса *всем* DNS-серверам, о которых известно системе – из статической настройки, от DHCP и RAS. Если ни один из этих серверов не ответил в пределах двух секунд, клиент производит повторную посылку запросов всем серверам. Удвоение интервала ожидания и повторение запросов происходит не более четырех раз и занимает по времени до 15 секунд. Подробности – в статье Q198550 базы знаний Microsoft.

Для системных администраторов такое поведение означает повышенную нагрузку на DNS-серверы, поэтому следует позаботиться о том, чтобы первый из упомянутых в списке клиента SP4 (*DNS Server Search Order*) серверов был быстрым (и возвращал большую часть ответов менее чем за секунду), и чтобы в списке DNS-серверов *DNS Server Search Order* был лишь необходимый минимум.

Windows 2000

Клиент в системе Windows 2000 найти не очень просто. Чтобы добраться до него, нажмите на кнопку *Start*, выберите пункт *Settings*, а затем *Network and Dial-up Connections*. Это приводит к открытию окна, показанного на рис. 6.4.

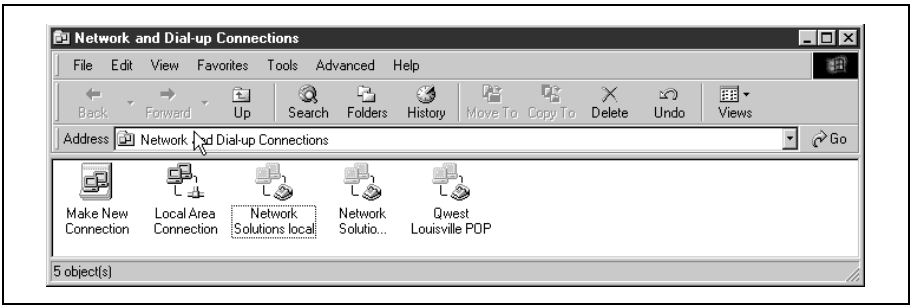

Рис. 6.4. *Windows 2000: Сеть и удаленный доступ*

Нажатием правой кнопки мыши вызовите контекстное меню для *Local Area Connection* и выберите пункт *Properties*. Это приводит к открытию окна, показанного на рис. 6.5.

Двойной щелчок по *Internet Protocol (TCP/IP)* приводит к появлению окна настройки, показанного на рис. 6.6.

Если выбрать вариант *Obtain DNS server address automatically*, клиент посылает запросы DNS-серверу, о котором сообщает локальный DHCP-сервер. В случае выбора *Use the following DNS server addresses*, клиент использует DNS-серверы, перечисленные в полях *Preferred DNS server* и *Alternate DNS server*.[1]

Более подробные настройки клиента доступны по нажатию (ну, разумеется) кнопки *Advanced*... Закладка *DNS* выглядит, как показано на рис. 6.7.

---

[1] И снова похвалы Microsoft – за более прозрачные обозначения. В предыдущих версиях Windows серверы имен могли обозначаться как *Primary DNS* и *Secondary DNS*. Это иногда приводило к тому, что пользователи перечисляли первичный мастер- и вторичный серверы имен для какой-либо зоны в этих полях. Кроме того, сокращение «DNS» расшифровывается как «Domain Name System» (система доменных имен), а не «domain name server» (сервер доменных имен).

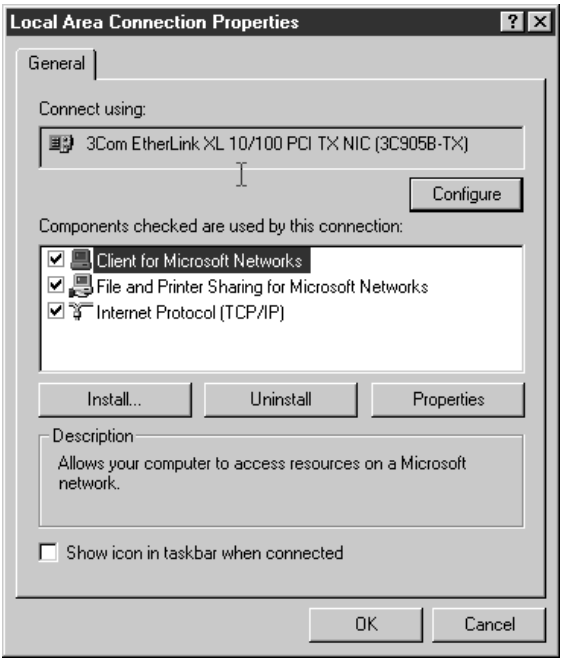

Рис. 6.5. *Windows 2000: свойства Local Area Connection*

Если адреса DNS-серверов, которым посылаются запросы, были определены в основном окне настройки клиента, они будут присутствовать в окне дополнительных настроек в разделе *DNS server addresses, in order of use:*. Как и в окне настройки клиента Windows NT 4.0, кнопки позволяют добавлять, редактировать, удалять DNS-серверы, а также изменять порядок перечисления. Ограничения на количество DNS-серверов в списке, судя по всему, нет, но и нет особого смысла указывать больше трех.

В клиенте Windows 2000 используется такой же алгоритм переключения, как в клиенте Windows NT 4.0 SP4: происходит повторная передача всем указанным DNS-серверам. И поскольку для каждого сетевого интерфейса (или адаптера, как выражаются в Microsoft) может создаваться отдельный список DNS-серверов, общее количество запросов к различным DNS-серверам может быть довольно большим. Подробности – в статье Q217769 базы знаний Microsoft.

Во времена расщепленных пространств имен вполне реально получить разные ответы от двух DNS-серверов, поэтому клиент Windows 2000 временно игнорирует отрицательные ответы (отсутствие доменного имени или запрошенного типа данных) при посылке параллельных запросов нескольким DNS-серверам. Только при получении отрицательного ответа от DNS-сервера, присутствующего в списках всех интер-

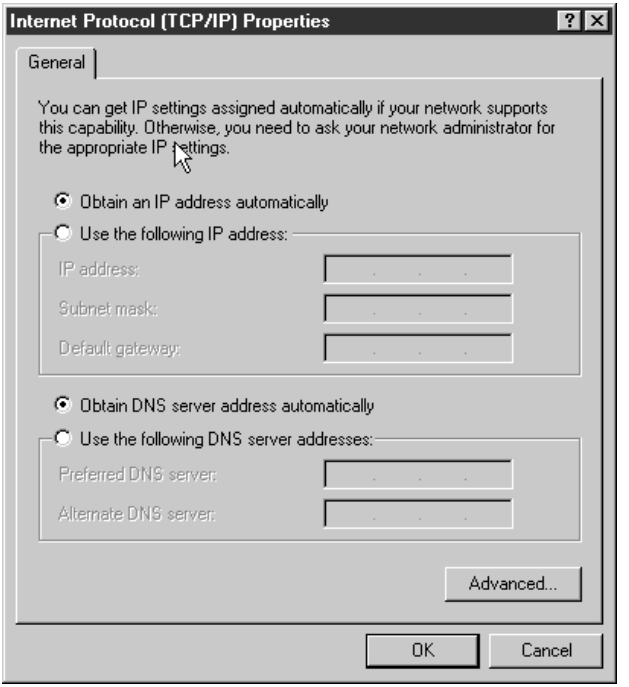

Рис. 6.6. *Основные настройки клиента Windows 2000*

фейсов, клиент возвращает отрицательный ответ. Если клиент получает хотя бы один положительный ответ от любого из DNS-серверов, то возвращает этот ответ.

Выбор варианта *Append primary and connection specific DNS suffixes* предписывает клиенту использовать основной (primary) суффикс DNS и суффиксы, определенные для частных соединений, для создания списка поиска. Суффикс DNS для данного конкретного соединения можно определить в этом окне, в поле справа от надписи *DNS suffix for this connection*. Основной суффикс DNS устанавливается из панели управления (Control Panel): следует щелкнуть по компоненте *System*, выбрать закладку *Network Identification*, затем нажать на кнопку *Properties*, и далее *More...* В результате открывается окно, показанное на рис. 6.8.

Основной суффикс DNS следует указать в поле под надписью *Primary DNS suffix of this computer*.

Установка флажка *Append parent suffixes of the primary domain suffix* (рис. 6.7) позволяет настроить клиент на работу со списком поиска в стиле BIND 4.8.3, для создания которого используется основной суффикс DNS. Для основного суффикса *fx.movie.edu* список поиска будет содержать *fx.movie.edu* и *movie.edu*. Помните, что суффиксы DNS, определенные для частных соединений, не «переходят» (в терминологии

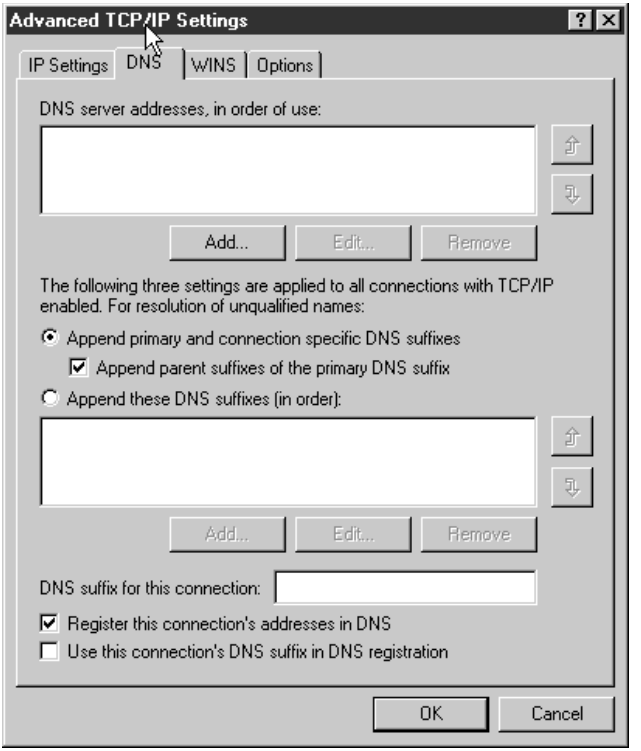

Рис. 6.7. *Дополнительные настройки клиента Windows 2000*

Microsoft) в список поиска, но в случае наличия частного суффикса для соединения он включается в список поиска.

Выбор варианта *Append these DNS suffixes (in order)* позволяет настроить клиент на использование списка поиска, составляемого перечисленными ниже записями. Как и в случае со списком DNS-серверов, доступны операции добавления, редактирования, удаления и изменения порядка записей с помощью кнопок и стрелок.

Наконец, следует упомянуть две возможности внизу окна. *Register this connection's addresses in DNS* определяет, следует ли клиенту пробовать производить динамические обновления для добавления адресной записи, отображающей имя клиента в адрес, для данного соединения. *Use this connection's suffix in DNS registration* позволяет определить, какое имя следует использовать при обновлении – доменное имя, связанное с данным соединением, или суффикс DNS компьютера.

Смысл автоматической регистрации в том, чтобы доменное имя клиента Windows 2000 всегда было связано с текущим IP-адресом, даже если адрес получен от DHCP-сервера. (На самом деле сервер DHCP добав-

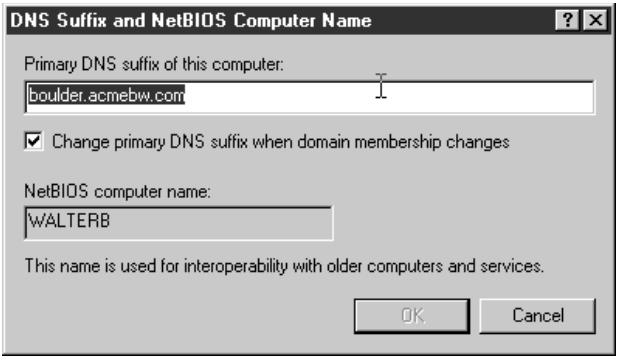

Рис. 6.8. *Настройка основного DNS-суффикса в Windows 2000*

ляет PTR-запись для отображения IP-адреса клиента в его доменное имя.) Автоматическая регистрация – это похоронный звон для WINS (Windows Internet Name Service, Windows-служба имен Интернета) – фирменной службы Microsoft для NetBIOS, которая подвергается незаслуженным нападкам. Когда все клиенты будут работать под управлением Windows 2000, все они будут использовать динамические обновления для управления правильностью преобразования имен в адреса, и появится повод забить осиновый кол в сердце WINS.

Однако динамическое обновление зон клиентами связано с некоторыми, скажем так, сложностями, которые мы рассмотрим в последней главе книги.

7

Работа с BIND

- *Управление DNS-сервером*
- *Обновление файлов данных зон*
- *Организация файлов*
- *Перемещение системных файлов в BIND 8 и 9*
- *Ведение log-файла в BIND 8 и 9*
- *Основы благополучия*

– У нас, – сказала Алиса, с трудом переводя дух, – когда долго бежишь со всех ног, непременно попадешь в другое место.

– Какая медлительная страна! – сказала Королева. – Ну, а здесь, знаешь ли, приходится бежать со всех ног, чтобы только остаться на том же месте! Если же хочешь попасть в другое место, тогда нужно бежать по меньшей мере вдвое быстрее!

Эта глава содержит ряд родственных тем, связанных с сопровождением DNS-сервера. Мы затронем управление DNS-серверами, изменение файлов данных зон, обновление файла корневых указателей. Будут упомянуты распространенные ошибки, которые можно встретить в log-файле демона *syslog*, а также статистика, которую хранит BIND.

В этой главе не рассматриваются вопросы диагностирования и разрешения проблем. Сопровождение включает обеспечение актуальности информации и наблюдение за работой DNS-серверов. Разрешение проблем похоже на тушение пожара – периодические выезды по тревоге DNS. Борьба с пожарами описана в главе 14 «Разрешение проблем DNS и BIND».

Управление DNS–сервером

Традиционно администраторы управляли DNS-сервером *named* с помощью сигналов Unix. DNS-сервер интерпретирует получение определенного сигнала в качестве руководства к определенным действиям, например, перезагрузке всех изменившихся зон, для которых он является первичным мастер-сервером. Однако количество существующих сигналов ограничено, а помимо этого сигналы не дают возможности передать с командой дополнительную информацию (например, доменное имя зоны), которую следует перезагрузить.

В BIND 8.2 консорциум ISC представил новый метод управления DNS-сервером – посылку сообщений через специальный управляющий канал. Управляющий канал может быть сокетом Unix, либо TCP-портом, через который DNS-сервер принимает сообщения. Управляющий канал не ограничивается конечным числом абстрактных сигналов и потому является механизмом более мощным и более гибким. ISC утверждает, что управляющий канал – дорога в будущее, и что администраторам следует использовать для всех операций управления DNS-серверами именно этот инструмент, а не сигналы.

Послать сообщение DNS-серверу через управляющий канал можно с помощью программы *ndc* (BIND 8) или *rndc* (BIND 9). *ndc* появилась в BIND версии 4.9, но до появления пакета BIND 8.2 это был просто сценарий командного интерпретатора, который позволял использовать привычные аргументы-команды (скажем, *reload*) вместо сигналов (к примеру, *HUP*). Об этой версии *ndc* мы вспомним чуть позже.

ndc и controls (BIND 8)

Выполняемая без аргументов, программа *ndc* пытается связаться с DNS-сервером, работающим на локальном узле, посылая сообщения через Unix-сокет. Этот сокет обычно называется */var/run/ndc*, хотя в некоторых операционных системах используются другие имена. Сокет обычно принадлежит пользователю *root*, и только владелец имеет права на чтение и запись. DNS-серверы BIND версии 8.2 и более поздних создают Unix-сокет при запуске. Существует возможность указать альтернативное имя файла или права доступа к сокету – с помощью оператора *controls*. К примеру, чтобы изменить имя сокета на */etc/ndc*, а группу-владельца на *named*, а также сделать сокет доступным для чтения и записи владельцем и группой, можно воспользоваться следующим оператором:

```
controls {
  unix "/etc/ndc" perm 0660 owner 0 group 53;  // группа 53 - это "named"
};
```

Значение, определяющее права доступа, должно быть указано в восьмеричной системе счисления (этот факт отражен вводной цифрой 0). Те из читателей, кто не знаком с этим форматом, могут обратиться к страницам руководства команды *chmod(1)*. Значения группы и владельца также должны задаваться численными идентификаторами.

ISC рекомендует – и мы совершенно согласны – предоставлять доступ к этому Unix-сокету только управляющему персоналу, который имеет право работать с DNS-сервером.

ndc можно также использовать для посылки DNS-серверу сообщений через TCP-сокет, причем вполне возможно – с удаленного узла. Для этого следует выполнить *ndc* с ключом командной строки *-c*, указав

имя или адрес DNS-сервера, а затем, после символа слэша, номер порта, через который сервер принимает управляющие сообщения. Пример:

```
# ndc -c 127.0.0.1/953
```

Настройка сервера на прослушивание определенного TCP-порта с целью получения управляющих сообщений производится с помощью оператора *controls*:

```
controls {
 inet 127.0.0.1 port 953 allow { localhost; };
};
```

По умолчанию DNS-серверы BIND 8 не производят прослушивание каких-либо TCP-портов. DNS-серверы BIND 9 по умолчанию принимают сообщения через порт 953, именно этот порт мы и использовали для настройки. В данном случае DNS-серверу предписывается принимать сообщения только с локального адреса loopback-интерфейса и пропускать только сообщения с локального узла. Это не слишком благоразумно, поскольку любой человек, имеющий доступ на локальный хост, сможет контролировать DNS-сервер. Если бы мы были еще более неосмотрительны (никогда не будьте такими), то могли бы изменить список доступа и разрешить DNS-серверу принимать сообщения со всех локальных сетевых интерфейсов:

```
controls {
 inet * port 953 allow { localnets; };
};
```

ndc поддерживает два режима работы – диалоговый и командный. В командном режиме команда DNS-серверу указывается в командной строке, к примеру, так:

```
# ndc reload
```

Если не указать команду в качестве аргумента, произойдет переход в диалоговый режим:

```
# ndc
Type   help  -or-   /h   if you need help.
ndc>
```

Команда */h* приводит к получению перечня команд, которые понимает *ndc* (не DNS-сервер). Эти команды относятся к работе *ndc*, а не сервера:

```
ndc> /h
        /h(elp)                 this text
        /e(xit)                 leave this program
        /t(race)                toggle tracing (protocol and system events)
        /d(ebug)                toggle debugging (internal program events)
        /q(uiet)                toggle quietude (prompts and results)
```

```
        /s(ilent)              toggle silence (suppresses nonfatal errors)
ndc>
```

Команда */d* является предписанием *ndc* создавать отладочный вывод (к примеру, содержащий информацию о том, что посылается DNS-серверу и какие ответы возвращаются). Эта команда никак не связана с уровнем отладки для DNS-сервера, в отличие от описанной позже команды *debug*.

Обратите внимание, что именно команда */e* (а не */x* или */q*) позволяет завершить работу с *ndc*. Можно сказать, это несколько непривычно.

help перечисляет существующие команды, которые позволяют осуществлять управление DNS-сервером:

```
ndc> help
getpid
status
stop
exec
reload [zone] ...
reconfig [-noexpired] (just sees new/gone zones)
dumpdb
stats
trace [level]
notrace
querylog
qrylog
help
quit
ndc>
```

Существуют две команды, которые не отражены в списке, но могут использоваться: *start* и *restart*. Их нет в списке, потому что в данном случае *ndc* перечисляет команды, которые понимает DNS-сервер, а не сама программа *ndc*. DNS-сервер не может выполнить команду *start* – чтобы сделать это, он должен работать (а если он работает, то его незачем запускать). DNS-сервер также не может выполнить команду *restart*, поскольку, если он завершает работу, то после этого уже не может запустить себя вновь. Однако все эти тонкости не мешают *ndc* выполнять команды *start* и *restart*.

Вот что делают перечисленные команды:

getpid

Отображает текущий идентификатор процесса DNS-сервера.

status

Отображает довольно объемную информацию о состоянии DNS-сервера, включая его версию, уровень отладки, число выполняющихся передач зон, а также информацию о том, включена ли регистрация запросов.

start

Запускает DNS-сервер. Если необходимо передать DNS-серверу *named* определенные аргументы командной строки, их можно указать после команды *start*. Пример: *start -c /usr/local/etc/named.conf*.

stop

Завершение работы DNS-сервера с записью динамических зон в файл данных.

restart

Останов и последующий запуск DNS-сервера. Как и для команды *start*, могут быть указаны аргументы командной строки для *named*.

exec

Останов и последующий запуск DNS-сервера. В отличие от *restart*, *exec* не позволяет передавать аргументы командной строки для *named*; DNS-сервер просто стартует еще раз с теми же аргументами.

reload

Перезагрузка DNS-сервера. Эту команду можно послать первичному мастер-серверу DNS после изменения файла его настройки либо одного или нескольких файлов данных зон. Для дополнительных DNS-серверов версии 4.9 и более поздних эта команда является предписанием обновить хранимые зоны, если информация в них не является актуальной. В качестве аргументов *reload* можно указывать доменные имена; в этом случае, происходит перезагрузка только для указанных зон.

reconfig [-noexpired]

Предписывает DNS-серверу проверить файл настройки на предмет обнаружения создания новых или удаления старых зон. Эту команду можно послать DNS-серверу, если были удалены или добавлены зоны, но данные существующих зон не изменились. Указание ключа *-noexpired* предписывает DNS-серверу не реагировать сообщениями об ошибках, связанных с устареванием зональных данных. Такая возможность очень удобна, если DNS-сервер является авторитативным для тысяч зон, и необходимо избежать получения излишних сообщений об устаревании зон.

dumpdb

Создает дамп внутренней базы данных DNS-сервера в файле *named_dump.db* – в каталоге */usr/tmp* (BIND 4) либо в текущем каталоге DNS-сервера (BIND 8).

stats

Записывает статистику работы DNS-сервера в конец файла *named.stats*, который расположен в каталоге */usr/tmp* (BIND 4) либо в текущем каталоге DNS-сервера (BIND 8).

trace [level]

Добавляет отладочную информацию в конец файла *named.run*, расположенного в каталоге */usr/tmp* (BIND 4) либо в текущем каталоге DNS-сервера (BIND 8). Степень подробности отладочного вывода контролируется увеличением уровня отладки (*level*). Информация о данных, предоставляемых на каждом из уровней, содержится в главе 13 «Чтение отладочного вывода BIND».

notrace

Выключает отладку.

querylog (или qrylog)

Включает или выключает регистрацию всех запросов в log-файле *syslog*. Регистрация запросов происходит с приоритетом LOG_INFO. Сервер *named* должен быть собран с ключом QRYLOG (по умолчанию QRYLOG задан). Механизм регистрации запросов появился в BIND 4.9.

quit

Завершение сеанса управления.

rndc и controls (BIND 9)

В BIND 9 оператор *controls* точно так же используется для определения способа приема управляющих сообщений. Синтаксис оператора отличается незначительно – допустимо только одно предписание *inet*. (BIND 9.1.0 не поддерживает Unix-сокеты для управляющего канала, и, исходя из заявлений консорциума ISC, Unix-сокеты в BIND 9 никогда не появятся.)

В BIND 9 можно не указывать номер порта, и в этом случае сервер будет прослушивать стандартный порт 953. Необходимо включать в предписание раздел *keys*:

```
controls {
       inet * allow { any; } keys { "rndc-key"; };
};
```

Таким образом определяется криптографический ключ, с помощью которого должны идентифицировать себя пользователи *rndc* перед отправкой DNS-серверу управляющих сообщений. Если раздел *keys* отсутствует, после запуска DNS-сервера в log-файле можно обнаружить следующее сообщение:

```
Jan 13 18:22:03 terminator named[13964]: type 'inet' control channel
has no 'keys' clause; control channel will be disabled
```

Ключ или ключи, перечисленные в разделе *keys*, должны быть предварительно определены с помощью оператора *key*:

```
key "rndc-key" {
```

```
        algorithm hmac-md5;
        secret "Zm9vCg==";
};
```

Оператор *key* может присутствовать непосредственно в файле *named.conf*, но если *named.conf* доступен для чтения не только владельцу (группе), будет безопаснее поместить определение ключа в отдельный файл, который имеет более ограниченные права доступа, и включать этот файл в *named.conf* следующим образом:

```
include "/etc/rndc.key";
```

В настоящее время поддерживается только алгоритм HMAC-MD5, то есть механизм идентификации на основе быстрого MD5-алгоритма создания устойчивого хеш-значения.[1] Ключ является общим для *named* и пользователей *rndc* паролем в кодировке Base 64. Ключ можно сгенерировать с помощью таких программ как *mmencode* и *dnssec-keygen* из пакета BIND. Подробности об их применении приводятся в главе 11 «Безопасность».

К примеру, чтобы получить строку *foobarbaz* в кодировке Base 64, можно воспользоваться программой *mmencode*:

```
% mmencode
foobarbaz
Zm9vYmFyYmF6
```

Чтобы пользоваться программой *rndc*, следует создать файл *rndc.conf* – источник информации о ключах и серверах имен для программы *rndc*. *rndc.conf* обычно проживает в каталоге */etc*. Вот пример простого файла *rndc.conf*:

```
options {
        default-server localhost;
        default-key "rndc-key";
};

key "rndc-key" {
        algorithm hmac-md5;
        secret "Zm9vCg==";
};
```

Синтаксис этого файла очень похож на синтаксис файла *named.conf*. В операторе *options* определяется DNS-сервер, которому по умолчанию посылаются управляющие сообщения (эту настройку можно изменить из командной строки), а также имя ключа, который по умолчанию предоставляется удаленным DNS-серверам (имя ключа также можно изменить из командной строки).

[1] Более подробная информация по алгоритму HMAC-MD5 содержится в документах RFC 2085 и 2104.

Синтаксис оператора *key* идентичен используемому в файле *named.conf*, который описан ранее. Имя ключа в *rndc.conf*, а также связанный с именем секрет должны совпадать с определением ключа в *named.conf*.

В случае, когда ключи – по сути дела, пароли – хранятся в файлах *rndc.conf* и *named.conf*, следует убедиться, что ни один из этих файлов не может быть прочитан пользователями, которые не должны управлять DNS-сервером.

Если *rndc* применяется для управления единственным DNS-сервером, настройка не представляет сложностей. Ключ идентификации определяется идентичными операторами *key* в файлах *named.conf* и *rndc.conf*. Затем DNS-сервер указывается в качестве используемого по умолчанию в разделе *default-server* оператора *options*, в файле *rndc.conf*, а ключ – в качестве ключа, используемого по умолчанию, в *default-key*. После этого следует выполнить *rndc* следующим образом:

```
% rndc reload
```

Если существует необходимость управлять набором серверов, можно связать каждый из них с отдельным ключом. Ключи следует определить в отдельных операторах *key*, а затем связать каждый из ключей с сервером с помощью набора операторов *server*:

```
server localhost {
 key "rndc-key";
};

server wormhole.movie.edu {
 key "wormhole-key";
};
```

После этого можно выполнять *rndc*, указывая в качестве аргумента ключа *-s* имя DNS-сервера, с которым следует работать:

```
% rndc -s wormhole.movie.edu reload
```

Если ключ не был связан с определенным DNS-сервером, используемый ключ можно задать в командной строке, используя ключ *-y* программы *rndc*:

```
% rndc -s wormhole.movie.edu -y rndc-wormhole reload
```

И наконец, если DNS-сервер ожидает получения управляющих сообщений через нестандартный порт (то есть не через порт с номером 953), следует использовать ключ командной строки *-p* для указания порта:

```
% rndc -s terminator.movie.edu -p 54 reload
```

А вот и плохая новость: в BIND 9.0.0 программа *rndc* поддерживает только команду *reload*, но не перезагрузку отдельных зон, которая не

поддерживается вплоть до версии 9.1.0. В BIND 9.1.0 поддерживаются не все существующие в BIND 8 команды; поддерживаются – *reload*, *stop*, *stats*, *querylog* и *dumpdb*, а также новые команды *refresh* и *halt*.

refresh

Принудительное выполнение служебных операций для вторичной зоны.

halt

Прекращение работы DNS-сервера без записи в файлах журнала информации о незавершенных обновлениях файлов зон.

Сигналы

В стародавние времена для управления DNS-серверами существовали только сигналы. Живущим в тех временах (применяющим BIND версии более ранней, чем 8.2) придется использовать сигналы. Мы приведем перечень сигналов, которые могут посылаться DNS-серверу, и для каждого укажем эквивалентную команду *ndc*. Если используется реализация *ndc* на языке командного интерпретатора (из пакета BIND версий с 4.9 по 8.1.2), можно не обращать внимания на названия сигналов, поскольку *ndc* самостоятельно преобразует команды в соответствующие сигналы. Не рекомендуется использовать *ndc* из пакета BIND 4 для работы с DNS-сервером BIND 8, поскольку между версиями произошло изменение сигнала для получения статистики.

Команда	**Сигнал**
reload	HUP
dumpdb	INT
stats	ABRT (BIND 4) или ILL (BIND8)
trace	USR1
notrace	USR2
querylog	WINCH
stop (BIND 8)	TERM

Переключение регистрации запросов в более старой версии *ndc* можно произвести с помощью команды:

```
# ndc querylog
```

точно так же, как и в более новых версиях. Но в действительности, в более старых версиях *ndc* находит идентификатор процесса *named* и посылает процессу сигнал WINCH.

В отсутствие *ndc* придется производить те же действия вручную: выяснить идентификатор процесса *named* и послать процессу соответству-

ющий сигнал. DNS-сервер BIND записывает идентификатор процесса в *pid-файл*, так что процесс охоты значительно сокращается – совершенно не обязательно применять *ps*. Наиболее распространенное имя *pid-файла* – */var/run/named.pid*. В некоторых системах *pid-файл* называется */etc/named.pid*. Чтобы выяснить, в каком каталоге системы проживает файл *named.pid*, сверьтесь со страницами руководства по *named*. Поскольку номер процесса DNS-сервера – единственное, что записано в *pid-файле*, посылка сигнала HUP может быть произведена вот так:

```
# kill -HUP `cat /var/run/named.pid`
```

Если вы не можете найти *pid-файл*, идентификатор процесса всегда можно узнать с помощью команды *ps*. На BSD-системе выполните команду:

```
% ps -ax | grep named
```

На системе типа SYS V:

```
% ps -ef | grep named
```

При использовании *ps* может выясниться, что процессов с именем *named* больше одного, поскольку DNS-сервер BIND порождает процессы для выполнения передачи зоны. При передаче зоны сервер, получающий данные, то есть вторичный, может создать специальный процесс; то же справедливо и для сервера, передающего данные. Сейчас мы сделаем отступление и расскажем, зачем используются порождаемые процессы.

Вторичные DNS-серверы BIND 4 и BIND 8 в целях выполнения передачи зоны порождают новые процессы. Это позволяет вторичному DNS-серверу продолжать отвечать на запросы в процессе получения зоны от основного сервера. Когда зона сохранена в локальном файле, вторичный DNS-сервер производит загрузку новых данных. Порождаемые процессы решили проблему, которая существовала в BIND версий более ранних, чем 4.8.3, – вторичные DNS-серверы не отвечали на запросы в процессе получения зоны. Это было особенно неприятно в случае DNS-серверов с большим числом зон либо с зонами большого объема: они не отвечали на запросы в течение больших промежутков времени.

В BIND 9 реализована новая программная архитектура, поэтому DNS-серверу нет необходимости порождать еще один процесс, чтобы продолжать отвечать на запросы при получении зоны. DNS-сервер способен передавать зону и одновременно отвечать на запросы.

Основные серверы в BIND 8 и 9 *не* порождают процессы для передачи зон вторичным DNS-серверам. Вместо этого основной сервер производит передачу зоны параллельно с ответами на запросы. Если мастер-сервер загружает новую копию данных зоны из файла в процессе передачи этой же зоны, передача прерывается. Вторичному DNS-серверу

придется сделать еще одну попытку получить зону после того, как она будет загружена мастером.

Мастер-сервер DNS BIND версии 4 порождает процесс для передачи зоны вторичному DNS-серверу. Это создает дополнительную нагрузку на узел, на котором работает мастер-сервер, особенно в случае крупных зон и параллельной передачи нескольких зон сразу.

Если вывод программы *ps*[1] содержит информацию о нескольких серверах, то родительский процесс должен быть довольно легко отличим от порожденных. Порожденный DNS-сервер, запущенный вторичным сервером в целях получения копии зоны, называется *named-xfer*, а не *named*:

```
root  548 547  0 22:03:17 ?     0:00 named-xfer -z movie.edu
      -f /usr/tmp/NsTmp0 -s 0 -P 53 192.249.249.3
```

Порожденный DNS-сервер, запущенный мастером, изменяет аргументы командной строки, показывая какому вторичному DNS-серверу передается зона:

```
root 1137 1122 6 22:03:18 ?     0:00 /etc/named -zone XFR
     to [192.249.249.1]
```

Может встретиться версия *named*, в которой не происходит изменения аргументов командной строки, но отношения между различными процессами *named* можно понять, изучив их идентификаторы. Все порожденные процессы в качестве идентификатора родительского процесса будут иметь идентификатор процесса DNS-сервера. Это может казаться очевидным, но сигналы следует посылать только *родительскому* процессу DNS-сервера. Порожденные процессы заканчивают работу после завершения передачи зон.

Обновление файлов данных зон

В сети постоянно что-то меняется – появляются новые рабочие станции, старые машины и реликты отправляются на пенсию либо распродаются, узлы перемещаются в другие сети. Любое такое изменение связано с изменением файлов данных зоны. Как следует вносить изменения – вручную или быть тряпкой и облегчить себе жизнь с помощью специального инструмента?

Сначала мы обсудим, как следует вносить изменения самостоятельно. Затем мы поговорим о вспомогательном инструменте – программе *h2n*. Вообще говоря, мы рекомендуем пользоваться каким-либо специальным инструментом для создания файлов данных зоны, а про тряп-

[1] На BSD-системах следует использовать *ps* с ключом *–axww*, чтобы увидеть строку полностью: в ней более 80 символов. – *Примеч. науч. ред.*

ки просто шутим. По крайней мере, можно воспользоваться инструментом, который будет увеличивать порядковый номер зоны. Синтаксис файла данных зоны позволяет делать многочисленные ошибки. Не спасает и то, что адресные записи и записи указателей содержатся в разных файлах, которые должны быть согласованы друг с другом. Но даже при использовании инструмента обязательно надо понимать, что происходит при обновлении файлов, так что мы начнем с первого варианта работы – вручную.

Добавление и удаление узлов

После первоначального создания файлов данных зоны должно быть достаточно очевидно, что именно следует изменять при добавлении нового узла. Мы рассмотрим этот процесс на тот случай, если вам достались уже готовые файлы данных или просто нравится, когда все разложено по полочкам. Изменения следует вносить в файлы данных, которые хранятся первичным мастер-сервером зоны. Если изменить файлы резервных копий *вторичного* DNS-сервера, то данные на вторичном сервере изменятся, но будут перезаписаны при следующей передаче зоны.

1. Обновите порядковый номер в файле *db.DOMAIN*. Вероятнее всего, порядковый номер расположен в начале файла, так что можно легко сделать это сразу, чтобы позже не забыть.
2. Добавьте все записи типа A (адресные), CNAME (псевдонимы) и MX (почтовых ретрансляторов) для нового узла в файл *db.DOMAIN*. При появлении в нашей сети узла *cujo* мы добавили следующие RR-записи в файл *db.movie.edu*:

```
cujo  IN  A  192.253.253.5  ; Интернет-адрес cujo
     IN MX  10 cujo        ; отправлять почту напрямую cujo, если возможно
      IN MX  20 terminator  ; в противном случае -
                            ; через наш почтовый концентратор
```

3. Обновите порядковые номера и добавьте PTR-записи для всех файлов *db.ADDR*, в которых есть адрес узла. *cujo* имеет всего один адрес, в сети 192.253.253/24; поэтому мы добавили следующую PTR-запись в файл *db.192.253.253*:

```
5  IN PTR cujo.movie.edu.
```

4. Перезагрузите первичный мастер-сервер DNS, чтобы обеспечить загрузку новых данных:

```
# ndc reload
```

Если речь идет о шикарном сервере BIND версии 8.2 или более поздней, можно перезагрузить только зоны, которых коснулись изменения:

```
# ndc reload movie.edu 253.253.192.in-addr.arpa
```

Первичный мастер-сервер DNS загрузит новые данные, дополнительные DNS-серверы загрузят эти новые данные в пределах временного интервала обновления, определенного в SOA-записи.

Нетерпеливые пользователи не желают ждать, когда дополнительные DNS-серверы получат новые данные, этим пользователям информация нужна немедленно. (Вы вздрагиваете или понимающе киваете, читая это?) Можно ли заставить вторичный DNS-сервер произвести синхронизацию? Если речь идет о первичных и вторичных серверах версии 8 или 9, вторичные серверы очень быстро получают новые данные, поскольку первичный мастер-сервер DNS уведомляет вторичные о произведенных изменениях в пределах пятнадцати минут после загрузки новых данных. Если используется DNS-сервер версии 4.9 или более поздней, можно перезагрузить его точно так же, как и первичный мастер. Перезагрузка приводит к обновлению всех хранимых зон, для которых сервер является вторичным. Если речь идет о версии 4.8.3 или более ранней, следует удалить все резервные копии файлов хранимых зон (либо только файл зон, для которых необходимо произвести принудительное обновление), принудительно завершить вторичный сервер, а затем заново запустить его. Поскольку резервные копии файлов отсутствуют, вторичный сервер вынужден немедленно заняться получением новых копий зон.

Чтобы удалить узел, следует удалить RR-записи для этого узла из файла *db.DOMAIN*, а также из каждого файла *db.ADDR*. Необходимо увеличить порядковый номер каждой зоны, которой коснулось изменение, а затем перезагрузить первичный мастер-сервер DNS.

Порядковые номера записи SOA

Каждый из файлов данных зоны содержит порядковый номер. При каждом изменении данных в таком файле порядковый номер должен увеличиваться. В противном случае, дополнительные DNS-серверы не будут получать обновленные данные.

Увеличить порядковый номер легко. Допустим, в исходном файле данных зоны присутствовала следующая SOA-запись:

```
movie.edu. IN SOA terminator.movie.edu. al.robocop.movie.edu. (
                                100     ; Порядковый номер
                                3h      ; Обновление
                                1h      ; Повторение
                                1w      ; Устаревание
                                1h )    ; Отрицательное TTL
```

SOA-запись обновленного файла будет выглядеть так:

```
movie.edu. IN SOA terminator.movie.edu. al.robocop.movie.edu. (
                                101     ; Порядковый номер
                                3h      ; Обновление
                                1h      ; Повторение
```

```
                    1w      ; Устаревание
                    1h )    ; Отрицательное TTL
```

Это примитивное изменение является ключом к распространению обновленных данных зоны на дополнительные DNS-серверы. Наиболее распространенная ошибка про обновлении зоны – сохранение старого порядкового номера. В первые несколько раз, внося изменения в файл данных зоны, администратор не забывает обновлять порядковый номер, поскольку это непривычное действие, и оно способствует концентрации внимания. Когда изменение файла данных становится рутинной операцией, администратор вносит небольшие изменения «на скорую руку», забывая обновить порядковый номер... и все дополнительные DNS-серверы продолжают работать со старыми копиями зон. Именно поэтому следует использовать инструмент, который будет заниматься обновлением порядкового номера! Это может быть программа *h2n* или программа, которую администратор напишет самостоятельно, – в любом случае, идея использования специального инструмента заслуживает внимания.

BIND позволяет использовать десятичные дроби (вроде числа 1,1) в качестве порядковых номеров, но мы рекомендуем использовать только целочисленные значения. BIND версии 4 работает с порядковыми номерами в виде десятичных дробей следующим образом: если в номере присутствует десятичная запятая, BIND умножает число слева от запятой на 1000. Затем к результату – простым слиянием строк – добавляется число справа от запятой. Так, число 1,1 преобразуется во внутренний формат с результатом 10001, а 1,10 представляется в виде 100010. Это приводит к появлению определенных аномалий; к примеру, число 1,1 «больше», чем число 2, а 1,10 «больше», чем 2,1. Поскольку логика в этом случае практически полностью отсутствует, лучше всего придерживаться целочисленных порядковых номеров.

Существует несколько наиболее правильных способов работы с целочисленными порядковыми номерами. Самый очевидный связан с использованием обычного счетчика: порядковый номер при каждом изменении файла увеличивается на единицу. Второй способ – использовать порядковые номера, основанные на датах. К примеру, можно использовать число из восьми цифр в формате ГГГГММДД. Допустим, сегодня 15 января 1997 года. При использовании такого формата мы получим порядковый номер 19970115. Но такая схема позволяет производить лишь одно обновление в день, чего может быть недостаточно. Добавим к номеру еще две цифры, чтобы определять номер обновления за текущий день. Первый порядковый номер для 15 января 1997 года – 1997011500. Следующее обновление в тот же день приведет к замене порядкового номера на 1997011501. Таким образом можно вносить обновления до ста раз в день. Помимо этого, предлагаемая схема оставляет возможность определить, когда в последний раз был увеличен порядковый номер. Вызов программы *h2n* с ключом *-y* приводит к

созданию порядкового номера на основе текущей даты. В любом случае порядковый номер не должен превышать максимального значения 32-битного целого числа.

Цикл порядкового номера

Что делать в случае, когда порядковый номер одной из зон случайно становится чрезмерно большим, и появляется необходимость уменьшить его до более разумного значения? Существует способ, который работает для всех версий BIND, способ, который работает для версии 4.8.1 и более поздних, а также способ, который работает для версии 4.9 и более поздних.

Способ, работающий для всех версий BIND: заставить дополнительные DNS-серверы забыть о старом порядковом номере. Затем можно начинать нумерацию с единицы (или с другого удобного числа). Собственно способ: во-первых, следует изменить порядковый номер на первичном мастер-сервере имен и перезапустить этот сервер. Первичный мастер-сервер DNS уже работает с новым порядковым номером. Соединитесь с одним из узлов, на которых работают дополнительные DNS-серверы, и принудительно завершите процесс *named* командой *ndc stop*. Удалите резервные копии файлов данных зоны (к примеру, *rm bak.movie.edu bak.192.249.249 bak.192.253.253*) и снова запустите вторичный сервер. Резервные копии были удалены, поэтому вторичный сервер обязан загрузить новую версию зональных данных – получив и новые порядковые номера. Эту процедуру следует повторить для каждого DNS-сервера. Если какой-либо из дополнительных серверов вам не подвластен, придется связаться с администратором этого сервера и попросить его произвести описанную процедуру.

Если все дополнительные DNS-серверы используют версию BIND более позднюю, чем 4.8.1 (умоляем читателей не использовать версию 4.8.1), но более раннюю, чем BIND 9, можно задать специальный порядковый номер – нулевой. Если установить нулевой порядковый номер для зоны, это обяжет каждый из дополнительных DNS-серверов произвести получение зоны при следующей проверке. По сути дела, передача зоны будет происходить *каждый* раз при проверке актуальности хранимых данных, поэтому после синхронизации дополнительных серверов по нулевому порядковому номеру следует не забыть увеличить порядковый номер на основном сервере. При этом существует предел того, насколько можно увеличить порядковый номер. Читайте дальше.

Еще один способ исправить ситуацию с порядковым номером (вторичные серверы версии 4.9 и более поздних) довольно прост для понимания, если сначала освоить вводный материал. Порядковый номер в DNS – 32-битное положительное целое число из интервала от 0 до 4 294 967 295. Для работы с порядковыми номерами используется *непрерывное арифметическое пространство*, то есть для любого поряд-

кового номера половина чисел пространства (2 147 483 647 чисел) меньше, чем это число, а вторая половина – больше.

Рассмотрим на конкретном примере. Предположим, порядковый номер представлен числом 5. Порядковые номера с 6 по (5 + 2 147 483 647) – больше, чем порядковый номер 5, а порядковые номера с (5 + 2 147 483 649) по 4 – меньше. Обратите внимание, что после достижения порядкового номера 4 294 967 295 совершается переход в начало пространства – до номера 4. Кроме того, мы не рассматриваем номер (5 + 2 147 483 648), поскольку он отстоит от исходного номера ровно на половину пространства номеров и может быть больше или меньше 5, в зависимости от реализации. Лучше всего этот номер просто не использовать.

Вернемся к исходной проблеме. Допустим, порядковый номер зоны – 25 000, и мы хотим продолжить нумерацию с цифры 1. Можно преодолеть оставшееся пространство номеров в два шага. Во-первых, следует увеличить порядковый номер до возможного максимума (25 000 + 2 147 483 647 = 2 147 508 647). Если прибавляемое число больше, чем 4 294 967 295 (максимальное значения 32-битного числа), получить новый номер в начале адресного пространства можно вычитанием из этого числа 4 294 967 296. После изменения порядкового номера, следует дождаться, когда все дополнительные серверы синхронизируют хранимые копии зоны. Во-вторых, следует изменить порядковый номер зоны на желаемое значение (1), которое теперь *больше*, чем текущий порядковый номер (2 147 508 647). Вот и все, осталось только дождаться, когда дополнительные серверы вновь произведут синхронизацию с основным сервером!

Дополнительные записи в файле данных зоны

По прошествии некоторого времени после начала работы DNS-сервера у администратора может возникнуть желание добавить данные, которые облегчат сопровождение зоны. Вас никогда не озадачивали вопросом, *где* находится тот или иной хост? Администратор может даже не помнить, что это за хост. В наше время администраторам приходится пасти постоянно растущие стада хостов, что не очень способствует запоминанию подобной информации. Можно призвать на помощь DNS-сервер. Если один из хостов начинает капризничать, и это заметно со стороны, DNS-сервер позволит заметившему странности связаться с администратором.

До сих пор мы рассматривали только записи типов SOA, NS, A, CNAME, PTR и MX. Эти записи жизненно необходимы для рутинной работы DNS, DNS-серверов и приложений, запрашивающих данные этого типа. Но в системе DNS определено намного больше типов данных. Наиболее полезными из прочих типов записей являются записи TXT и RP; их можно использовать для определения местоположения и ответственного лица для каждого из узлов. Перечень широко (и не очень

широко) применяемых RR-записей содержится в приложении А «Формат сообщений DNS и RR-записей».

Общая текстовая информация

TXT – это сокращение TeXT (текст). TXT-записи представляют собой списки строк, каждая из которых не может быть длиннее 255 символов. В версиях BIND, более ранних, чем 4.8.3, поддержка TXT-записей не реализована. В версии 4 BIND ограничивает TXT-запись файла данных зоны одной строкой размером почти в 2 килобайта данных.

TXT-записи могут использоваться в любых целях; к примеру, для указания местоположения хоста:

```
cujo  IN  TXT  "Location: machine room dog house"
```

BIND 8 и 9 также ограничивают размер TXT-записи пределом в два килобайта, но позволяют задавать TXT-запись в несколько строк:

```
cujo  IN  TXT  "Location:" "machine room dog house"
```

Ответственное лицо

Администраторы доменов находятся в особых отношениях с записью типа RP (Responsible Person, ответственное лицо). RP-запись может быть связана с любым доменным именем, внутренним или расположенным на конце ветви дерева имен, она определяет ответственное лицо для узла или зоны. Это позволяет, к примеру, найти злодея, бомбардирующего запросами службу доменных имен. А также облегчает людям задачу поиска администратора, один из узлов которого ведет себя неподобающе.

Запись содержит два поля данных: адрес электронной почты в формате доменного имени и доменное имя, связанное с дополнительной информацией о контактном лице. Адрес электронной почты записывается в том же формате, что и в SOA-записях: символ «@» заменяется точкой. Второе поле содержит доменное имя, для которого должна существовать TXT-запись. В свою очередь TXT-запись содержит информацию о контактном лице (к примеру, полное имя и номер телефона) – в произвольном формате. Если значение одного из полей отсутствует, следует поместить в него указатель на корневой домен («.»).

Вот примеры RP-записей и связанных с ними TXT-записей:

```
robocop      IN  RP   root.movie.edu.  hotline.movie.edu.
             IN  RP   richard.movie.edu.  rb.movie.edu.
hotline      IN  TXT  "Movie U. Network Hotline, (415) 555-4111"
rb           IN  TXT  "Richard Boisclair, (415) 555-9612"
```

Обратите внимание, что в TXT-записях для *root.movie.edu* и *richard.movie.edu* нет необходимости, поскольку речь идет о доменном имени, кодирующем адрес электронной почты.

RP-записи не существовали в момент разработки BIND 4.8.3, но поддерживаются уже в BIND 4.9. Прежде чем начать использовать RP-записи, уточните в документации вашей версии DNS-сервера, реализована ли поддержка этого типа данных.

Создание файлов данных зоны из таблицы узлов

Как читатели могли видеть в главе 4 «Установка BIND», мы определили процесс для преобразования таблицы узлов в зональные данные. Для автоматизации этого процесса мы разработали на языке Perl программу, которая называется *h2n*.[1] Использование специального инструмента для создания данных имеет одно большое преимущество: в файлах данных зоны не будет ошибок и несоответствий – разумеется, если программа *h2n* работает правильно! Довольно часто бывает, что для узла создана адресная запись, но не создана соответствующая PTR-запись, или наоборот. Поскольку эти записи хранятся в различных файлах, ошибиться очень легко.

Что делает *h2n*? На основе существующего файла */etc/hosts* и нескольких ключей командной строки *h2n* создает файлы данных для зон. Системный администратор следит за актуальностью информации в таблице узлов. И каждый раз, внося изменения в таблицу узлов, вызывает *h2n*. *h2n* создает каждый файл данных заново, увеличивая значение порядкового номера зоны. Программу можно выполнять вручную либо ежедневно с помощью демона *cron*. Используя *h2n*, администратор может больше не беспокоиться об увеличении порядковых номеров зон.

Во-первых, *h2n* требуется знать доменное имя зоны прямого отображения и номера сетей. (*h2n* самостоятельно вычислит имена зон обратного отображения, исходя из номеров сетей.) Эти имена легко преобразуются в имена файлов данных зон: данные зоны *movie.edu* записываются в файл *db.movie*, а данные для сети 192.249.249/24 – в файл *db.192.249.249*. Доменное имя зоны прямого отображения и номер сети указываются в качестве аргументов ключей *-d* и *-n*:

-d доменное имя

Доменное имя зоны прямого отображения.

-n номер сети

Номер сети. В случае создания файлов для нескольких сетей можно использовать несколько ключей *-n* в командной строке. Из номера сети следует исключать последние нули и маску сети.

Команда *h2n* требует использования ключа *-d* и минимум одного ключа *-n*; значения по умолчанию для аргументов этих ключей отсутству-

[1] Если вы забыли, где можно взять *h2n*, обратитесь к разделу «Примеры программ» в предисловии.

ют. К примеру, чтобы создать файл данных для зоны *movie.edu*, состоящей из двух сетей, выполним команду:

```
% h2n -d movie.edu -n 192.249.249 -n 192.253.253
```

Существуют следующие дополнительные ключи:

-s сервер

DNS-серверы для NS-записей. Как и для ключа *-n*, допускается использование нескольких ключей *-s* в случае, если необходимо создать записи для нескольких первичных или вторичных DNS-серверов. DNS-сервер версии 8 или 9 при изменении зоны посылает NOTIFY-уведомление, включающее этот список. По умолчанию используется имя узла, на котором выполняется *h2n*.

-h узел

Узел, указываемый в поле MNAME SOA-записи. *узел* должен являться первичным мастер-сервером DNS, что обеспечивает корректное функционирование механизма NOTIFY. По умолчанию используется имя узла, на котором выполняется *h2n*.

-u пользователь

Адрес электронной почты лица, отвечающего за сопровождение данных зоны. По умолчанию – регистрационная запись *root* на узле, с которого запущена программа *h2n*.

-o разное

Прочие значения SOA-записи, кроме порядкового номера, – в виде списка, элементы которого разделяются двоеточием. Значение по умолчанию: 10800:3600:604800:86400.

-f файл

Чтение ключей *h2n* из указанного файла, а не из командной строки. Если используется большое количество ключей, имеет смысл хранить их в отдельном файле.

-v 4|8

Создавать файлы настройки для BIND 4 или 8; по умолчанию создаются файлы для версии 4. Поскольку формат файла настройки BIND 9 практически не отличается от формата BIND 8, для DNS-серверов BIND 9 можно использовать ключ *-v 8*.

-y

Создать порядковый номер на основе даты.

Пример использования всех описанных ключей:

```
% h2n -f opts
```

Содержимое файла *opts*:

```
-d movie.edu
```

```
-n 192.249.249
-n 192.253.253
-s terminator.movie.edu
-s wormhole
-u al
-h terminator
-o 10800:3600:604800:86400
-v 8
-y
```

Если в качестве аргумента ключа должно выступать имя узла, можно указать полное доменное имя (например, *terminator.movie.edu*) либо просто имя узла (например, *terminator*). В последнем случае *h2n* создает полное доменное имя, добавляя к имени узла аргумент ключа *-d*. (Если существует необходимость в точке в конце имени, *h2n* также добавляет ее самостоятельно.)

Мы перечислили далеко не все существующие ключи *h2n*. Полный перечень ключей с описаниями представлен на страницах руководства этой команды.

Разумеется, некоторые виды RR-записей не могут быть созданы на основе файла */etc/hosts*, поскольку он не содержит необходимой для этого информации. Вполне возможно, что эти записи надо будет добавлять вручную. Но раз *h2n* производит перезапись файлов данных, эта информация может быть потеряна.

Поэтому в *h2n* существует «черный ход»: возможность автоматически добавлять данные такого рода. Эти особые записи следует поместить в файл с именем вида *spcl.DOMAIN*, где *DOMAIN* – первая метка доменного имени зоны. При обнаружении такого файла *h2n* «включает» его с помощью строки следующего вида:

```
$INCLUDE spcl.DOMAIN
```

в конце файла *db.DOMAIN*. (Директива $INCLUDE будет описана позже в этой главе.) Так, администратор зоны *movie.edu* может создать дополнительные MX-записи в файле *spcl.movie*, что позволит пользователям посылать почту напрямую в *movie.edu*, а не конкретным узлам зоны. Обнаружив этот файл, *h2n* добавит строку:

```
$INCLUDE spcl.movie
```

к концу файла данных зоны *db.movie*.

Сопровождение файла корневых указателей

Как мы рассказывали в главе 4, файл корневых указателей содержит координаты корневых DNS-серверов, которые могут использоваться локальным DNS-сервером. Этот файл необходимо регулярно обновлять. Корневые DNS-серверы меняются не очень часто, но все-таки ме-

няются. Хорошим тоном будет проверять актуальность файла корневых указателей раз в месяц-два. В главе 4 мы говорили, что файл можно получать FTP-копированием с узла *ftp.rs.internic.net*. И это, видимо, самый лучший способ его обновлять.

Если доступна копия программы *dig*, которая во многом похожа на *nslookup* и входит в состав дистрибутива BIND, можно получить текущий список корневых DNS-серверов, выполнив следующую команду:

```
% dig @a.root-servers.net  .  ns > db.cache
```

Организация файлов

Когда администратор впервые производит настройку зон, организация файлов примитивна – все они проживают в единственном каталоге. Существует один файл настройки и несколько файлов данных зон. Однако со временем обязанности администратора растут. Добавляются сети, а следовательно, и новые зоны *in-addr.arpa*. Возможно даже, происходит делегирование поддоменов. Локальные DNS-серверы используются в качестве резервных для других сетей. Через какое-то время вывод команды *ls* перестает умещаться на экране. Пора производить реорганизацию. В пакете BIND существуют механизмы, облегчающие этот процесс.

DNS-серверы BIND 4.9 и более поздних версий поддерживают оператор файла настройки *include*, который позволяет включать содержимое произвольного файла в текущий файл настройки. Это позволяет разбить очень большой файл настройки на много маленьких частей.

В файлах данных зон (во всех версиях BIND) могут использоваться две директивы[1]: $ORIGIN и $INCLUDE. Директива $ORIGIN позволяет изменять суффикс по умолчанию для файла данных зоны, а директива $INCLUDE реализует включение содержимого другого файла в текущий файл данных зоны. Директивы не являются RR-записями, они предназначены для сопровождения данных DNS. В частности, они облегчают разбиение зоны на поддомены, позволяя хранить данные каждого из поддоменов в отдельном файле.

Увеличение числа каталогов

Одним из способов организации файлов является их хранение в нескольких каталогах. Если DNS-сервер является первичным мастером для нескольких зон площадки (зон как прямого, так и обратного отображения), можно хранить каждый из файлов данных в отдельном каталоге. Еще вариант – все файлы первичного мастер-сервера DNS хра-

[1] На самом деле – три, если считать директиву $TTL, поддержка которой реализована в серверах имен BIND 8.2 и более поздних версий.

нятся в одном каталоге, а все файлы вторичного – в другом. Вот так может выглядеть файл настройки BIND 4 в случае разделения «основных» и «дополнительных» зон:

```
directory /var/named
;
; Эти файлы являются общими для всех зон
;
cache    .                          db.cache
primary  0.0.127.in-addr.arpa       db.127.0.0
;
; Файлы зон контрольного DNS-сервера
;
primary  movie.edu                  primary/db.movie.edu
primary  249.249.192.in-addr.arpa   primary/db.192.249.249
primary  253.253.192.in-addr.arpa   primary/db.192.253.253
;
; Файлы зон вторичного DNS-сервера
;
secondary ora.com                   198.112.208.25 slave/bak.ora.com
secondary 208.112.198.in-addr.arpa 198.112.208.25 slave/bak.198.112.208
```

Те же настройки в формате BIND 8:

```
options { directory "/var/named"; };
//
// Эти файлы являются общими для всех зон
//
zone "." {
        type hint;
        file "db.cache";
};
zone "0.0.127.in-addr.arpa" {
        type master;
        file "db.127.0.0";
};
//
// Файлы зон контрольного DNS-сервера
//
zone "movie.edu" {
        type master;
        file "primary/db.movie.edu";
};
zone "249.249.192.in-addr.arpa" {
        type master;
        file "primary/db.192.249.249";
};
zone "253.253.192.in-addr.arpa" {
        type master;
        file "primary/db.192.253.253";
};
//
```

```
// Файлы зон вторичного DNS-сервера
//
zone "ora.com" {
        type slave;
        file "slave/bak.ora.com";
        masters { 198.112.208.25; };
};
zone "208.112.192.in-addr.arpa" {
        type slave;
        file "slave/bak.198.112.208";
        masters { 198.112.208.25; };
};
```

Существует еще одна вариация такого деления, связанная с разбивкой файла настройки на три: основной файл, файл, который содержит все *primary*-записи, и файл, который содержит все *secondary*-записи. Вот так может выглядеть основной файл настройки для BIND 4:

```
directory /var/named
;
; Эти файлы являются общими для всех зон
;
cache    .                          db.cache
primary  0.0.127.in-addr.arpa       db.127.0.0
;
include  named.boot.primary
include  named.boot.slave
```

Файл *named.boot.primary* (BIND 4):

```
;
; Файлы зон контрольного DNS-сервера
;
primary  movie.edu                  primary/db.movie.edu
primary  249.249.192.in-addr.arpa   primary/db.192.249.249
primary  253.253.192.in-addr.arpa   primary/db.192.253.253
```

Файл *named.boot.slave* (BIND 4):

```
;
; Файлы зон вторичного DNS-сервера
;
secondary ora.com                   198.112.208.25 slave/bak.ora.com
secondary 208.112.198.in-addr.arpa 198.112.208.25 slave/bak.198.112.208
```

Те же файлы в формате BIND 8 и 9:

```
options { directory "/var/named"; };
//
// Эти файлы являются общими для всех зон
//
zone "." {
        type hint;
        file "db.cache";
```

```
};
zone "0.0.127.in-addr.arpa" {
        type master;
        file "db.127.0.0";
};

include "named.conf.primary";
include "named.conf.slave";
```

Файл *named.conf.primary* (BIND 8 и 9):

```
//
// Файлы зон контрольного DNS-сервера
//
zone "movie.edu" {
        type master;
        file "primary/db.movie.edu";
};
zone "249.249.192.in-addr.arpa" {
        type master;
        file "primary/db.192.249.249";
};
zone "253.253.192.in-addr.arpa" {
        type master;
        file "primary/db.192.253.253";
};
```

Файл *named.conf.slave* (BIND 8 и 9):

```
//
// Файлы зон вторичного DNS-сервера
//
zone "ora.com" {
        type slave;
        file "slave/bak.ora.com";
        masters { 198.112.208.25; };
};
zone "208.112.192.in-addr.arpa" {
        type slave;
        file "slave/bak.198.112.208";
        masters { 198.112.208.25; };
};
```

Можно предположить, что организация была бы лучше, если бы файл настройки с инструкциями *primary* располагался в подкаталоге *primary* – после добавления соответствующей инструкции *directory*, предписывающей использовать этот каталог в качестве рабочего, и удаления компоненты *primary/* из всех имен файлов. Аналогичные изменения можно было бы произвести для файла с инструкциями *secondary*. К сожалению, это не будет работать. DNS-серверы BIND 8 и 9 позволяют задавать только один рабочий каталог. DNS-серверы BIND 4 позволяют переопределять рабочий каталог с помощью допол-

нительных инструкций *directory*, но это скорее недосмотр, нежели полезное свойство. При переключении DNS-сервера между различными рабочими каталогами осложняются даже простые вещи – резервные копии файлов данных зон оказываются в последнем из рабочих каталогов DNS-сервера, а при перезагрузке сервер может не найти основной файл настройки, если он отсутствует в каталоге запуска (разумеется, если указано относительное имя файла настройки).

Изменение суффикса по умолчанию в файле данных зоны

В BIND суффиксом по умолчанию для файлов данных зоны является второе поле инструкции *primary* или *secondary* в файле *named.boot* (BIND 4) либо второе поле оператора *zone* в файле *named.conf* (BIND 8 и 9). Суффикс по умолчанию – это доменное имя, автоматически добавляемое ко всем именам, которые не заканчиваются точкой. Суффикс по умолчанию может быть изменен в файле данных зоны с помощью директивы $ORIGIN. В качестве аргумента директивы $ORIGIN должно указываться доменное имя. (Не забывайте последнюю точку, если используете полное доменное имя!) Начиная со строки, следующей за директивой, ко всем именам, которые не заканчиваются точкой, будет добавляться новый суффикс по умолчанию. Если зона (скажем, *movie.edu*) содержит несколько поддоменов, директиву $ORIGIN можно использовать для сброса суффикса по умолчанию и упрощения файла данных зоны. Пример:

```
$ORIGIN classics.movie.edu.
maltese       IN  A  192.253.253.100
casablanca    IN  A  192.253.253.101

$ORIGIN comedy.movie.edu.
mash          IN  A  192.253.253.200
twins         IN  A  192.253.253.201
```

Создание поддоменов мы более подробно изучим в главе 9 «Материнство».

Включение файлов данных

После разделения зоны описанным способом может оказаться, что удобнее хранить записи для каждого поддомена в отдельном файле. Подобное разбиение можно организовать с помощью директивы $INCLUDE:

```
$ORIGIN classics.movie.edu.
$INCLUDE db.classics.movie.edu

$ORIGIN comedy.movie.edu.
$INCLUDE db.comedy.movie.edu
```

Чтобы упростить файл еще больше, можно указывать новый суффикс по умолчанию и включаемый файл в одной строке:

```
$INCLUDE db.classics.movie.edu classics.movie.edu.
$INCLUDE db.comedy.movie.edu   comedy.movie.edu.
```

В этом случае измененный суффикс по умолчанию используется только для конкретного включаемого файла. К примеру, суффикс по умолчанию *comedy.movie.edu* действует только для имен в файле *db.comedy.movie.edu*. После включения файла *db.comedy.movie.edu* суффикс по умолчанию принимает прежнее значение, даже если в файле *db.comedy.movie.edu* также использовалась директива $ORIGIN.

Перемещение системных файлов в BIND 8 и 9

BIND 8 и 9 позволяют изменять имена и места проживания следующих системных файлов: *named.pid*, *named-xfer*, *named_dump.db* и *named.stats*. Большинству администраторов это никогда не понадобится – совершенно необязательно изменять имена файлов только потому, что существует такая возможность.

Если же места проживания файлов, создаваемых DNS-сервером (*named.pid*, *named_dump.db* и *named.stats*), изменяются из соображений безопасности, следует выбирать каталоги, доступные для записи только владельцу. Нам неизвестно об удачных попытках взлома, связанных с записью в эти файлы, но стоит следовать принципу, чтобы лишний раз не подвергаться опасности.

Полное имя файла *named.pid* обычно */var/run/named.pid* или */etc/named.pid*. Причиной изменения стандартного имени этого файла может послужить работа нескольких серверов на одном узле. (Кошмар! Разве это может кому-то понадобиться?) В главе 10 «Дополнительные возможности» приводится пример работы двух DNS-серверов на одном узле. В файле настройки каждого из серверов можно определить уникальное имя для *named.pid*:

```
options { pid-file "server1.pid"; };
```

Файл *named-xfer* обычно называется */usr/sbin/named-xfer* или */etc/named-xfer*. Читатели, наверное, помнят, что *named-xfer* используется вторичными DNS-серверами для получения зон. Причина, по которой стандартное место проживания этого файла может быть изменено, связано со сборкой и испытаниями новой версии пакета BIND в локальном каталоге. Испытуемый *bind* можно настроить на использование локальной версии *named-xfer*:

```
options { named-xfer "/home/rudy/named/named-xfer"; };
```

Поскольку в BIND 9 *named-xfer* не применяется, особой пользы от предписания *named-xfer* в этой версии BIND нет.

DNS-сервер создает файл *named_dump.db* в текущем каталоге (BIND 8 и 9) при получении команды на создание дампа (образа) базы данных. В следующем примере показано, как изменить имя файла дампа:

```
options { dump-file "/home/rudy/named/named_dump.db"; };
```

При получении команды на создание статистики DNS-сервер создает файл *named.stats* в текущем каталоге (BIND 8 или 9.1.0 и более поздние версии). Вот так можно изменить имя файла статистики:

```
options { statistics-file "/home/rudy/named/named.stats"; };
```

Ведение log-файла в BIND 8 и 9

В BIND 4 реализована мощная система ведения log-файла – записи информации в отладочный файл, а также через демон *syslog*. Но в BIND 4 ограничено управление процессом ведения log-файла – можно устанавливать определенный уровень отладки, но это и все. В BIND 8 и 9 реализована та же система ведения log-файла, но обе новые ветви BIND предоставляют возможности управления, которых не было в BIND 4.

Новое управление имеет свою цену – администратор должен многому научиться, прежде чем сможет производить эффективную настройку этой подсистемы. Если времени на эксперименты с ведением log-файла нет, воспользуйтесь стандартными установками и возвращайтесь к этой теме позже. У большинства администраторов не возникает потребности изменять стандартное поведение подсистемы ведения log-файла.

При работе с log-файлом существует два основных понятия: *каналы* и *категории*. Канал определяет, куда записываются данные: в log-файл демона *syslog*, в файл, в стандартный поток ошибок *named* либо в черную дыру. Категория определяет, какого рода информация записывается в log-файл. В исходных текстах BIND большинство заносимых в log-файл сообщений разбивается на категории в соответствии с функциями кода, к которому они относятся. К примеру, сообщение, поступившее от той части BIND, которая обрабатывает динамические обновления, вероятнее всего принадлежит к категории *update*. Полный перечень категорий мы приведем чуть позже.

Информация каждой категории может поступать в один либо несколько каналов. К примеру, запросы (рис. 7.1) записываются в файл, в то время как статистические данные записываются в файл и в log-файл *syslog*.

Каналы позволяют фильтровать сообщения по их важности. Вот перечень существующих приоритетов важности, начиная с наиболее высокого:

```
critical
error
```

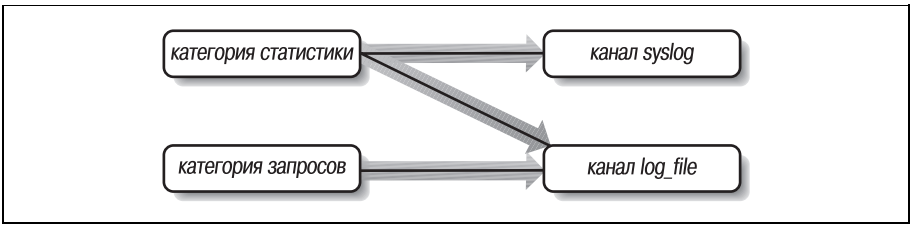

Рис. 7.1. *Запись информации различных категорий в каналы*

```
warning
notice
info
debug [level]
dynamic
```

Первые пять приоритетов (*critical*, *error*, *warning*, *notice* и *info*) – стандартные уровни, знакомые по демону *syslog*. Два других (*debug* и *dynamic*) существуют только в BIND 8 и 9.

debug – отладочное сообщение DNS-сервера, для которого может быть указан уровень. По умолчанию используется уровень 1. Если уровень отладки указан, сообщения этого уровня будут отображаться в случае включения отладки DNS-сервера (например, если задать «debug 3», то при посылке даже единственной команды trace DNS-серверу начнется запись отладочных сообщений третьего уровня). Приоритет *dynamic* позволяет регистрировать сообщения, соответствующие установленному уровню отладки (например, если послать одну команду trace DNS-серверу, будет происходить регистрация сообщений первого уровня отладки. Если послать три команды trace, будут регистрироваться все сообщения для уровней с 1 по 3.) Приоритет по умолчанию – *info*, то есть отладочные сообщения не регистрируются, пока не будет указан конкретный приоритет.

Канал можно настроить на регистрацию как отладочных сообщений, так и сообщений, передаваемых *syslog*, в файле. Обратное неверно: невозможно настроить канал на регистрацию отладочных сообщений и сообщений, предназначенных демону *syslog*, в log-файле *syslog*, поскольку отладочные сообщения не могут быть записаны в этот log-файл.

Сейчас мы настроим пару каналов, чтобы продемонстрировать читателям принципы их работы. Первый канал связывается с демоном *syslog* и *syslog*-потоком *daemon*; в этот канал записываются сообщения приоритета *info* и более высокого. Второй канала связывается с файлом, в который записываются отладочные сообщения любого уровня, а также сообщения для демона *syslog*. Вот оператор *logging* из файла настройки BIND 8 или 9:

```
logging {
  channel my_syslog {
     syslog daemon;
     // Отладочные сообщения не могут посылаться демону syslog, поэтому нет
     // смысла выставлять приоритет debug или dynamic;
     // воспользуемся самым низким уровнем syslog: info.
     severity info;
  };
  channel my_file {
     file "log.msgs";
     // Приоритет dynamic - регистрация всех отладочных сообщений.
     severity dynamic;
  };
};
```

Настроив два канала, мы должны объяснить DNS-серверу, что именно следует записывать в эти каналы. Мы реализуем схему, отображенную на рис. 7.1: статистка записывается в канал *syslog* и в файл, запросы регистрируются в файле. Спецификация *category* является частью оператора *logging*, поэтому мы используем только что созданный оператор:

```
logging {
  channel my_syslog {
     syslog daemon;
     severity info;
  };
  channel my_file {
     file "log.msgs";
     severity dynamic;
  };

  category statistics { my_syslog; my_file; };
  category queries { my_file; };
};
```

Добавив этот оператор *logging* в файл настройки DNS-сервера, мы можем запустить сервер и послать ему несколько запросов. Файл *log.msgs* пуст! (Вообще говоря, если подождать достаточно длительное время, в файле *log.msgs* появится статистика DNS-сервера.) Мы ожидали, что запросы будут регистрироваться. Да, но для этого необходимо сначала включить отладку DNS-сервера:

```
# ndc trace
```

Если теперь послать DNS-серверу несколько запросов, они будут отражены в файле *log.msgs*. Взглянем в рабочий каталог DNS-сервера – в нем появился новый файл, который называется *named.run*. Вся прочая отладочная информация записывается в этот файл. Но нам не нужна вся отладочная информация, только статистика и запросы. Как избавиться от файла *named.run*?

Существует особая категория, о которой мы еще не рассказали: *default*. Если определенная категория не связана хотя бы с одним каналом, BIND записывает сообщения этой категории в тот канал, с которым связана категория *default*. Изменим настройки для категории *default* таким образом, чтобы удалять все сообщения (для этой цели существует канал *null*):

```
logging {
   channel my_syslog {
      syslog daemon;
      severity info;
   };
   channel my_file {
      file "log.msgs";
      severity dynamic;
   };

   category default { null; };
   category statistics { my_syslog; my_file; };
   category queries { my_file; };
};
```

Теперь запустим сервер, включим первый уровень отладки и пошлем несколько запросов. Запросы отражаются в файле *log.msgs*, а файл *named.run* существует, но остается пустым. Отлично! Мы уже начинаем осваиваться с этим механизмом.

Прошло несколько дней. Один из наших коллег заметил, что DNS-сервер записывает в канал *syslog* не так много сообщений, как раньше. Более того, в log-файл *syslog* попадают только сообщения статистики. А сообщения, связанные с передачами зон (за которыми следил коллега), исчезли. Что произошло?

По умолчанию сообщения категории *default* записываются как в канал *syslog*, так и в файл отладки (*named.run*). Связав категорию *default* с каналом *null*, мы отключили и все прочие сообщения канала *syslog*. Следовало использовать такую настройку:

```
category default { my_syslog; };
```

Это приводит к записи *syslog*-сообщений в log-файл канала *syslog*, но не к записи отладочных или *syslog*-сообщений в файл.

Мы уже говорили, что необходимо какое-то время поэкспериментировать с ведением log-файла, чтобы достигнуть желаемого результата. Мы надеемся, что этот пример дал читателям представление о том, с чем можно столкнуться в процессе. А теперь мы переходим к подробностям.

Оператор logging

Ниже приводится синтаксис оператора *logging*. Выглядит он устрашающе, но мы рассмотрим еще несколько примеров и объясним, для чего служит каждое из предписаний:

```
logging {
  [ channel channel_name {
    ( file path_name
       [ versions ( number | unlimited ) ]
       [ size size_spec ]
     | syslog ( kern | user | mail | daemon | auth | syslog | lpr |
                news | uucp | cron | authpriv | ftp |
                local0 | local1 | local2 | local3 |
                local4 | local5 | local6 | local7 )
     | stderr
     | null );

    [ severity ( critical | error | warning | notice |
                 info  | debug [ level ] | dynamic ); ]
    [ print-category yes_or_no; ]
    [ print-severity yes_or_no; ]
    [ print-time yes_or_no; ]
   }; ]

  [ category category_name {
    channel_name; [ channel_name; ... ]
  }; ]
  ...
};
```

А вот стандартные каналы, которые безо всякой настройки, автоматически, создаются DNS-сервером. Эти каналы невозможно переопределить, можно лишь добавить новые:

```
channel default_syslog {
    syslog daemon;        // запись в syslog-поток daemon
    severity info;        // приоритет info, либо более высокий
};

channel default_debug {
    file "named.run";     // запись в файл named.run в рабочем каталоге
    severity dynamic;     // сообщения текущего уровня отладки сервера
};

channel default_stderr {  // запись в поток to stderr
    stderr;               // определение собственного потока ошибок
                          // доступно только BIND 9, а в BIND 8 существует
                          // встроенный канал default_stderr.
    severity info;        // приоритет info, либо более высокий
};

channel null {
```

```
    null;                   // удалять все, что записывается в этот канал
};
```

Если не определить каналы для категорий *default*, *panic*, *packet* и *eventlib*, DNS-сервер связывает их по умолчанию со следующими каналами:

```
logging {
    category default { default_syslog; default_debug; };
    category panic { default_syslog; default_stderr; };
    category packet { default_debug; };
    category eventlib { default_debug; };
};
```

В сервере имен BIND 9 по умолчанию используется оператор *logging* следующего вида:

```
logging {
        category default {
            default_syslog;
            default_debug;
        };
};
```

Как мы уже говорили, сообщения категории *default* записываются как в канал *syslog*, так и в файл отладки (который по умолчанию называется *named.run*). Это значит, что все *syslog*-сообщения приоритета *info* либо более высокого записываются в канал *syslog*, а при включенной отладке *syslog*-сообщения и отладочные сообщения записываются в файл *named.run*. Это более или менее соответствует поведению BIND 4.

Каналы в подробностях

Канал может быть связан с файлом, с демоном *syslog* либо с «черной дырой».

Файловые каналы

Если канал связан с файлом, необходимо указать имя этого файла. Дополнительно можно указать количество версий файла, которые могут существовать одновременно, и то, насколько большим может быть файл.

Если разрешить параллельное существование трех версий, DNS-сервер BIND 8 или 9 будет создавать файлы с именами *file*, *file.0*, *file.1* и *file.2*. После запуска или перезагрузки DNS-сервер переименовывает *file.1* в *file.2*, *file.0* в *file.1*, *file* в *file.0*, а затем начинает записывать в новую копию файла *file*. Неограниченное число версий позволяет одновременно хранить 99 файлов.

Если указан максимальный размер файла, при достижении указанного размера DNS-сервер прекращает запись информации в файл. При

достижении максимального размера не происходит переименование и открытие нового файла, запись просто прекращается. Если максимальный размер файла не задан, он может расти неограниченно.

Вот пример определения файлового канала с применением предписаний *versions* и *size*:

```
logging{
  channel my_file {
     file "log.msgs" versions 3 size 10k;
     severity dynamic;
  };
};
```

Спецификация размера может содержать множитель масштаба (как в приводимом примере). Буквы *K* и *k* позволяют определять размер в килобайтах, *M* и *m* – в мегабайтах, *G* и *g* – в гигабайтах.

Если мы хотим видеть отладочные сообщения, следует указать приоритет *debug* либо *dynamic*. По умолчанию используется приоритет *info*, что позволяет наблюдать только *syslog*-сообщения.

syslog-каналы

Если канал связывается с демоном *syslog*, его сообщения могут быть направлены в один из следующих *syslog*-потоков: *kern*, *user*, *mail*, *daemon*, *auth*, *syslog*, *lpr*, *news*, *uucp*, *cron*, *authpriv*, *ftp*, *local0*, *local1*, *local2*, *local3*, *local4*, *local5*, *local6* или *local7*. По умолчанию это поток *daemon*, его мы и рекомендуем использовать.

Вот пример для канала, связываемого с *syslog* через внутренний *syslog*-поток *local0* вместо *daemon*:

```
logging {
     channel my_syslog {
        syslog local0;         // запись в syslog-канал local0
        severity info;         // приоритет info, либо более высокий
     };
};
```

Канал stderr

Существует заранее определенный канал *default_stderr*, предназначенный для сообщений, которые пишутся в файловый дескриптор *stderr* DNS-сервера. В BIND 8 невозможно связать другие файловые дескрипторы с каналом *stderr*. Но это можно делать в BIND 9.

Канал null

Существует заранее определенный канал *null*, предназначенный для сообщений, которые следует просто выбрасывать.

Каналы и форматирование данных

Механизм ведения log-файла в BIND 8 и 9 также предоставляет возможность до некоторой степени изменять формат сообщений. К сообщениям можно добавлять временную отметку, категорию и информацию о приоритете.

Вот пример отладочного сообщения, содержащего все дополнительные поля:

```
01-Feb-1998 13:19:18.889 config: debug 1: source = db.127.0.0
```

Сообщение принадлежит к категории *config*, уровень приоритета *debug 1*.

А вот пример настройки простого канала, включающей добавление этой информации к сообщениям:

```
logging {
  channel my_file {
     file "log.msgs";
     severity debug;
     print-category yes;
     print-severity yes;
     print-time yes;
  };
};
```

Нет особого смысла добавлять временную отметку к сообщениям, которые записываются в канал *syslog*, поскольку *syslog* делает это самостоятельно.

Категории в подробностях

В BIND 8 и 9 – огромное число категорий, просто огромное! И, к сожалению, все категории разные. Мы перечислим их здесь, чтобы читатели были в курсе. Не советуем пытаться понять, какие категории вас интересуют, гораздо проще настроить DNS-сервер на запись всех сообщений в log-файл, с отметками категорий и приоритетов, а затем выбрать только те категории, которые представляют интерес. Мы расскажем, как это сделать, но сначала – собственно категории.

Категории BIND 8

default

Если для какой-либо категории сообщений не определен канал записи, используется канал, связанный с категорией *default*. В этом смысле *default* является синонимом всех прочих категорий. Однако существуют сообщения, которые не принадлежат к конкретной категории. Поэтому даже в случае, когда для каждой категории канал определен явным образом, следует указать и канал для катего-

рии *default*, чтобы иметь возможность наблюдать сообщения, не привязанные к классификации.

Канал для категории *default* определяется по умолчанию следующим образом:

```
category default { default_syslog; default_debug; };
```

cname

Ошибки CNAME (к примеру, «... has CNAME and other data»).

config

Высокоуровневая обработка файла настройки.

db

Работа с базой данных.

eventlib

Системные события; категория должна привязываться к единственному файловому каналу. По умолчанию:

```
category eventlib { default_debug; };
```

insist

Ошибки, полученные в процессе проверки внутренней целостности.

lame-servers

Обнаружение некорректного делегирования.

load

Сообщения, связанные с загрузкой зон.

maintenance

Периодические служебные события (к примеру, системные запросы).

ncache

События, связанные с кэшированием отрицательных ответов.

notify

Асинхронные уведомления об изменениях зон.

os

Проблемы, связанные с работой операционной системы.

packet

Декодирование получаемых и посылаемых пакетов; категория должна привязываться к единственному файловому каналу. По умолчанию:

```
category packet { default_debug; };
```

panic

Проблемы, приводящие к прекращению работы сервера. Эти проблемы регистрируются в категории *panic* и, одновременно, в своих «родных» категориях. По умолчанию:

```
category panic { default_syslog; default_stderr; };
```

parser

Низкоуровневая обработка файла настройки.

queries

Аналогично регистрации запросов в BIND 4.

response-checks

Некорректные ответные сообщения, ненужная дополнительная информация и т. д.

security

Разрешение/запрещение запросов.

statistics

Периодические отчеты по деятельности сервера.

update

События, связанные с динамическими обновлениями.

xfer-in

События, связанные с получением зон с удаленного DNS-сервера.

xfer-out

События, связанные с передачей зон удаленному DNS-серверу.

Категории BIND 9

default

Как и в BIND 8, категория *default* включает сообщения всех категорий, не связанных явным образом с каналами записи. Но в BIND 9 категория *default* может включать только сообщения BIND, принадлежащие к одной из существующих категорий. Все «прочие» сообщения в случае BIND 9 принадлежат к категории *general*.

general

Категория *general* содержит все сообщения BIND, не классифицированные явным образом.

client

Обработка запросов клиентов.

config

Разбор и обработка файла настройки.

database

Сообщения, связанные с внутренней базой данных BIND, в которой хранятся зональные данные и кэшированные записи.

dnssec

Обработка ответных сообщений с DNSSEC-подписями.

lame-servers

Обнаружение некорректного делегирования (категория вновь добавлена в BIND 9.1.0; до этого подобные сообщения регистрировались в категории *resolver*).

network

Сетевые операции.

notify

Асинхронные уведомления об изменениях в зонах.

queries

Аналогично регистрации запросов в BIND 8 (категория добавлена в BIND 9.1.0).

resolver

Разрешение имен, включая обработку рекурсивных запросов от DNS-клиентов.

security

Разрешение/запрещение запросов.

update

События, связанные с динамическими обновлениями.

xfer-in

События, связанные с получением зон с удаленного DNS-сервера.

xfer-out

События, связанные с передачей зон удаленному DNS-серверу.

Просмотр сообщений всех категорий

Первый набег на механизм ведения log-файла можно совершить следующим образом: настроить DNS-сервер на регистрацию всех сообщений в файле, включая информацию о категориях и приоритетах, а затем выбрать только сообщения, которые нас интересуют.

Мы уже говорили о категориях, настроенных по умолчанию. Вот эти категории для BIND 8:

```
logging {
    category default { default_syslog; default_debug; };
    category panic { default_syslog; default_stderr; };
    category packet { default_debug; };
```

```
    category eventlib { default_debug; };
};
```

И для BIND 9:

```
logging {
    category default { default_syslog; default_debug; };
};
```

По умолчанию категория и приоритет не включаются в сообщения, записываемые в канал *default_debug*. Чтобы увидеть все сообщения, а также их категории и приоритеты, следует настроить каждую из категорий самостоятельно.

Вот оператор *logging* для BIND 8, который позволяет это сделать:

```
logging {
   channel my_file {
     file "log.msgs";
     severity dynamic;
     print-category yes;
     print-severity yes;
   };

   category default  { default_syslog; my_file; };
   category panic    { default_syslog; my_file; };
   category packet   { my_file; };
   category eventlib { my_file; };
   category queries  { my_file; };
};
```

(В операторе *logging* для BIND 9 не будет категорий *panic*, *packet* и *eventlib*.)

Обратите внимание, что сообщения каждой категории записываются дополнительно в канал *my_file*. Помимо этого, мы добавили еще одну категорию, которая отсутствовала в предыдущем операторе *logging*: *queries*. Запросы не отображаются, если не настроить запись для категории *queries*.

Запустите DNS-сервер и включите отладку первого уровня. В файле *log.msgs* появятся сообщения следующего вида:

```
queries: info: XX /192.253.253.4/foo.movie.edu/A
default: debug 1: req: nlookup(foo.movie.edu) id 4 type=1 class=1
default: debug 1: req: found 'foo.movie.edu' as 'foo.movie.edu' (cname=0)
default: debug 1: ns_req: answer -> [192.253.253.4].2338 fd=20 id=4 size=87
```

Определив, какие сообщения представляют интерес, вы можете настроить DNS-сервер на регистрацию только этих сообщений.

Основы благополучия

Существенную роль в процессе сопровождения играет предупреждение неприятностей – до того, как они превратятся в проблемы. Если обнаружить проблему на ранней стадии, ее намного легче будет решить. Как гласит старая пословица, легче болезнь предупредить, чем потом ее лечить.

Речь идет не о разрешении проблем – этому мы посвятим отдельную главу – но скорее о «подготовке к разрешению проблем». Разрешение проблем (лечение) – это действия, выполняемые, когда возникли осложнения, и необходимо определить их причину по наблюдаемым симптомам.

Следующие два раздела посвящены превентивным мерам: периодическому чтению log-файла *syslog* и статистики DNS-сервера BIND с целью пресечения возможных проблем в зародыше. Условимся считать это регулярным медосмотром DNS-сервера.

Распространенные syslog-сообщения

Существует очень много сообщений, записываемых сервером *named* в log-файл демона *syslog*. На практике наблюдаются лишь некоторые из этих сообщений. Мы рассмотрим сообщения, которые чаще всего встречаются в log-файле *syslog*, за исключением сообщений о синтаксических ошибках в файлах данных зон.

При каждом запуске *named*, в log-файл записывается сообщение с приоритетом LOG_NOTICE. Это сообщение выглядит следующим образом (DNS-сервер BIND 8):

```
Jan 10 20:48:32 terminator named[3221]: starting.  named 8.2.3 Tue May 16
09:39:40 MDT 2000 ^Icricket@huskymo.boulder.acmebw.com:/usr/local/src/bind-
8.2.3/src/bin/named
```

В BIND 9 сообщение существенно короче:

```
Jul 27 16:18:41 terminator named[7045]: starting BIND 9.1.0
```

Данное сообщение отмечает тот факт, что DNS-сервер *named* начал работу в определенное время, а также отображает используемую версию BIND, а также (в BIND 8) автора и место сборки. Разумеется, это сообщение не является поводом для беспокойства. Но на него имеет смысл обращать внимание, если неизвестно точно, какие версии BIND поддерживаются операционной системой. (В более старых версиях BIND было сообщение «restarted» (перезапуск) вместо «starting» (запуск).)

Каждый раз при получении команды reload DNS-сервер BIND 8 создает следующее сообщение с приоритетом LOG_NOTICE:

```
Jan 10 20:50:16 terminator named[3221]: reloading nameserver
```

То же для DNS-сервера BIND 9:

```
Jul 27 16:27:45 terminator named[7047]: loading configuration from '/etc/
named.conf
```

Эти сообщения просто отражают тот факт, что *named* произвел перезагрузку базы данных (выполняя команду *reload*) в определенный момент времени. И снова поводов волноваться нет. Это сообщение может представлять интерес, если необходимо определить, сколь долго в данных зоны присутствовала неправильная запись либо сколь долго вся зона была недоступна из-за ошибки во время обновления.

Еще одно сообщение, которое может появиться через некоторое время после запуска DNS-сервера:

```
Jan 10 20:50:20 terminator named[3221]: cannot set resource limits on
                this system
```

Смысл сообщения таков: DNS-сервер считает, что операционная система не поддерживает системные вызовы *getrlimit( )* и *setrlimit( )*, которые нужны при определении *coresize*, *datasize*, *stacksize* или *files* на DNS-сервере BIND 8/9. Неважно, были ли в действительности использованы эти предписания в файле настройки; BIND все равно создаст это сообщение. Если предписания не использовались, сообщение можно просто проигнорировать. В противном случае (и если администратор считает, что операционная система все-таки поддерживает *getrlimit( )* и *setrlimit( )*) – придется пересобрать BIND с определенным ключом HAVE_GETRUSAGE. Сообщение имеет приоритет LOG_INFO.

Если DNS-сервер работает на узле с многочисленными сетевыми интерфейсами (в особенности – виртуальными сетевыми интерфейсами), вскоре после запуска либо через какое-то время может появиться сообщение следующего характера:

```
Jan 10 20:50:31 terminator named[3221]: fcntl(dfd, F_DUPFD, 20): Too
                many open files
Jan 10 20:50:31 terminator named[3221]: fcntl(sfd, F_DUPFD, 20): Too
                many open files
```

Это означает, что у сервера BIND кончились файловые дескрипторы. BIND активно использует файловые дескрипторы в работе: по два на каждый из прослушиваемых сетевых интерфейсов (один для UDP и один для TCP) и один для чтения файлов зон. Если общее число превышает ограничение, накладываемое операционной системой на процессы, BIND окажется не в состоянии открыть необходимые файловые дескрипторы, в результате чего будет создано сообщение. Приоритет сообщения зависит от того, в какой части BIND произошла ошибка при попытке получить файловый дескриптор: чем более критична для работы эта подсистема, тем выше приоритет сообщения.

Действия в этом случае: сократить число файловых дескрипторов, используемых DNS-сервером BIND, либо сделать менее строгим ограничение на число дескрипторов, используемых процессом BIND.

- Если сервер BIND не должен прослушивать отдельные сетевые интерфейсы (в особенности виртуальные), следует воспользоваться предписанием *listen-on*, и настроить BIND на прослушивание только необходимых сетевых интерфейсов. Синтаксис *listen-on* подробно описан в главе 10.
- Если операционная система поддерживает вызовы *getrlimit( )* и *setrlimit( )*, настройте DNS-сервер на использование большего числа файлов с помощью предписания *files*. Синтаксис *files* подробно описан в главе 10.
- Если операционная система слишком сильно ограничивает число открытых файлов, увеличьте пороговое значение перед запуском *named* – с помощью команды *ulimit*.

При каждой загрузке зоны DNS-сервер BIND 8 создает сообщение с приоритетом LOG_INFO:

```
Jan 10 21:49:50 terminator named[3221]: master zone "movie.edu" (IN)
                Loaded (serial 1996011000)
```

(DNS-серверы BIND 4.9 говорят «primary zone» вместо «master zone.») Сообщение содержит информацию о времени загрузки зоны, классе зоны (в данном случае – IN) и порядковый номер зоны из SOA-записи. DNS-серверы BIND 9 – на момент существования версии 9.1.0 – не уведомляют о загрузке зон.

Примерно каждый час DNS-сервер BIND 8 записывает снимок текущей статистики с приоритетом LOG_INFO:

```
Feb 18 14:09:02 terminator named[3565]:  USAGE 824681342 824600158
                CPU=13.01u/3.26s CHILDCPU=9.99u/12.71s
Feb 18 14:09:02 terminator named[3565]: NSTATS 824681342 824600158
                A=4 PTR=2
Feb 18 14:09:02 terminator named[3565]: XSTATS 824681342 824600158
                RQ=6 RR=2 RIQ=0 RNXD=0 RFwdQ=0 RFwdR=0 RDupQ=0 RDupR=0
                RFail=0 RFErr=0 RErr=0 RTCP=0 RAXFR=0 RLame=0 Ropts=0
                SSysQ=2 SAns=6 SFwdQ=0 SFwdR=0 SDupQ=5 SFail=0 SFErr=0
                SErr=0 RNotNsQ=6 SNaAns=2 SNXD=1
```

(Снимки статистики существовали также в BIND 4.9 до версии 4.9.3 и были отключены в сервере версии 4.9.4. BIND 9 – на момент существования версии 9.1.0 – не поддерживает снимки статистики.) Первые два числа в каждом сообщении – временные отметки. Если вычесть второе число из первого, можно узнать, сколько секунд непрерывно работает DNS-сервер. (Странно, что DNS-сервер самостоятельно не производит вычитание.) Запись CPU позволяет определить, сколько времени сервер работал в пользовательском режиме (13,01 секунды) и

системном (3,26 секунды). Такая же статистика приводится для порожденных процессов. Сообщение NSTATS содержит типы запросов, полученных DNS-серверами, и счетчики для каждого из типов. Сообщение XSTATS содержит дополнительную статистику. Более подробно статистика из сообщений NSTATS и XSTATS рассмотрена позже в этой главе.

Если BIND версии 4.9.4 или более поздней (но не BIND 9 до версии 9.1.0, поскольку в BIND 9 не реализована соответствующая проверка) обнаруживает имя, которое не соответствует формату, описанному в документе RFC 952, в log-файл демона *syslog* записывается сообщение об ошибке:

```
Jul 24 20:56:26 terminator named[1496]: owner name "ID_4.movie.edu IN"
                            (primary) is invalid - rejecting
```

Сообщение имеет приоритет LOG_NOTICE. Правила именования узлов описаны в главе 4.

Еще одно сообщение *syslog*, имеющее приоритет LOG_INFO, предупреждает о проблемах, связанных с данными зоны:

```
Jan 10 20:48:38 terminator named[3221]: terminator2 has CNAME
                 and other data (invalid)
```

Допустим, существуют следующие записи:

```
terminator2  IN  CNAME t2
terminator2  IN  MX    10 t2
t2           IN  A     192.249.249.10
t2           IN  MX    10 t2
```

MX-запись для узла *terminator2* некорректна и приводит к получению такого сообщения. *terminator2* – это псевдоним для имени *t2*, которое является каноническим. Как уже говорилось, если при поиске для определенного имени DNS-сервер обнаруживает CNAME-запись, то заменяет исходное имя каноническим и повторяет поиск для канонического имени. Таким образом, при поиске MX-данных для узла *terminator2* DNS-сервер находит CNAME-запись, а затем производит поиск MX-записи для имени *t2*. Поскольку сервер при поиске использует запись CNAME для *terminator2*, до MX-записи *terminator2* дело не доходит; на самом деле, эта запись является недопустимой. Другими словами, все RR-записи для узла должны содержать каноническое имя узла, использование псевдонима вместо канонического имени является ошибкой.

Мы знаем, что начинаем повторяться, но DNS-сервер BIND 9 не обнаруживает эту ошибку до версии 9.1.0.

Следующее сообщение отражает тот факт, что вторичный DNS-сервер BIND 4 или 8 не смог связаться ни с одним из мастер-серверов DNS при попытке получить зону:

```
Jan 10 20:52:42 wormhole named[2813]: zoneref: Masters for
                secondary zone "movie.edu" unreachable
```

Что говорят в этом случае вторичные DNS-серверы BIND 9:

```
Jul 27 16:50:55 terminator named[7174]: refresh_callback: zone movie.edu/IN:
failure for 10.0.0.1#53: timed out
```

Сообщение имеет приоритет LOG_NOTICE в BIND 4 и 8 или LOG_INFO в BIND 9 и посылается только при первой неудачной попытке получения зоны. Когда наконец зона успешно получена, DNS-сервер версии 4.9 или более поздней сообщает об этом, записывая соответствующее сообщение в log-файл *syslog*. Более старые DNS-серверы не упоминают об успешном получении зоны. Когда приведенное выше сообщение появляется впервые, нет необходимости предпринимать какие-либо меры. DNS-сервер продолжит попытки получить зону, исходя из значения интервала повторения в SOA-записи. Через несколько дней (или по прошествии половины интервала устаревания данных), можно проверить, произошло ли получение зоны. В случае серверов, которые не создают сообщения *syslog* при передаче зон, это можно сделать с помощью временной отметки резервной копии файла зоны. Если получение прошло успешно, создается новая резервная копия. Если DNS-сервер обнаруживает, что хранимая зона актуальна, то «дотрагивается» до резервной копии (на манер команды *touch*, существующей в Unix-системах). В обоих случаях временная отметка резервной копии обновляется, так что достаточно соединиться с узлом вторичного DNS-сервера и выполнить команду *ls -l /usr/local/named/db**. Это позволит узнать, когда в последний раз произошла синхронизация хранимой зоны. Разрешение проблем, которые приводят к невозможности получения зон вторичными DNS-серверами, рассмотрено в главе 14.

При чтении *syslog*-сообщений сервера версии 4.9 или более поздней можно обнаружить сообщение с приоритетом LOG_INFO, которое отражает факт получения новой зоны вторичным DNS-сервером либо передачу зоны с помощью инструмента вроде *nslookup*:

```
Mar  7 07:30:04 terminator named[3977]: approved AXFR from
                        [192.249.249.1].2253 for "movie.edu"
```

И снова BIND 9 – на момент существования версии 9.1.0 – не создает такого сообщения.

Если для указания серверов, которым разрешено получения зоны, используется инструкция файла настройки *xfrnets* (BIND 4) или предписание *allow-transfer* (BIND 8) – речь о них пойдет в главе 10 – то можно столкнуться с приводимым выше сообщением, в котором вместо слова *approved* применяется *unapproved*. DNS-сервер BIND 9 сообщает следующее:

```
Jul 27 16:59:26 terminator named[7174]: client 192.249.249.1#1386:   zone
transfer denied
```

А вот это сообщение *syslog* можно увидеть только при чтении сообщений с приоритетом LOG_INFO:

```
Jan 10 20:52:42 wormhole named[2813]: Malformed response
                from 192.1.1.1
```

Чаще всего появление этого сообщения означает, что какая-то ошибка в DNS-сервере привела к отправке некорректного ответного пакета. Вероятно, ошибка произошла на удаленном (192.1.1.1), а не на локальном сервере (*wormhole*). Для диагностирования такой ошибки необходимо воспользоваться сетевым анализатором и декодировать ошибочный пакет. Ручное декодирование пакетов DNS выходит за пределы этой книги, поэтому мы не будем вдаваться в детали. Ошибку такого типа можно увидеть, если пакет ответного сообщения утверждает, что содержит несколько ответов в разделе ответа (например, четыре адресные записи), но в действительности присутствует только один ответ. Разумный курс действий в таком случае – уведомить администратора узла-нарушителя, написав ему письмо (предполагается, что мы можем узнать имя узла, выполнив поиск по адресу). Это сообщение также может указывать на то, что транспортная сеть каким-то образом изменила (повредила) ответные UDP-пакеты. Проверка контрольных сумм UDP-пакетов является необязательной, поэтому такая ошибка может не обнаружиться на более низких уровнях системы.

DNS-сервер BIND 4.9 или 8 создает следующее сообщение при попытке добавить в файл данных зоны записи, которые принадлежат совсем другой зоне:

```
Jun 13 08:02:03 terminator named[2657]: db.movie.edu:28: data "foo.bar.edu"
                            outside zone "movie.edu" (ignored)
```

named BIND 9 в таком случае сообщает:

```
Jul 27 17:07:01 terminator named[7174]: dns_master_load: db.movie.edu:28:
ignoring out-of-zone data
```

К примеру, если бы мы попытались использовать такие зональные данные:

```
robocop       IN A  192.249.249.2
terminator    IN A  192.249.249.3

; добавить эту запись в кэш DNS-сервера
foo.bar.edu.  IN A  10.0.7.13
```

то сделали бы попытку добавить данные зоны *bar.edu* в файл данных зоны *movie.edu*. Старомодный DNS-сервер версии 4.8.3 добавит *foo.bar.edu* в данные кэша и даже не подумает удостовериться, что все данные в файле *db.movie.edu* принадлежат зоне *movie.edu*. Сервер версии более поздней, чем 4.9, уже не обмануть. Данное сообщение *syslog* имеет приоритет LOG_INFO.

Ранее в тексте книги мы говорили, что запрещено использовать псевдонимы в информативной части RR-записи. BIND версий 4.9 и 8 обнаруживают подобные нарушения:

```
Jun 13 08:21:04 terminator named[2699]: "movie.edu IN NS" points to a
                                        CNAME (dh.movie.edu)
```

Того же нельзя сказать о BIND 9 – на момент существования версии 9.1.0.

Вот пример некорректных RR-записей:

```
@                       NS       terminator.movie.edu.
                        NS       dh.movie.edu.
terminator.movie.edu.   IN A     192.249.249.3
diehard.movie.edu.      IN A     192.249.249.4
dh                      IN CNAME diehard
```

Во второй NS-записи должен быть указан сервер *diehard.movie.edu*, а не *dh.movie.edu*. Это сообщение не появляется в log-файле немедленно после запуска DNS-сервера.

Это *syslog*-сообщение появляется в log-файле только при поиске некорректных записей. DNS-серверами BIND 4.9.3 и BIND 8 используется в этом случае приоритет LOG_INFO, серверами с 4.9.4 по 4.9.7 – приоритет LOG_DEBUG.

Следующее сообщение показывает, что DNS-сервер, возможно, отражал один из видов сетевых атак:

```
Jun 11 11:40:54 terminator named[131]: Response from unexpected source
                                       ([204.138.114.3].53)
```

DNS-сервер отправил запрос удаленному DNS-серверу, но полученный ответ был отправлен не с адреса удаленного DNS-сервера. Вот типичный сценарий атаки: злоумышленник побуждает DNS-сервер послать запрос удаленному DNS-серверу и в то же время самостоятельно посылает ответы (претендуя на то, что ответы исходят от удаленного DNS-сервера), которые, как он надеется, будут кэшированы атакуемым DNS-сервером. Вполне возможно, что злоумышленник посылает ложную PTR-запись, которая связывает IP-адрес одного из его узлов с доменным именем узла, которому доверяет наша система. Как только ложная PTR-запись попадает в кэш DNS-сервера, злоумышленник использует одну из *r*-команд BSD-систем (к примеру, *rlogin*) для получения доступа к нашей системе.

Администраторы, страдающие паранойей в меньшей степени, поймут, что такая ситуация может образоваться и в том случае, если DNS-сервер родительской зоны знает только один из IP-адресов DNS-сервера порожденной зоны, входящего в несколько сетей. Родитель сообщает вашему DNS-серверу единственный известный ему IP-адрес, и когда ваш DNS-сервер посылает запрос удаленному DNS-серверу, то получа-

ет ответ с другого IP-адреса. Это не должно происходить, если на удаленном сервере работает BIND, поскольку BIND прилагает все возможные усилия, чтобы ответить с того же IP-адреса, для которого получен запрос. Данное сообщение имеет приоритет LOG_INFO.

Вот интересное сообщение *syslog*:

```
Jun 10 07:57:28 terminator named[131]: No root nameservers for
                class 226
```

В настоящее время определены следующие классы: класс 1, Интернет (IN); класс 3, Chaos (CH); класс 4, Hesiod (HS). Что за класс 226? Именно это DNS-сервер и пытается сказать своим сообщением – что-то не так, поскольку класса 226 не существует. Что можно сделать? Практически ничего. Сообщение не содержит достаточно информации – неизвестно, от кого получен запрос или с какой целью был сделан этот запрос. Впрочем, если искажено поле класса, вполне возможно, что та же участь постигла и доменное имя в запросе. Действительной причиной проблемы может быть неправильно работающий удаленный DNS-сервер или клиент либо искаженная UDP-дейтаграмма. Данное *syslog*-сообщение имеет приоритет LOG_INFO.

Следующее сообщение может обнаружиться, если DNS-сервер используется в качестве резервного для какой-либо зоны:

```
Jun  7 20:14:26 wormhole named[29618]: Zone "253.253.192.in-addr.arpa"
                (class 1) SOA serial# (3345) rcvd from [192.249.249.10]
                is < ours (563319491)
```

Ага, вредный администратор зоны *253.253.192.in-addr.arpa* изменил формат порядкового номера и забыл сказать об этом вам. Вот она, благодарность за сопровождение вторичного DNS-сервера этой зоны! Черкните письмо администратору, чтобы понять, было ли это сделано специально или просто в результате опечатки. Если изменения были сделаны сознательно либо если вы не хотите контактировать с этим администратором, придется решить проблему локально – остановить DNS-сервер, удалить резервную копию файла зоны и снова запустить DNS-сервер. Эта процедура стирает из памяти вторичного сервера старый порядковый номер, и после этого сервер радостно принимает зону с новым порядковым номером. Данное сообщение *syslog* имеет приоритет LOG_NOTICE.

Между прочим, если тот вредный администратор использует DNS-сервер BIND 8 или 9, то, по всей видимости, он пропустил (или проигнорировал) сообщение, записанное в log-файл его DNS-сервером, а именно – сообщение, отражающее тот факт, что порядковый номер зоны уменьшился. Для DNS-сервера BIND 8 сообщение выглядит следующим образом:

```
Jun 7 19:35:14 terminator named[3221]: WARNING: new serial number < old
               (zp->z_serial < serial)
```

Оно же для DNS-сервера BIND 9:

```
Jun 7 19:36:41 terminator named[9832]: dns_zone_load: zone movie.edu/IN: zone
serial has gone backwards
```

Приоритет сообщения – LOG_NOTICE.

Возможно, имеет смысл напомнить этому администратору о мудрости, предписывающей проверять log-файл демона *syslog* после внесения изменений в работу DNS-сервера.

Следующее сообщение BIND 8 многие администраторы, вне всякого сомнения, выучат наизусть:

```
Aug 21 00:59:06 terminator named[12620]: Lame server on 'foo.movie.edu'
         (in 'MOVIE.EDU'?): [10.0.7.125].53 'NS.HOLLYWOOD.LA.CA.US':
          learnt (A=10.47.3.62,NS=10.47.3.62)
```

В BIND 9 оно выглядит так:

```
Jan 15 10:20:16 terminator named[14205]: lame server on 'foo.movie.edu' (in
'movie.EDU'?): 10.0.7.125#53
```

«Господин капитан, судно обрастает грязью!» В водах сети Интернет грязь существует в виде некорректного делегирования. DNS-сервер родительской зоны делегирует поддомен DNS-серверу порожденной зоны, а DNS-сервер порожденной зоны не является авторитативным для поддомена. В данном случае DNS-сервер зоны *edu* делегирует зону *movie.edu* адресу 10.0.7.125, и DNS-сервер, работающий на этом узле, не является авторитативным для *movie.edu*. Если не знать, кто является администратором зоны *movie.edu*, то скорее всего ничего поделать нельзя. Данное сообщение *syslog* имеет приоритет LOG_WARNING (DNS-сервер версии 4.9.3), LOG_DEBUG (DNS-серверы с 4.9.4 по 4.9.7), либо LOG_INFO (BIND 8 и 9).

Если в файле настройки DNS-сервера BIND версии 4.9 или более поздней присутствует строка:

```
options query-log
```

либо в файле настройки BIND 8/9 присутствует строка:

```
logging { category queries { default_syslog; }; };
```

сообщение приоритета LOG_INFO будет записываться в log-файл *syslog* для каждого запроса, получаемого DNS-сервером:

```
Feb 20 21:43:25 terminator named[3830]:
            XX /192.253.253.2/carrie.movie.edu/A
Feb 20 21:43:32 terminator named[3830]:
            XX /192.253.253.2/4.253.253.192.in-addr.arpa/PTR
```

BIND 9 поддерживает регистрацию запросов на момент существования версии 9.1.0. Однако формат немного отличается:

```
Jan 13 18:32:25 terminator named[13976]: client 192.253.253.2#1702: query:
carrie.movie.edu IN A
Jan 13 18:32:42 terminator named[13976]: client 192.253.253.2#1702: query:
4.253.253.192.in-addr.arpa IN PTR
```

Каждое сообщение включает IP-адрес узла, который сделал запрос, а также собственно запрос. В серверах BIND версии 8.2.1 и более поздних рекурсивные запросы отмечаются подстрокой XX+, а не XX. Перед включением регистрации запросов на загруженном сервере имен следует убедиться в наличии достаточного объема дискового пространства. (Включение и выключение регистрации запросов для работающего сервера можно выполнять с помощью команды *querylog*.)

Начиная с BIND версии 8.1.2 существует возможность встретиться с приводимым ниже набором *syslog*-сообщений:

```
May 19 11:06:08 named[21160]: bind(dfd=20, [10.0.0.1].53):
                Address already in use
May 19 11:06:08 named[21160]: deleting interface [10.0.0.1].53
May 19 11:06:08 named[21160]: bind(dfd=20, [127.0.0.1].53):
                Address already in use
May 19 11:06:08 named[21160]: deleting interface [127.0.0.1].53
May 19 11:06:08 named[21160]: not listening on any interfaces
May 19 11:06:08 named[21160]: Forwarding source address
                is [0.0.0.0].1835
May 19 11:06:08 named[21161]: Ready to answer queries.
```

Для DNS-серверов BIND 9 этот набор выглядит так:

```
Jul 27 17:15:58 terminator named[7357]: listening on IPv4 interface lo,
127.0.0.1#53
Jul 27 17:15:58 terminator named[7357]: binding TCP socket: address in use
Jul 27 17:15:58 terminator named[7357]: listening on IPv4 interface eth0,
206.168.194.122#53
Jul 27 17:15:58 terminator named[7357]: binding TCP socket: address in use
Jul 27 17:15:58 terminator named[7357]: listening on IPv4 interface eth1,
206.168.194.123#53
Jul 27 17:15:58 terminator named[7357]: binding TCP socket: address in use
Jul 27 17:15:58 terminator named[7357]: couldn't add command channel
0.0.0.0#953: address in use
```

Произошло вот что: DNS-сервер был запущен, и при этом администратор запустил вторую копию DNS-сервера, не остановив работу первой. Вопреки ожиданиям, второй DNS-сервер продолжает работу, просто он не производит прослушивание сетевых интерфейсов.

Интерпретация статистики BIND

Администратору имеет смысл периодически заглядывать в статистику, связанную с работой отдельных DNS-серверов, пусть даже только для того, чтобы увидеть, насколько сильно они загружены. Мы приве-

дем примеры статистики, создаваемой DNS-сервером, и объясним смысл всех показателей. В процессе нормальной работы DNS-серверы обрабатывают много запросов и ответов, поэтому сначала мы покажем, как может выглядеть типичный обмен.

Объяснения показателей статистики будет сложнее понять без умозрительной диаграммы процессов, происходящих при поиске в DNS. Чтобы читателям было проще понять статистику DNS-сервера, мы покажем (рис. 7.2), что может происходить, когда приложение производит поиск для доменного имени. Приложение, в данном случае FTP-клиент, посылает запрос локальному DNS-серверу. Локальный DNS-

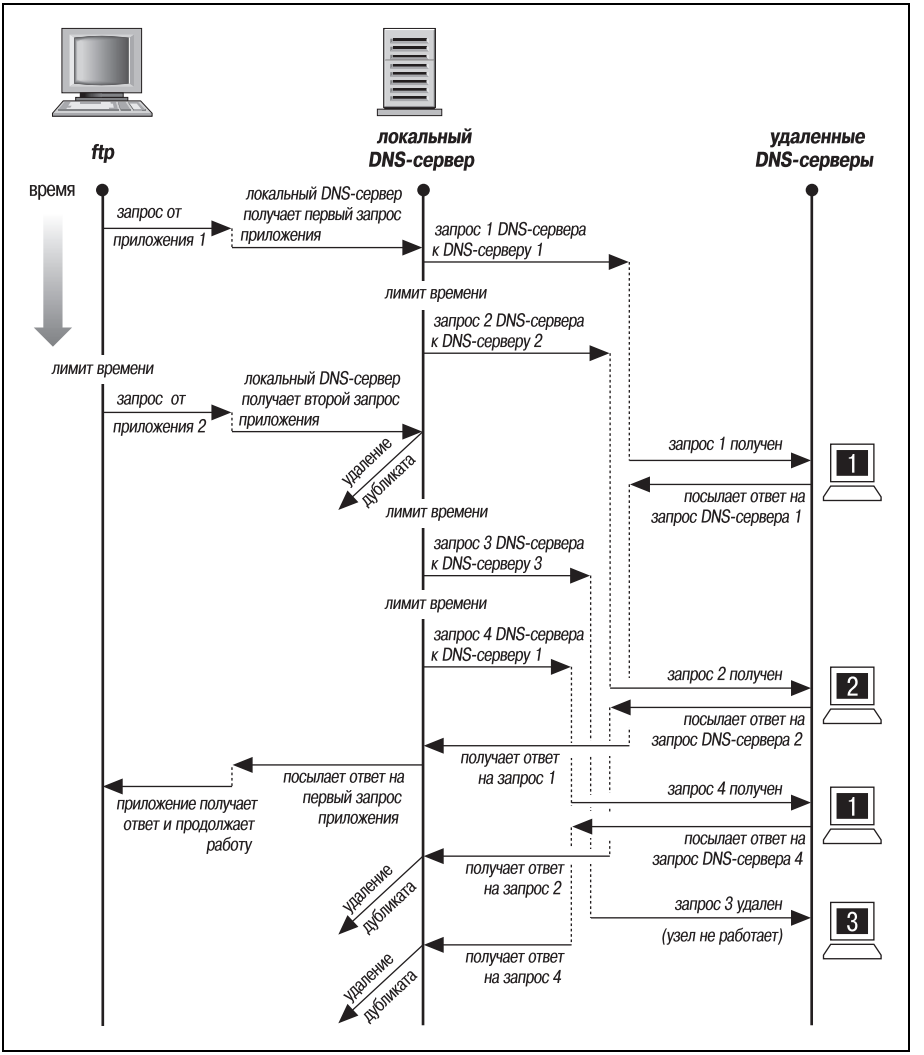

Рис. 7.2. *Запросы и ответы, пример обмена*

сервер до этого уже запрашивал данные из той же зоны, поэтому помнит, где расположены удаленные DNS-серверы. Он посылает запросы каждому из этих серверов – причем одному из них дважды – в попытках найти ответ. Тем временем, интервал ожидания в приложении истекает, и приложение посылает второй запрос, касающийся той же информации.

Следует иметь в виду вот что: несмотря на то, что DNS-сервер отправил запрос к удаленному DNS-серверу, удаленный сервер мог получить запрос не сразу. Запрос мог задержаться или потеряться в процессе доставки, а узел удаленного DNS-сервера мог быть занят обработкой других запросов.

DNS-сервер BIND способен обнаруживать запросы-дубликаты только тогда, когда находится в процессе поиска ответа на исходный запрос. Локальный DNS-сервер обнаруживает запрос-дубликат, посланный приложением, поскольку все еще работает с этим запросом. Но удаленный DNS-сервер 1 не может обнаружить запрос-дубликат, поступивший с локального DNS-сервера, поскольку на предыдущий запрос уже был дан ответ. После получения локальным DNS-сервером ответа от удаленного сервера 1, все прочие ответы удаляются как дублирующие. Диалог на диаграмме потребовал следующих взаимодействий:

Взаимодействие	**Число**
Приложение – локальному DNS-серверу	2 запроса
Локальный DNS-сервер – приложению	1 ответ
Локальный DNS-сервер – удаленному DNS-серверу 1	2 запроса
Удаленный DNS-сервер 1 – локальному DNS-серверу	2 ответа
Локальный DNS-сервер – удаленному DNS-серверу 2	1 запрос
Удаленный DNS-сервер 2 – локальному DNS-серверу	1 ответ
Локальный DNS-сервер – удаленному DNS-серверу 3	1 запрос
Удаленный DNS-сервер 3 – локальному DNS-серверу	0 ответов

Эти взаимодействия приводят к добавлению следующих величин к показателям статистики DNS-сервера:

Показатель	**Причина**
2 запроса получено	От приложения на локальном узле
1 запрос-дубликат	От приложения на локальном узле
1 ответ отправлен	Приложению на локальном узле
3 ответа получено	От удаленных DNS-серверов
2 ответа-дубликата	От удаленных DNS-серверов
2 А-запроса	Запросы адресной информации

В нашем примере локальный DNS-сервер получал запросы только от приложения, но сам посылал запросы удаленным DNS-серверам. Обычно локальный DNS-сервер получает еще и запросы от удаленных DNS-серверов (то есть локальный сервер не только запрашивает требуемую информацию у других DNS-серверов, но и предоставляет другим DNS-серверам информацию, актуальную для них), но в целях упрощения мы не учитывали такие запросы.

Статистика BIND 4.9 и BIND 8

Изучив типичный обмен между приложениями и DNS-серверами, а также генерируемую при обмене статистику, рассмотрим более подробные примеры статистики. Чтобы получить статистику DNS-сервера BIND 8, воспользуемся командой *ndc*:

```
# ndc stats
```

Если используется более старый DNS-сервер BIND 4 без программы *ndc*, следует послать процессу *named* сигнал ABRT:

```
# kill -ABRT `cat /var/run/named.pid`
```

(На Unix-системах версий более ранних, чем SVR4, идентификатор процесса хранится в файле */etc/named.pid*.) Необходимо подождать несколько секунд, а затем изучить файл *named.stats* в рабочем каталоге DNS-сервера (BIND 8) либо файл */var/tmp/named.stats* или */usr/tmp/named.stats* (BIND 4). Если статистика отсутствует в этом файле, при сборке DNS-сервера, вероятно, не был определен ключ STATS, и сервер, таким образом, не занимается сбором статистики. Ниже приводится статистика одного из серверов BIND 4.9.3 Пола Викси. DNS-серверы BIND 8 создают статистику для всех присутствующих здесь пунктов, за исключением RnotNsQ, и отображают ее в несколько ином порядке. DNS-серверы BIND 9 на момент существования версии 9.1.0 создают совершенно отличный набор статистической информации, о котором мы расскажем в следующем разделе.

```
+++ Statistics Dump +++ (800708260) Wed May 17 03:57:40 1995
746683    time since boot (secs)
392768    time since reset (secs)
14        Unknown query types
268459    A queries
3044      NS queries
5680      CNAME queries
11364     SOA queries
1008934   PTR queries
44        HINFO queries
680367    MX queries
2369      TXT queries
40        NSAP queries
27        AXFR queries
8336      ANY queries
```

```
++ Name Server Statistics ++
 (Legend)
       RQ    RR    RIQ   RNXD    RFwdQ
       RFwdR RDupQ RDupR RFail   RFErr
       RErr  RTCP  RAXFR RLame   ROpts
       SSysQ SAns  SFwdQ SFwdR   SDupQ
       SFail SFErr SErr  RNotNsQ SNaAns
       SNXD
 (Global)
   1992938 112600 0 19144 63462 60527 194 347 3420 0  5 2235 27 35289 0
   14886 1927930 63462 60527 107169  10025 119 0 1785426 805592  35863
[15.255.72.20]
   485 0 0 0 0  0 0 0 0 0  0 0 0 0 0  0 485 0 0 0  0 0 0 0 485  0
[15.255.152.2]
   441 137 0 1 2 108 0 0 0 0  0 0 0 0 0  13 439 85 7 84  0 0 0 0 431  0
[15.255.152.4]
   770 89 0 1 4  69 0 0 0 0  0 0 0 0 0  14 766 68 5 7  0 0 0 0 755  0
...  <множество удаленных записей>
```

Если в статистике DNS-сервера BIND 8 отсутствуют частные разделы для отдельных IP-адресов после строки «(Global)», следует установить значение *host-statistics* в операторе *options*, если необходимо отслеживать статистику для узлов:

```
options {
 host-statistics yes;
};
```

Однако хранение статистики по отдельным узлам требует больших объемов памяти, так что совершенно необязательно собирать такую статистику постоянно, достаточно делать это в случае необходимости произвести профилировку работы DNS-сервера.

Рассмотрим статистику построчно.

```
+++ Statistics Dump +++ (800708260) Wed May 17 03:57:40 1995
```

Дата создания раздела статистики. Число в скобках (800708260) определяет количество секунд, прошедших с начала эры Unix, то есть с первого января 1970 года. К счастью, BIND преобразует это значение в традиционную временную отметку: May 17, 1995, 3:57:40 a.m.

```
746683    time since boot (secs)
```

Время непрерывной работы локального DNS-сервера. Чтобы получить число дней, следует разделить показатель на 86 400 (60×60×24, количество секунд в сутках). Этот сервер работал примерно 8,5 дней.

```
392768    time since reset (secs)
```

Время работы локального DNS-сервера с момента последней перезагрузки. Это время будет отличаться от времени непрерывной работы только в том случае, если сервер является первичным мастер-сервером

DNS для одной или нескольких зон. Дополнительные DNS-серверы автоматически воспринимают новые данные при получении зон, и обычно нет необходимости их перезагружать. Поскольку для *этого* сервера явно была произведена перезагрузка, вероятно, он является первичным мастер-сервером DNS какой-либо зоны.

```
14        Unknown query types
```

DNS-сервер получил 14 запросов данных неизвестного типа. Либо кто-то экспериментирует с новыми типами записей, либо работает с некорректной реализацией DNS, либо Полу пора обновить свой DNS-сервер.

```
268459    A queries
```

Выполнено 268459 запросов, связанных с поиском адресов. Как правило, адресные запросы встречаются чаще всего.

```
3044      NS queries
```

Выполнено 3044 запросов NS-записей. DNS-серверы создают скрытые NS-запросы в процессе поиска DNS-серверов корневой зоны. Для поиска NS-записей можно также использовать приложения *dig* и *nslookup*.

```
5680      CNAME queries
```

Некоторые из версий *sendmail* создают CNAME-запросы в процессе канонизации почтового адреса (то есть в процессе замены псевдонима каноническим именем). Прочие версии *sendmail* используют с этой целью запросы ANY (до которых мы вскоре доберемся). В остальных случаях CNAME-запросы поступают в основном от приложений вроде *dig* и *nslookup*.

```
11364     SOA queries
```

SOA-запросы выполняются вторичными DNS-серверами в целях проверки актуальности хранимых зон. Если данные не являются актуальными, следом выполняется AXFR-запрос, инициирующий получение зоны. Поскольку в этом наборе статистики отражены AXFR-запросы, можно сделать вывод, что DNS-серверы получают зональные данные с этого сервера.

```
1008934   PTR queries
```

PTR-запросы связаны с необходимостью преобразования адресов в имена. Многие программы производят поиск по IP-адресам: *inetd*, *rlogind*, *rshd*, а также программное обеспечение для управления сетями и сетевой трассировки.

```
44        HINFO queries
```

Запросы информации об узлах вероятнее всего исходят от человека, производящего поиск HINFO-записей в диалоговом режиме.

```
680367    MX queries
```

Почтовые программы вроде *sendmail* создают запросы MX-записей в процессе стандартной процедуры доставки электронных сообщений.

```
2369      TXT queries
```

Исходя из порядка полученного числа, можно сделать вывод, что запросы текстовых записей создаются приложениями. Вполне возможно, кто-то использует инструмент, подобный программе *Harvest*, реализующей технологию поиска информации, разработанную в Университете штата Колорадо.

```
40        NSAP queries
```

NSAP – это относительно новый тип записей, который применяется для отображения доменных имен в адреса OSI Network Service Access Point.

```
27        AXFR queries
```

Дополнительные DNS-серверы создают AXFR-запросы, инициирующие получение зон.

```
8336      ANY queries
```

Запросы ANY позволяют производить поиск записей любого типа для доменного имени. Чаще всего этот тип запросов используется программой *sendmail*. Поскольку *sendmail* запрашивает записи CNAME, MX, а также адресные для конечного адресата, эффективно сделать запрос ANY, чтобы все RR-записи для имени были кэшированы локальным DNS-сервером.

Вся остальная статистика относится к отдельным узлам. Если просмотреть список узлов, с которыми DNS-сервер обменивался пакетами, можно увидеть, насколько общителен сервер – в списке будут сотни или даже тысячи узлов. Размер списка, конечно, может впечатлять, но собственно статистика не особо полезна. Мы расскажем обо *всех* статистических показателях, даже о нулевых, несмотря на то, что администраторам пригодятся мало какие из них. Чтобы облегчить чтение статистики, понадобится специальный инструмент, поскольку исходный формат весьма компактен. Мы написали программу, которая называется *bstat* и выполняет ровно эту задачу. Вот так выглядит вывод программы:

```
hpcvsop.cv.hp.com
        485 queries received
        485 responses sent to this name server
        485 queries answered from our cache
relay.hp.com
        441 queries received
        137 responses received
           1 negative response received
           2 queries for data not in our cache or authoritative data
```

```
          108 responses from this name server passed to the querier
           13 system queries sent to this name server
          439 responses sent to this name server
           85 queries sent to this name server
            7 responses from other name servers sent to this name server
           84 duplicate queries sent to this name server
          431 queries answered from our cache
hp.com
          770 queries received
           89 responses received
            1 negative response received
            4 queries for data not in our cache or authoritative data
           69 responses from this name server passed to the querier
           14 system queries sent to this name server
          766 responses sent to this name server
           68 queries sent to this name server
            5 responses from other name servers sent to this name server
            7 duplicate queries sent to this name server
          755 queries answered from our cache
```

В необработанной статистике для каждого IP-адреса узла приводится таблица счетчиков. Заголовок таблицы представляет собой загадочные письмена – легенду. Легенда разбита на несколько строк, но статистика для каждого узла содержится в одной строке. В следующем разделе мы коротко объясним смысл каждой из колонок, когда будем изучать статистику для одного из узлов-собеседников DNS-сервера – узла с адресом 15.255.152.2 (*relay.hp.com*). Для удобства мы будем приводить заголовок колонки из легенды (скажем, RQ) и счетчик из этой колонки, связанный с узлом *relay*.

```
RQ 441
```

RQ – это счетчик запросов, полученных от узла *relay*. Эти запросы были сделаны, поскольку узлу *relay* была нужна информация о зоне, обслуживаемой данным DNS-сервером.

```
RR 137
```

RR – это счетчик ответов, полученных от узла *relay*. Речь идет об ответах на запросы, сделанные данным DNS-сервером. Не стоит пытаться сопоставлять это число с показателем RQ, поскольку никакой связи нет. RQ – счетчик вопросов, заданных узлом *relay*; RR – счетчик ответов, данных узлом *relay* DNS-серверу (этот DNS-сервер запрашивал информацию у узла *relay*).

```
RIQ 0
```

RIQ – это счетчик обратных запросов, полученных от узла *relay*. Обратные запросы изначально предназначались для отображения адресов в имена, но эту функциональность в настоящее время обеспечивают PTR-записи. Старые версии *nslookup* использовали обратные запро-

сы при запуске, поэтому в теории счетчик RIQ может оказаться ненулевым.

```
RNXD 1
```

RNXD – счетчик ответов «no such domain» (домен не существует), полученных от узла *relay*.

```
RFwdQ 2
```

RFwdQ – это счетчик запросов, которые были получены от узла *relay* (RQ) и потребовали дальнейшей обработки для получения ответа. Этот показатель гораздо выше для узлов, DNS-клиенты которых настроены (с помощью файла *resolv.conf*) на посылку всех запросов данному DNS-серверу.

```
RFwdR 108
```

RFwdR – это счетчик полученных от узла *relay* ответных сообщений (RR), которые содержали ответы на исходные вопросы и были переданы пользовательским приложениям.

```
RDupQ 0
```

RDupQ – это счетчик дубликатов запросов, полученных от узла *relay*. Дубликаты появляются только в том случае, когда клиент настроен на посылку запросов данному DNS-серверу.

```
RDupR 0
```

RDupR – это счетчик дубликатов ответов, полученных от узла *relay*. Ответ считается дубликатом в случае, когда DNS-сервер не может найти в списке текущих запросов исходный, который привел к получению данного ответа.

```
RFail 0
```

RFail – это счетчик SERVFAIL-ответов, полученных от узла *relay*. Ответ SERVFAIL указывает на сбой DNS-сервера. Чаще всего ответ SERVFAIL говорит о том, что удаленный DNS-сервер нашел синтаксическую ошибку при чтении файла данных зоны. Любые запросы, касающиеся данных из этой зоны, будут приводить к получению ответа SERVFAIL от удаленного сервера. Помимо этого, причиной сообщения о сбое сервера может быть ошибка выделения памяти на удаленном сервере, либо устаревание данных зоны, хранимой вторичным DNS-сервером.

```
RFErr 0
```

RFErr – это счетчик FORMERR-ответов, полученных от узла *relay*. Ошибка FORMERR связана с некорректным форматом запроса.

```
RErr 0
```

RErr – это счетчик ошибок (кроме SERVFAIL и FORMERR).

```
RTCP 0
```

RTCP – это счетчик запросов, полученных от узла *relay*, через TCP-соединения. (В большинстве запросов используется UDP.)

```
RAXFR 0
```

RAXFR – это счетчик инициированных процессов передачи зон. Нулевой показатель говорит о том, что узел *relay* не является вторичным сервером ни для одной из зон, обслуживаемых данным DNS-сервером.

```
RLame 0
```

RLame – это счетчик случаев некорректного делегирования. Если значение счетчика ненулевое, это означает, что одна из зон делегирована DNS-серверу по текущему IP-адресу, но DNS-сервер не является авторитативным для этой зоны.

```
ROpts 0
```

ROpts – это счетчик полученных пакетов с установленными IP-параметрами.

```
SSysQ 13
```

SSysQ – это счетчик системных запросов, посланных узлу *relay*. Системными называются запросы, которые по собственной инициативе делает локальный DNS-сервер. Большинство системных запросов обращены к корневым DNS-серверам, поскольку системные запросы используются для обновления перечня корневых DNS-серверов. Помимо этого системные запросы также нужны для поиска адреса DNS-сервера в случае, когда адресная запись устарела раньше, чем NS-запись. Поскольку узел *relay* не является корневым DNS-сервером, имеет место второй случай.

```
SAns 439
```

SAns – это счетчик ответов, посланных узлу *relay*. Данный DNS-сервер ответил на 439 запросов из 441 (RQ), полученного от узла *relay*. Интересно, что случилось с двумя запросами, которые остались без ответов...

```
SFwdQ 85
```

SFwdQ – это счетчик запросов, которые были посланы (ретранслированы) узлу *relay* в связи с тем, что ответа не было в зональных данных или кэше данного DNS-сервера.

```
SFwdR 7
```

SFwdR – это счетчик ответов от какого-то DNS-сервера, которые были посланы (ретранслированы) узлу *relay*.

```
SDupQ 84
```

SDupQ – это счетчик дубликатов запросов, отправленных узлу *relay*. Все не так плохо, как кажется. Счетчик дубликатов увеличивается, если запрос был до этого отправлен любому другому DNS-серверу. Вполне возможно, что узел *relay* с первого раза ответил на все полученные запросы, но некоторые из них все равно привели к увеличению счетчика дубликатов, поскольку до этого посылались другим DNS-серверам.

```
SFail 0
```

SFail – это счетчик SERVFAIL-ответов, посланных узлу *relay*.

```
SFErr 0
```

SFErr – это счетчик FORMERR-ответов, посланных узлу *relay*.

```
SErr 0
```

SErr – это счетчик системных вызовов *sendto( )*, завершившихся неудачно для адресата *relay*.

```
RNotNsQ 0
```

RNotNsQ – это счетчик запросов, полученных не со стандартного порта DNS-сервера, 53. До появления BIND 8 все запросы к DNS-серверам поступали с порта 53. Запросы, поступавшие с любого другого порта, являлись запросами клиента. Но DNS-серверы BIND 8 посылают запросы через порты, отличные от стандартного, что делает этот статистический показатель бесполезным, поскольку запросы клиентов невозможно отличить от запросов DNS-серверов. Поэтому в BIND 8 показатель RNotNsQ не включается в статистику.

```
SNaAns 431
```

SNaAns – это счетчик неавторитативных ответов, посланных узлу *relay*. Из 439 ответов (SAns), посланных узлу *relay*, 431 ответ был извлечен из кэшированных данных.

```
SNXD 0
```

SNXD – счетчик ответов «no such domain», посланных узлу *relay*.

Статистика BIND 9

BIND 9.1.0 – первая версия пакета BIND 9, в которой реализован сбор статистической информации. Для получения статистики в BIND 9 можно воспользоваться командой *rndc*:

```
% rndc stats
```

DNS-сервер записывает статистику (как и в случае BIND 8) в файл с именем *named.stats* в своем рабочем каталоге. Однако статистика разительно отличается от получаемой в BIND 8. Вот содержимое файла статистики одного из наших DNS-серверов BIND 9:

```
+++ Statistics Dump +++ (979436130)
success 9
referral 0
nxrrset 0
nxdomain 1
recursion 1
failure 1
--- Statistics Dump --- (979436130)
+++ Statistics Dump +++ (979584113)
success 651
referral 10
nxrrset 11
nxdomain 17
recursion 296
failure 217
--- Statistics Dump --- (979584113)
```

DNS-сервер добавляет новый раздел статистической информации (заключенный между строк «+++ Statistics Dump +++» и «--- Statistics Dump ---») при каждом получении команды *stats*. Число в скобках (979436130), как и в других рассмотренных файлах статистики, определяет число секунд, истекших с начала эпохи Unix. К сожалению BIND не производит преобразования значений. С этой целью можно воспользоваться командой date и получить более понятную дату. К примеру, чтобы преобразовать 979584113 секунд эпохи Unix (которая началась 1 января 1970 года) в дату, можно выполнить команду:

```
% date -d '1970-01-01 979584113 sec'
Mon Jan 15 18:41:53 MST 2001
```

Рассмотрим полученную статистику построчно.

```
success 651
```

Количество запросов, успешно рассмотренных DNS-сервером. То есть запросов, которые не привели к ошибкам или перенаправлениям.

```
referral 10
```

Количество запросов, в ответ на которые DNS-сервер вернул ссылки.

```
nxrrset 11
```

Количество запросов, в ответ на которые DNS-сервер сообщил, что записей запрошенного типа для данного доменного имени не существует.

```
nxdomain 17
```

Количество запросов, в ответ на которые DNS-сервер сообщил, что доменное имя, фигурирующее в запросе, не существует.

```
recursion 296
```

Количество запросов, потребовавших рекурсивного разрешения для получения ответа.

```
failure 217
```

Количество запросов, приведших к ошибкам, не относящимся к случаям *nxrrset* и *nxdomain*.

Можно видеть, что статистической информации не так много, как в BIND 8, но в будущих версиях BIND 9 эта ситуация, вероятнее всего, изменится.

Использование статистики BIND

«Здоров» ли DNS-сервер? Известно ли, как выглядит «нормальная» работа? Невозможно определить, правильно ли работает DNS-сервер, изучив его состояние в отдельный момент времени. Чтобы сделать это, необходимо проследить генерируемую статистику за некоторый период времени, и понять, какой порядок чисел является нормальным для существующей конфигурации. Числа могут заметно различаться для различных DNS-серверов, в зависимости от набора приложений, посылающих запросы, типа сервера (первичный мастер, вторичный, только кэширующий) и уровня пространства имен для зон, обслуживаемых сервером.

В полученной статистике имеет смысл обращать внимание на число запросов, получаемых DNS-сервером за одну секунду. Число полученных запросов следует разделить на число секунд работы DNS-сервера. Сервер Пола, BIND 4.9.3, получил 1992938 запросов за 746683 секунды, то есть примерно 2,7 запроса в секунду, то есть не был сильно загружен.[1] Если получившееся для сервера число кажется неправильным, следует обратить внимание на узлы, от которых исходит большая часть запросов, и попытаться понять, насколько необходим этим узлам такой объем запросов. В какой-то момент, возможно, придется увеличить число DNS-серверов, чтобы справиться с нагрузкой. Такую ситуацию мы рассмотрим в следующей главе.

[1] Вспомним, что корневые DNS-серверы, на которых используется стандартный BIND, могут обрабатывать тысячи запросов в секунду.

8

- *Сколько DNS-серверов?*
- *Добавление DNS-серверов*
- *Регистрация DNS-серверов*
- *Изменение значений TTL*
- *Подготовка к бедствиям*
- *Борьба с бедствиями*

Развитие домена

– А какого роста ты хочешь быть? – спросила, наконец, Гусеница.
– Ах, все равно, – быстро сказала Алиса. – Только, знаете, так неприятно все время меняться...
– А теперь ты довольна? – спросила Гусеница.
– Если вы не возражаете, сударыня, – отвечала Алиса, – мне бы хотелось хоть капельку подрасти.

Сколько DNS-серверов?

В главе 4 «Установка BIND» мы настроили два DNS-сервера. Два сервера – этот тот минимум, меньше которого администратору вряд ли придется использовать. В зависимости от размера сети может потребоваться гораздо больше, чем пара серверов. Нет ничего удивительного в наборе из пяти-семи серверов, один из которых работает вне основной площадки. Сколько серверов будет достаточно? Это следует определять, исходя из требований конкретной сети. Вот некоторые основные положения, которые можно использовать:

- В каждой сети или подсети должен работать по меньшей мере один DNS-сервер. В этом случае маршрутизаторы перестают быть критичными для работы. Следует максимально использовать существующие узлы, входящие одновременно в несколько сетей.
- Если существует файл-сервер, обслуживающий группу бездисковых станций, следует установить DNS-сервер на файл-сервере – в целях обслуживания этой группы машин.
- DNS-серверы должны присутствовать неподалеку от крупных многопользовательских машин, но необязательно на этих машинах.

Пользователи и их процессы, вероятно, служат источником многочисленных запросов, и администратор будет стараться содержать такой узел в рабочем состоянии. Однако следует сопоставлять нужды пользователей с риском иметь запущенный DNS-сервер – для которого безопасность имеет особое значение – на системе, к которой есть доступ у большого числа пользователей.

- По крайней мере один DNS-сервер должен работать вне основной площадки. В этом случае данные будут доступны даже если недоступна ваша сеть. Можно, разумеется, возразить, что нет смысла в поиске адреса узла, если с этим узлом невозможно соединиться. Впрочем, внешний DNS-сервер может быть доступен, если доступна и сама сеть, но прочие DNS-серверы не работают. Организация в Интернете, с которой вы поддерживаете какие-то отношения, например, другой университет или бизнес-партнеры, может согласиться сопровождать вторичный DNS-сервер вашей сети.

Вот пример топологии (рис. 8.1) и краткий анализ, который поможет понять принципы работы.

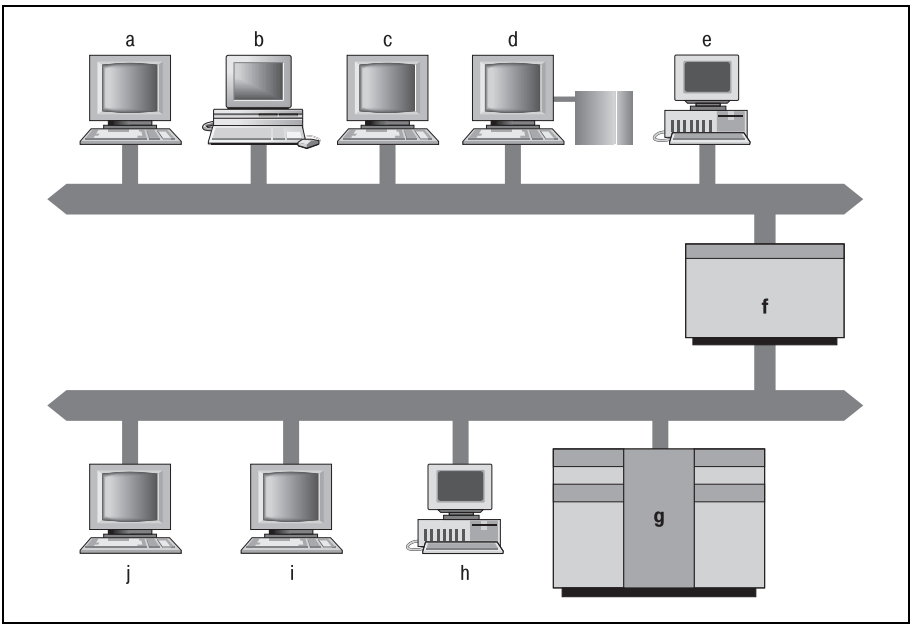

Рис. 8.1. *Пример топологии сети*

Обратите внимание, что если следовать нашим советам, то выбор мест для размещения DNS-серверов по-прежнему сохраняется. Узел *d*, файловый сервер для узлов *a*, *b*, *c* и *e*, может отлично подойти для этой цели. Другой кандидат – узел *g*, мощная машина с большим числом пользователей, входящая в состав нескольких сетей. Но лучшим выбором, вероятно, будет узел *f* – узел помельче, с интерфейсами в обеих

сетях. Можно будет обойтись одним DNS-сервером вместо двух и иметь возможность за ним присматривать. Если нужно иметь более одного сервера в любой из сетей, можно воспользоваться узлами *d* и *g*.

Где размещать DNS-серверы?

В дополнение к примерным представлениям о числе DNS-серверов, эти критерии должны также помочь администратору решить, *где* следует размещать DNS-серверы (к примеру, на файловых серверах и узлах, расположенных в нескольких сетях). Но существуют и другие важные моменты, связанные с выбором того или иного узла.

Факторы, о которых следует помнить: доступность узла, используемое программное обеспечение (BIND или что-то еще), сохранение однородности DNS-серверов, а также безопасность.

Доступность

Очень важно, чтобы DNS-серверы были легкодоступны. Установка DNS-сервера на самом быстром и самом надежном узле сети не принесет никакой пользы, если этот узел находится на задворках сети и подключен к ней через капризное последовательное соединение. Постарайтесь найти узел, расположенный недалеко от канала в сеть Интернет (если таковой присутствует), либо используйте легко доступный узел сети Интернет в качестве вторичного DNS-сервера для зоны. В пределах сети старайтесь размещать DNS-серверы как можно ближе к концентраторам.

Вдвойне важно, чтобы легко доступен был первичный мастер-сервер DNS. Первичному мастер-серверу DNS нужна отличная связь со всеми вторичными серверами, которая обеспечит отсутствие сбоев при синхронизации. Разумеется, как и всякий другой DNS-сервер, первичный мастер только выиграет от работы на быстрых и надежных каналах подключения.

Программное обеспечение

Другой фактор, который следует иметь в виду при выборе узла для DNS-сервера, – программное обеспечение, доступное на этом узле. В этом смысле наилучшим выбором будет версия BIND, поддерживаемая поставщиком системы, – BIND 8.2.3 или 9.1.0, плюс надежная реализация TCP/IP (лучше всего – основанная на сетевом коде системы 4.3/4.4 BSD Unix; мы снобы от Беркли). Можно также собрать BIND 8.2.3 или 9.1.0 из исходных текстов – это не так уж сложно, а более поздние версии очень надежны в работе, но скорее всего никакой поддержки от поставщика используемой операционной системы получить не удастся. Если без возможностей, предоставляемых BIND 8, вполне можно обойтись, имеет смысл попробовать BIND более ранней версии из состава операционной системы, например версию 4.9.7, что позволит получить поддержку фирмы-производителя, если она действительно поможет.

Однородность

Последнее, что следует учитывать – однородность DNS-серверов. Как бы мы не верили в «открытые системы», работа с разнородными вариантами Unix может привести в замешательство и даже отчаяние. Избегайте использования DNS-серверов на различных платформах, если это возможно. Можно потратить массу времени на перенос своих (или наших!) скриптов с одной операционной системы на другую или на поиск места жительства программы *nslookup* или файла *named.conf* в трех различных Unix-системах. Более того, версии Unix от разных поставщиков обычно включают различные версии BIND, что может приводить к всевозможным осложнениям. Если всем DNS-серверам необходимы механизмы безопасности, реализованные в BIND 8 и 9, следует выбрать платформу, поддерживающую BIND 8 или 9, и использовать ее для всех DNS-серверов.

Безопасность

Вне всякого сомнения, мало кому хочется, чтобы взломщик командовал DNS-сервером в целях атаки узлов внутренней сети или других сетей Интернет, поэтому очень важно обеспечить безопасность узла, на котором работает DNS-сервер. Не следует устанавливать DNS-сервер на крупной системе с большим числом пользователей, если этим пользователям нельзя доверять. Если существуют компьютеры, выделенные под сетевые службы, но не позволяющие пользователям работать на них, они являются неплохими кандидатами на установку DNS-серверов. Если действительно защищенных узлов очень мало или вообще один, следует отдать им предпочтение при установке первичного мастер-сервера DNS, поскольку нарушение его защиты приведет к более серьезным последствиям, чем нарушение защиты вторичных DNS-серверов.

Разумеется, все эти соображения являются лишь дополнительными – гораздо более важно, чтобы DNS-сервер существовал в определенной сети, чем то, чтобы он работал на идеальном узле, – но следует иметь их в виду в процессе принятия решений.

Резервирование мощностей

Если речь идет о крупных сетях или пользователях, которые выполняют работу, создающую серьезную нагрузку на DNS-серверы, может оказаться, что необходимо больше DNS-серверов, чем мы рекомендовали. Либо в течение определенного периода времени наши рекомендации могут подходить вам, но по мере добавления узлов и пользователей, по мере установки новых программ, создающих дополнительную нагрузку на DNS-серверы, может оказаться, что DNS-серверы уже не справляются с обработкой запросов.

Какие именно задачи «создают серьезную нагрузку на DNS-сервер»? Веб-серфинг может приводить к созданию такой нагрузки. Отправка

электронной почты, в особенности в крупные списки рассылки, может приводить к созданию такой нагрузки. Программы, выполняющие многочисленные удаленные вызовы процедур (RPC), обращенные к различным узлам, также могут создавать серьезную нагрузку. Даже работа в определенных графических пользовательских средах может подвергать DNS-сервер испытаниям. К примеру, пользовательские среды на основе системы X Window посылают запросы DNS-серверу при проверке списков доступа (среди прочего).

Самые проницательные (и не по годам) умные читатели уже спрашивают: «Как я пойму, что DNS-серверы перегружены? Какие симптомы следует искать?» Хороший вопрос!

Использование памяти, вероятно, наиболее важный аспект работы DNS-сервера, за которым имеет смысл наблюдать. Процесс *named* может становиться очень прожорливым в случае DNS-сервера, авторитативного для многих зон. Если размер *named* и размер прочих работающих процессов в сумме превышают существующий объем физической памяти узла, на узле происходит яростное пережевывание файла подкачки («thrash», пробуксовка), но не выполняется работа. Даже если на узле более чем достаточно памяти для выполнения всех существующих процессов, крупные DNS-серверы медленно запускаются и медленно порождают новые процессы *named* (скажем, при передаче зон). Вторая проблема, специфичная для BIND 4: поскольку DNS-сервер BIND 4 создает новые процессы *named* для передачи зон, вполне возможно одновременное существование нескольких процессов *named* – один из которых реагирует на запросы, а все остальные занимаются передачей данных. Если первичный мастер-сервер DNS и без того потребляет 5 или 10 мегабайт памяти, можно практически гарантировать, что время от времени объем потребляемой памяти будет увеличиваться в два или три раза.

Второй критерий, который можно использовать для измерения нагрузки на DNS-сервер – это нагрузка, которой процесс *named* подвергает процессор узла. Корректно настроенный DNS-сервер обычно не жадничает в смысле процессорного времени, поэтому высокая загрузка процессора часто является признаком ошибки в настройках. Средние показатели загрузки процессора можно получить с помощью программы вроде *top*.[1] К сожалению, не существует конкретных показателей загрузки процессора, которые можно было бы назвать нормальными. Тем не менее мы можем предложить грубое правило: 5% загрузка про-

[1] *top* – это очень удобная программа, созданная Биллом Лефевром, она позволяет получать непрерывно обновляющийся отчет по использованию процессорного времени различными работающими процессами. Последняя версия *top* доступна для анонимного FTP-копирования с сервера *eecs.nwu.edu* (имя файла */pub/top/top-3.4.tar.Z*).

цессора вполне приемлема, 10% – уже великовата, если только узел не выделен под работу DNS.

Чтобы получить представление о нормальных показателях, взгляните на следующий вывод программы *top* для относительно спокойно существующего DNS-сервера:

```
last pid: 14299; load averages: 0.11, 0.12, 0.12       18:19:08
68 processes: 64 sleeping, 3 running, 1 stopped
Cpu states: 11.3% usr, 0.0% nice, 15.3% sys, 73.4% idle, 0.0% intr, 0.0% ker
Memory: Real: 8208K/13168K act/tot Virtual: 16432K/30736K act/tot Free: 4224K

  PID USERNAME PRI NICE   SIZE   RES STATE  TIME   WCPU    CPU COMMAND
   89 root       1    0   2968K 2652K sleep  5:01  0.00%  0.00% named
```

Пожалуй, даже *слишком* спокойный сервер. Вот вывод *top* для занятого (хотя и не перегруженного) DNS-сервера:

```
load averages: 0.30, 0.46, 0.44                  system: relay 16:12:20
39 processes: 38 sleeping, 1 waiting
Cpu states: 4.4% user, 0.0% nice, 5.4% system, 90.2% idle, 0.0% unk5, 0.0%
unk6, 0.0% unk7, 0.0% unk8
Memory: 31126K (28606K) real, 33090K (28812K) virtual, 54344K free Screen #1/ 3

   PID USERNAME PRI NICE  SIZE   RES   STATE   TIME  WCPU   CPU  COMMAND
 21910 root       1    0  2624K  2616K sleep 146:21  0.00% 1.42% /etc/named
```

Второй статистический показатель, на который следует обратить внимание, – это число запросов, получаемых сервером за одну минуту (или за одну секунду, если речь идет о загруженном DNS-сервере). Опять же, нет конкретных цифр: быстрая машина с процессором Pentium III на системе NetBSD, вероятно, способна обрабатывать тысячи запросов в секунду без малейших усилий, а на более старом Unix-узле проблемы могут начаться уже при нескольких запросах в секунду.

Чтобы оценить объем запросов, получаемых DNS-сервером, проще всего взглянуть на внутреннюю статистику этого сервера, которая – при соответствующей настройке DNS-сервера – может регулярно записываться в log-файл демона *syslog*.[1] К примеру, можно настроить DNS-сервер на ежечасное создание статистических отчетов (к слову, это стандартное поведение серверов BIND 8) и сравнить показатели объемов в различные часы работы:

```
options {
        statistics-interval 60;
};
```

[1] Более старые серверы имен BIND нуждаются в специальной команде для создания статистического отчета: сигнале ABRT (IOT на более ранних системах). Серверы имен BIND 4.9 автоматически создают отчет каждый час, но версии с 4.9.4 по 4.9.7 следует принуждать с помощью сигнала ABRT.

В DNS-серверах BIND 9 не поддерживается предписание *statistics-interval*, но можно воспользоваться программами *rndc* и *crontab*, чтобы каждый час давать DNS-серверу BIND 9 команду на создание статистики:

```
0 * * * *  /usr/local/sbin/rndc stats
```

Следует обращать особое внимание на часы пик. Утро понедельника – время повышенной загрузки, поскольку многим людям не терпится ответить на почтовые сообщения, пришедшие за выходные.

Также представляет интерес статистика, полученная непосредственно после ланча – когда люди возвращаются на свои рабочие места и продолжают работу – все в одно и то же время. Разумеется, если организация охватывает несколько часовых поясов, придется воспользоваться здравым смыслом, чтобы определить моменты повышенной загрузки.

Вот выдержка из файла *syslog* DNS-сервера BIND 8.2.3:

```
Aug  1 11:00:49 terminator named[103]: NSTATS 965152849 959476930 A=8 NS=1
SOA=356966 PTR=2 TXT=32 IXFR=9 AXFR=204
Aug  1 11:00:49 terminator named[103]: XSTATS 965152849 959476930 RR=3243
RNXD=0 RFwdR=0 RDupR=0 RFail=20 RFErr=0 RErr=11 RAXFR=204 RLame=0 ROpts=0
SSysQ=3356

SAns=391191 SFwdQ=0 SDupQ=1236 SErr=0 RQ=458031 RIQ=25 RFwdQ=0 RDupQ=0
RTCP=101316 SFwdR=0 SFail=0 SFErr=0 SNaAns=34482 SNXD=0 RUQ=0 RURQ=0 RUXFR=10
RUUpd=34451
Aug  1 12:00:49 terminator named[103]: NSTATS 965156449 959476930 A=8 NS=1
SOA=357195 PTR=2 TXT=32 IXFR=9 AXFR=204
Aug  1 12:00:49 terminator named[103]: XSTATS 965156449 959476930 RR=3253
RNXD=0 RFwdR=0 RDupR=0 RFail=20 RFErr=0 RErr=11 RAXFR=204 RLame=0 ROpts=0
SSysQ=3360

SAns=391444 SFwdQ=0 SDupQ=1244 SErr=0 RQ=458332 RIQ=25 RFwdQ=0 RDupQ=0
RTCP=101388 SFwdR=0 SFail=0 SFErr=0 SNaAns=34506 SNXD=0 RUQ=0 RURQ=0 RUXFR=10
RUUpd=34475
```

Число полученных запросов содержится в поле *RQ* (выделено жирным шрифтом). Чтобы вычислить число запросов, полученных за час, следует вычесть первое значение *RQ* из второго: 458332 – 458031 = 301.

Даже если узел достаточно производителен, чтобы обработать все получаемые запросы, следует убедиться, что DNS-трафик не создает излишней нагрузки на сеть. В большинстве локальных сетей трафик DNS практически незаметен в масштабах существующих каналов, так что о нем можно не беспокоиться. Однако при использовании низкоскоростных выделенных каналов или коммутируемых соединений трафик DNS вполне может стать проблемой, занимая собой значительную часть полосы пропускания.

Чтобы произвести грубую оценку объема DNS-трафика в локальной сети, следует умножить сумму числа полученных запросов (RQ) и от-

правленных ответов (SAns) – за один час – на 800 битов (100 байтов – примерный средний размер сообщения DNS) и разделить на 3600 (секунд в одном часе). Таким образом можно приблизительно понять, какой процент полосы пропускания занят трафиком DNS.[1]

Мы попытаемся дать читателям представление о нормальных показателях. Последний отчет NSFNET по трафику (в апреле 1995 года) показал, что трафик DNS составил чуть больше 5% суммарного объема трафика (в байтах) магистрали этой сети. Показатели, приводимые в отчете NSFNET, основаны на выборке из конкретного трафика, а не на вычислениях по статистическим показателям DNS-сервера.[2] Если необходимо получить более точное предоставление о трафике, получаемом DNS-сервером, можно произвести собственную выборку по трафику с помощью протокольного анализатора для локальной сети.

Итак, мы выяснили, что DNS-серверы переутомляются. Что дальше? Во-первых, разумно будет проверить, что DNS-серверы не подвергаются бомбардировкам запросами со стороны некорректно работающей программы. Для этого придется всего лишь выяснить, откуда поступают запросы.

Если используется DNS-сервер BIND 4.9 или 8.1.2, можно выяснить, какие клиенты и DNS-серверы генерируют запросы путем создания статистического отчета. DNS-серверы этих версий хранят статистику взаимодействий с отдельными узлами, что весьма полезно при необходимости отследить причины высокой загруженности DNS-серверов. Серверы BIND 8.2 и более поздних версий по умолчанию не хранят такую статистику, но могут быть настроены с помощью предписания *host-statistics* в операторе *options*:[3]

```
options {
    host-statistics yes;
};
```

Рассмотрим для примера следующий отчет:

```
+++ Statistics Dump +++ (829373099) Fri Apr 12 23:24:59 1996
970779    time since boot (secs)
```

---

[1] Если нужен удобный пакет, позволяющий автоматизировать анализ статистики BIND, обратите внимание на разработку Найджела Кэмпбелла (Nigel Campbell) – инструмент *bindgraph*, информация о котором доступна на странице программного обеспечения каталога ресурсов DNS, по адресу *http://www.dns.net/dnsrd/tools.html*.

[2] Мы не знаем точно, насколько эти показатели отражают текущее состояние дел в сети Интернет, но невероятно трудно добиться раскрытия подобной информации от коммерческих провайдеров сетевых магистралей, которые пришли вслед за NSFNET.

[3] Кстати говоря, BIND 9 не поддерживает предписание *host-statistics* и создание статистики для отдельных узлов на момент существования версии 9.1.0.

```
471621    time since reset (secs)
0    Unknown query types
185108    A queries
6    NS queries
69213    PTR queries
669    MX queries
2361    ANY queries
++ Name Server Statistics ++
 (Legend)
    RQ      RR      RIQ      RNXD      RFwdQ
    RFwdR   RDupQ   RDupR    RFail     RFErr
    RErr    RTCP    RAXFR    RLame     ROpts
    SSysQ   SAns    SFwdQ    SFwdR     SDupQ
    SFail   SFErr   SErr     RNotNsQ   SNaAns
    SNXD
(Global)
    257357 20718 0 8509 19677  19939 1494 21 0 0  0 7 0 1 0
    824 236196 19677 19939 7643  33 0 0 256064 49269  155030
  [15.17.232.4]
    8736 0 0 0 717  24 0 0 0 0  0 0 0 0 0  0 8019 0 717 0
    0 0 0 8736 2141  5722
 [15.17.232.5]
    115 0 0 0 8  0 21 0 0 0  0 0 0 0 0  0 86 0 1 0  0 0 0 115 0  7
 [15.17.232.8]
    66215 0 0 0 6910  148 633 0 0 0  0 5 0 0 0  0 58671 0 6695 0
    15 0 0 66215 33697  6541
 [15.17.232.16]
    31848 0 0 0 3593  209 74 0 0 0  0 0 0 0 0  0 28185 0 3563 0
    0 0 0 31848 8695  15359
 [15.17.232.20]
    272 0 0 0 0  0 0 0 0 0  0 0 0 0 0  0 272 0 0 0  0 0 0 272 7  0
 [15.17.232.21]
    316 0 0 0 52  14 3 0 0 0  0 0 0 0 0  0 261 0 51 0  0 0 0 316 30  30
 [15.17.232.24]
    853 0 0 0 65  1 3 0 0 0  0 2 0 0 0  0 783 0 64 0  0 0 0 853 125  337
 [15.17.232.33]
    624 0 0 0 47  1 0 0 0 0  0 0 0 0 0  0 577 0 47 0  0 0 0 624 2  217
 [15.17.232.94]
    127640 0 0 0 1751  14 449 0 0 0  0 0 0 0 0  0 125440 0 1602 0
    0 0 0 127640 106  124661
 [15.17.232.95]
    846 0 0 0 38  1 0 0 0 0  0 0 0 0 0  0 809 0 37 0  0 0 0 846 79  81
-- Name Server Statistics --
--- Statistics Dump --- (829373099) Fri Apr 12 23:24:59 1996
```

Вслед за записью *Global* записи для отдельных узлов привязаны к IP-адресам этих узлов, заключаемым в квадратные скобки. Взглянув на легенду, можно понять, что первое поле в каждой записи содержит значение RQ, то есть определяет число полученных запросов. Так что мы получаем повод внимательно присмотреться к узлам 15.17.232.8,

15.17.232.16 и 15.17.232.94, которые отвечают за 88% полученных запросов.

Если используется более старый DNS-сервер, единственный способ узнать, какие именно клиенты и DNS-серверы являются авторами всех этих проклятых запросов, – включать отладку DNS-сервера. (Мы рассмотрим эту тему подробно в главе 13 «Чтение отладочного вывода BIND».) Интерес на практике представляют только IP-адреса, от которых исходят запросы, адресованные DNS-серверу. При изучении отладочного вывода следует обращать внимание на узлы, посылающие повторные запросы, в особенности повторные запросы одной и той же информации. Это может свидетельствовать о неправильно настроенной, либо некорректно работающей программе, используемой на таком узле, либо о том, что внешний DNS-сервер бомбардирует ваш сервер запросами.

Если все запросы выглядят обоснованно, добавьте новый DNS-сервер. Однако не стоит размещать его абы где, следует использовать полученную отладочную информацию при принятии решения. Если трафик DNS пожирает каналы локальной сети, нет смысла выбирать узел для размещения DNS-сервера случайным образом. Следует выяснить, какие узлы посылают большой объем запросов, а затем решить, как лучше всего обеспечить для них работу службы имен. Вот несколько советов, которые могут помочь принять решение:

- Ищите запросы от клиентов узлов, работающих с одним и тем же файл-сервером. DNS-сервер может быть установлен на этот файл-сервер.
- Ищете запросы от клиентов крупных узлов с большим числом пользователей. DNS-серверы могут размещаться на таких узлах.
- Ищите запросы от клиентов из другой подсети. Эти клиенты следует настроить на работу с DNS-сервером локальной подсети. Если в подсети не существует DNS-сервера, следует его создать.
- Ищите запросы от клиентов в том же сегменте (если используются коммутаторы). Если DNS-сервер будет работать в одном сегменте с клиентами, исчезнет необходимость коммутировать трафик через всю сеть.
- Ищите запросы от узлов, подключенных через другую, слабо загруженную сеть. DNS-сервер может быть размещен в этой другой сети.

Добавление DNS-серверов

Когда возникает необходимость в создании новых DNS-серверов для зон, самый легкий выход – создать новые вторичные серверы. Как это делается, читатели уже знают, мы рассказывали об этом в главе 4; а проведя одну установку вторичного DNS-сервера, можно затем клони-

ровать его без особого труда. Но беспорядочно плодя дополнительные DNS-серверы, можно нарваться на неприятности.

Если создано большое число вторичных DNS-серверов для зоны, может случиться так, что первичный мастер-сервер DNS будет с трудом справляться с периодическими запросами дополнительных DNS-серверов, желающих проверить, не пора ли производить синхронизацию. Для этого случая существует несколько вариантов дальнейших действий:

- Создать новые первичные мастер-серверы DNS.
- Увеличить интервал обновления, чтобы вторичные серверы реже производили проверку актуальности данных.
- Предписать отдельным вторичным DNS-серверам синхронизироваться с другими вторичными серверами, а не с первичными мастерами.
- Создать DNS-серверы, специализирующиеся на кэшировании (этот вид серверов будет описан позже).
- Создать «частично вторичные» DNS-серверы (аналогично).

Первичные и вторичные серверы

Создание новых первичных мастер-серверов DNS означает увеличение нагрузки на администратора, поскольку синхронизацию файлов */etc/named.conf* и файлов данных зон придется производить вручную. Разумеется, администратор сам решает, является ли эта альтернатива приемлемой. Для упрощения процесса синхронизации файлов могут использоваться инструменты вроде *rdist* и *rsync*.[1] Файл *distfile*[2] для синхронизации файлов контрольных серверов может выглядеть очень просто:

```
dup-primary:

# копировать named.conf в файловую систему двойника

/etc/named.conf  -> wormhole
    install ;

# копировать содержимое /var/named (файлы данных зон и все прочие) в файловую
систему двойника

/var/named -> wormhole
    install ;
```

[1] *rsync* – это инструмент для удаленной синхронизации файлов, который передает только различия между файлами. Более подробная информация о программе доступна по адресу *http://rsync.samba.org*.

[2] Этот файл содержит информацию о файлах, которые программа *rdist* должна обновлять.

либо – для нескольких контрольных серверов:

```
dup-primary:

primaries =  ( wormhole carrie )
/etc/named.conf  -> {$primaries}
     install ;

/var/named -> {$primaries}
     install ;
```

Более того, можно заставить *rdist* выполнять перезагрузку DNS-серверов, используя инструкцию *special* следующим образом:

```
special /var/named/* "ndc reload" ;
special /etc/named.conf "ndc reload" ;
```

rdist выполняет команду в кавычках при изменении любого из перечисленных файлов.

Второй вариант – увеличение интервала обновления для зоны. Но это несколько замедляет распространение новой информации. В некоторых случаях такое замедление приемлемо. Если данные зоны создаются с помощью *h2n* только раз в сутки в час ночи (по расписанию программы *cron*), а затем данные расходятся в течение шести часов, все дополнительные серверы будут обладать актуальной информацией уже к семи часам утра.[1] Этого может быть вполне достаточно для пользователей. Дополнительная информация содержится в разделе «Изменение прочих значений SOA-записи» далее по тексту главы.

Можно заставить некоторые из вторичных DNS-серверов синхронизироваться с другими вторичными серверами. Вторичные DNS-серверы *способны* загружать данные зоны, получаемые от других вторичных DNS-серверов, а не от первичного мастера. Вторичный DNS-сервер не в состоянии понять, с какого именно сервера он получает зону. Важно одно: чтобы DNS-сервер, обеспечивающий синхронизацию, являлся авторитативным для зоны. Никаких сложностей в настройке здесь не возникает. Вместо указания IP-адреса первичного мастер-сервера DNS в файле настройки вторичного просто указывается IP-адрес другого вторичного сервера.

Вот содержимое файла *named.conf*:

```
// этот вторичный DNS-сервер синхронизируется с узлом wormhole,
// который также является вторичным
zone "movie.edu" {
                 type slave;
                 masters { 192.249.249.1; };
                 file "bak.movie.edu";
};
```

[1] Разумеется, при использовании NOTIFY – гораздо раньше.

Для сервера BIND 4 настройки будут выглядеть несколько иначе.

Вот содержимое файла *named.boot*:

```
; этот вторичный DNS-сервер синхронизируется с узлом wormhole,
; который также является вторичным
secondary   movie.edu   192.249.249.1   bak.movie.edu
```

Однако при переходе на второй уровень распределения данных время распространения данных с первичного мастер-сервера DNS на все дополнительные может увеличиться вдвое. Следует помнить, что *интервал обновления* – это период, после истечения которого дополнительные DNS-серверы производят проверку актуальности данных хранимых зон. Таким образом, вторичные DNS-серверы первого уровня могут бездействовать в течение всего интервала обновления, прежде чем получить новую копию зоны от первичного мастер-сервера DNS. Точно так же вторичные серверы второго уровня могут бездействовать в течение всего интервала обновления, прежде чем получить новую копию зоны от вторичных DNS-серверов первого уровня. Таким образом, время распространения информации на все вторичные DNS-серверы может быть вдвое больше, чем интервал обновления.

Одним из способов избавиться от подобной задержки является использование механизма NOTIFY, реализованного в BIND 8 и 9. По умолчанию механизм работает и обеспечивает получение зоны вторичными серверами имен вскоре после ее изменения на первичном мастере. К сожалению, этот механизм работает только со вторичными DNS-серверами версий 8 или 9.[1] Более подробно мы рассмотрим механизм NOTIFY в главе 10 «Дополнительные возможности».

Если вы используете два или более уровней обновления для вторичных DNS-серверов, будьте бдительны и не создавайте петель обновления. Если бы мы настроили узел *wormhole* на синхронизацию с *diehard*, а затем по ошибке настроили *diehard* на синхронизацию с *wormhole*, ни один из них никогда бы не синхронизировался с первичным мастер-сервером DNS. Они просто сличали бы устаревшие порядковые номера своих хранимых зон и всю жизнь считали бы хранимые данные актуальными.

DNS-серверы, специализирующиеся на кэшировании

Создание DNS-серверов, специализирующихся на кэшировании, – еще одна альтернатива для случаев, когда необходимо увеличить число DNS-серверов. Специальные кэширующие DNS-серверы не являются авторитативными ни для каких зон, кроме *0.0.127.in-addr.arpa*. Такое название вовсе не означает, что первичный мастер и вторичные

[1] А еще так получилось с сервером Microsoft DNS.

DNS-серверы не занимаются кэшированием – как раз наоборот. Речь идет о том, что *единственная* функция, которую выполняет такой сервер, – поиск и кэширование данных. Такому серверу, как и любому другому, для работы требуется файл корневых указателей и файл *db.127.0.0*. Файл *named.conf* для специальных кэширующих DNS-серверов содержит следующие строки:

```
options {
    directory "/var/named";  // каталог с данными
};

zone "0.0.127.in-addr.arpa" {
    type master;
    file "db.127.0.0";
};

zone "." {
    type hint;
    file "db.cache";
};
```

В случае сервера BIND 4 *named.boot* выглядит следующим образом:

```
directory /var/named  ; каталог с данными

primary 0.0.127.in-addr.arpa  db.127.0.0  ; для адреса обратной связи
cache   .                     db.cache
```

Специальный кэширующий DNS-сервер способен – как и всякий другой сервер – производить поиск для имен, принадлежащих зоне, и для любых других. Разница состоит в том, что при первом поиске для имени из локальной зоны кэширующий сервер в итоге обращается к одному из первичных мастеров или вторичных DNS-серверов зоны. Первичный или вторичный DNS-сервер ответил бы на тот же вопрос, исходя из своей авторитативности для данных. К какому первичному или вторичному DNS-серверу обращается специальный кэширующий сервер? Как в случае поиска за пределами зоны, он запрашивает список серверов, обслуживающих локальную зону, у DNS-серверов родительской зоны. Существует ли способ произвести первичное заполнение кэша специального кэширующего DNS-сервера, чтобы он «знал», на каких узлах работают DNS-серверы вашей зоны? Нет. Невозможно использовать файл *db.cache* – он должен содержать только корневые указатели. И вообще говоря, лучше, чтобы кэширующий DNS-сервер узнавал об авторитативных DNS-серверах вашей зоны от DNS-серверов родительской зоны: так он всегда будет обладать актуальной информацией о делегировании. Если же вручную заполнить перечень авторитативных DNS-серверов, используемых кэширующим сервером, можно впоследствии забыть, что этот перечень тоже необходимо обновлять.

Настоящую ценность кэширующий DNS-сервер представляет, когда произошло заполнение кэша. Каждый раз, посылая запрос авторита-

тивному DNS-серверу и получая ответ, специальный сервер кэширует записи из ответа. Через некоторое время кэш наполняется информацией, которая наиболее часто запрашивается клиентами, посылающими запрос этому серверу. При этом отсутствует нагрузка, связанная с необходимостью получения зоны, поскольку специальным кэширующим DNS-серверам не нужны хранимые файлы зоны.

Частично вторичные DNS-серверы

Между специальными кэширующими и вторичными DNS-серверами существует промежуточный вид: DNS-сервер, который является вторичным лишь для нескольких локальных зон. Мы называем такой сервер *частично вторичным* (скорее всего, только мы его так и называем). Предположим, *movie.edu* состоит из двадцати сетей размера /24 (бывший класс C) и, соответственно, 20 зон *in-addr.arpa*. Вместо того чтобы создавать вторичный DNS-сервер для 21 зоны (все поддомены *in-addr.arpa* и *movie.edu*), мы можем создать частично вторичный сервер для *movie.edu* и лишь тех зон *in-addr.arpa*, в которые входит собственно узел. Если бы узел имел два сетевых интерфейса, DNS-сервер являлся бы вторичным для трех зон: *movie.edu* и двух зон *in-addr.arpa*.

Допустим, мы раздобыли машину для нового DNS-сервера. Назовем новый узел именем *zardoz.movie.edu* и присвоим ему IP-адреса 192.249.249.9 и 192.253.253.9. С помощью следующего файла *named.conf* мы создадим на узле *zardoz* частично вторичный DNS-сервер:

```
options {
    directory "/var/named";
};

zone "movie.edu" {
    type slave;
    masters { 192.249.249.3; };
    file "bak.movie.edu";
};

zone "249.249.192.in-addr.arpa" {
    type slave;
    masters { 192.249.249.3; };
    file "bak.192.249.249";
};

zone "253.253.192.in-addr.arpa" {
    type slave;
    masters { 192.249.249.3; };
    file "bak.192.253.253";
};

zone "0.0.127.in-addr.arpa" {
    type master;
    file "db.127.0.0";
```

```
};

zone "." {
    type hint;
    file "db.cache";
};
```

Файл *named.boot* для DNS-сервера BIND 4 выглядел бы так:

```
directory   /var/named
secondary   movie.edu                 192.249.249.3 bak.movie.edu
secondary   249.249.192.in-addr.arpa 192.249.249.3 bak.192.249.249
secondary   253.253.192.in-addr.arpa 192.249.249.3 bak.192.253.253
primary     0.0.127.in-addr.arpa      db.127.0.0
cache       .                         db.cache
```

Этот сервер является вторичным для *movie.edu* и двух из двадцати зон *in-addr.arpa*. Файл *named.conf* для «полного» вторичного DNS-сервера содержал бы 21 оператор *zone*.

Чем же так полезен частично вторичный DNS-сервер? Такие DNS-серверы просты в администрировании, поскольку файлы *named.conf* не особо меняются. На DNS-сервере, авторитативном для всех зон *in-addr.arpa*, пришлось бы добавлять и удалять зоны *in-addr.arpa* (и соответствующие записи в файле *named.conf*) в процессе эволюции сети. В больших сетях это может выливаться в потрясающе большие объемы работы.

При этом частично вторичный сервер способен отвечать на большинство запросов. Большинство этих запросов будет связано с данными из *movie.edu* и двух зон *in-addr.arpa*. Почему? Потому что большинство узлов, посылающих запросы этому DNS-серверу, принадлежат сетям, с которыми он напрямую связан: 192.249.249 и 192.253.253. И эти узлы, вероятно, взаимодействуют в основном с узлами из своих собственных сетей. Это служит причиной создания запросов для данных из зоны *in-addr.arpa*, соответствующей конкретной сети.

Регистрация DNS-серверов

Как только вы приметесь за активное создание новых и новых DNS-серверов, может возникнуть вопрос: неужели необходимо регистрировать *все* первичные мастера и вторичные DNS-серверы в родительской зоне? Не все, а только те DNS-серверы, которые необходимо сделать доступными для внешних DNS-серверов. К примеру, если существует девять DNS-серверов, обслуживающих зону, можно рассказать родительской зоне лишь о четырех из них. В пределах внутренней сети будут использоваться все девять серверов. Пять из девяти DNS-серверов будут использоваться только соответствующим образом настроенными (скажем, с помощью файла *resolv.conf*) клиентами узлов. DNS-серверы родительской зоны не делегируют локальную зону этим пяти

DNS-серверам, поэтому им никогда не будут поступать запросы от удаленных серверов. Лишь четыре сервера, зарегистрированных в родительской зоне, будут получать запросы от других DNS-серверов, включая специальные кэширующие и частично вторичные DNS-серверы внутренней сети. Структура отображена на рис. 8.2.

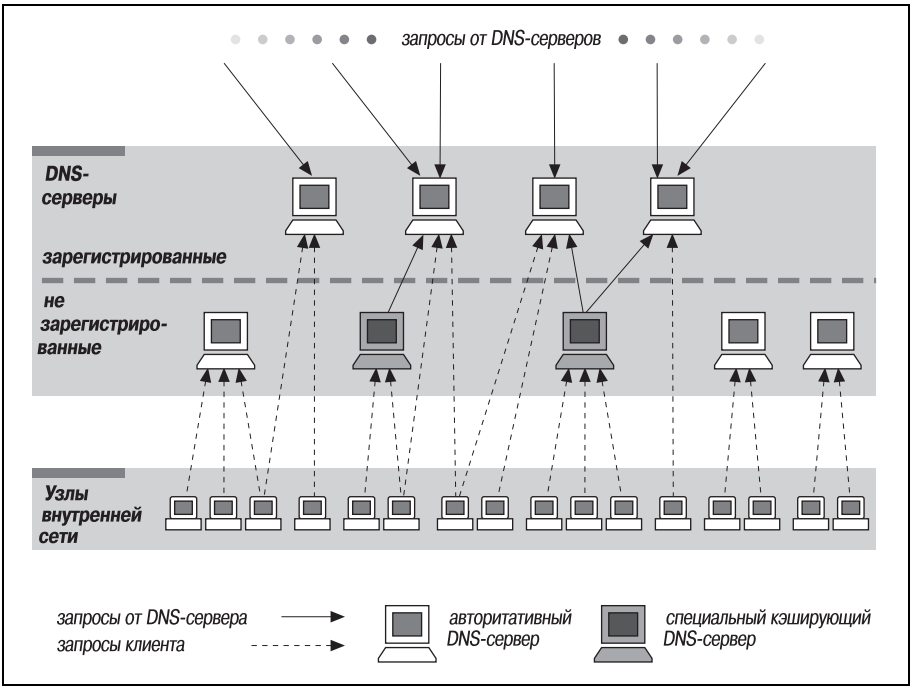

Рис. 8.2. *Регистрация отдельных DNS-серверов*

Помимо возможности выбирать, какие именно серверы будут подвергаться бомбардировке запросами извне, существует и техническое обоснование для регистрации лишь нескольких DNS-серверов зоны: количество DNS-серверов, информация о которых может поместиться в ответное сообщение UDP, ограничено. На практике сообщение может вместить примерно 10 NS-записей; в зависимости от данных (количества серверов в одном домене), может помещаться больше или меньше.[1] В любом случае нет особого смысла в регистрации более чем десяти DNS-серверов – если ни один из них не доступен, маловероятно, что доступен искомый узел.

[1] По этой причине были изменены доменные имена корневых серверов имен сети Интернет. Все корневые серверы были переведены в один домен, *root-servers.net*, чтобы по максимуму использовать преимущества упаковки доменных имен и предоставлять в одном UDP-пакете информацию о максимальном числе корневых серверов имен.

Если после установки нового авторитативного DNS-сервера администратор приходит к выводу, что сервер следует зарегистрировать, необходимо создать перечень родителей зон, для которых данный DNS-сервер является авторитативным. Придется связаться с администраторами всех родительских зон. Допустим, мы решили зарегистрировать только что созданный DNS-сервер *zardoz.movie.edu*. Чтобы зарегистрировать этот вторичный узел во всех нужных зонах, мы должны связаться с администраторами зон *edu* и *in-addr.arpa*. (Если необходим совет по идентификации контактных лиц для родительских зон, обратитесь к главе 3 «С чего начать?»)

Вступая в контакт с администраторами родительской зоны, старайтесь следовать (возможно) существующему протоколу, который обычно публикуется на веб-сайте зоны. Если стандартный процесс внесения изменений не регламентирован, следует послать администраторам доменное имя зоны (или имена зон), для которых новый DNS-сервер является авторитативным. Если новый DNS-сервер расположен в новой зоне, следует также сообщить IP-адрес(а) этого сервера. Вообще говоря, не существует официально утвержденного формата для передачи информации такого рода, так что обычно наилучшим решением является посылка родителям полного перечня зарегистрированных для зоны DNS-серверов (с адресами, при необходимости) в формате файла данных зоны. Это позволит избежать возможной путаницы.

Поскольку наши сети изначально распределялись организацией InterNIC, мы воспользовались веб-интерфейсом по адресу *http://www.arin.net/cgi-bin/amt.pl* для внесения изменений в данные регистрации. (Если бы мы предпочитали делать это вручную, то могли бы отправить почтой заполненную форму *http://www.arin.net/regserv/templates/modifytemplate.txt*.) А не будь у нас готового к использованию шаблона, наше сообщение, обращенное к администратору *in-addr.arpa*, могло бы выглядеть так:

```
Привет!

Я только что создал новый вторичный DNS-сервер на узле zardoz.movie.edu (зоны
249.249.192.in-addr.arpa и 253.253.192.in-addr.arpa) В связи с этим прошу
добавить NS-записи нового DNS-сервера к данным зоны in-addr.arpa.
В результате, наша информация о делегировании будет выглядеть следующим
образом:

253.253.192.in-addr.arpa. 86400 IN NS terminator.movie.edu.
253.253.192.in-addr.arpa. 86400 IN NS wormhole.movie.edu.
253.253.192.in-addr.arpa. 86400 IN NS zardoz.movie.edu.

249.249.192.in-addr.arpa. 86400 IN NS terminator.movie.edu.
249.249.192.in-addr.arpa. 86400 IN NS wormhole.movie.edu.
249.249.192.in-addr.arpa. 86400 IN NS zardoz.movie.edu.

Спасибо!

Albert LeDomaine
al@robocop.movie.edu
```

Нетрудно заметить, что мы явным образом указали значения TTL для NS- и A-записей. Дело в том, что родительские DNS-серверы не являются авторитативными для этих записей, в отличие от *наших* DNS-серверов. Включая эти значения, мы показываем свои предпочтения относительно времени жизни информации о делегировании. Разумеется, у родителя могут быть свои представления о предпочтительных значениях TTL.

В данном случае связующие данные – адресные записи для каждого из DNS-серверов – не являются необходимыми, поскольку доменные имена DNS-серверов не принадлежат зонам *in-addr.arpa*. Они принадлежат зоне *movie.edu*, поэтому DNS-сервер, перенаправленный к серверу *terminator.movie.edu* или *wormhole.movie.edu*, может определить их адреса, используя информацию о делегировании для DNS-серверов *movie.edu*.

Имеет ли смысл регистрация частично вторичного DNS-сервера в родительской зоне? Не очень большой, поскольку этот сервер является авторитативным лишь для нескольких зон *in-addr.arpa*. С точки зрения администрирования проще зарегистрировать только DNS-серверы, которые обслуживают *все* локальные зоны; в этом случае нет необходимости отслеживать, какие из DNS-серверов являются авторитативными для конкретных зон. Все родительские зоны могут производить делегирование синхронизированному набору DNS-серверов: первичному мастеру и «полным» вторичным.

Разумеется, администратор, в подчинении которого не так много DNS-серверов, либо администратор, который в состоянии запомнить, какая ответственность возложена на каждый из серверов, вполне может взять и зарегистрировать частично вторичный DNS-сервер.

С другой стороны, специальные кэширующие DNS-серверы *никогда* не должны регистрироваться. Кэширующий DNS-сервер редко обладает полной информацией хотя бы для одной зоны, его знания в основном ограничиваются отрывочными данными, для которых недавно выполнялся поиск. Если DNS-серверы родительской зоны по ошибке направляют «иностранца» к такому DNS-серверу, иностранец посылает нерекурсивный запрос. Специальный кэширующий DNS-сервер может найти ответ в кэшированных данных, но может и не найти. В последнем случае, он просто перенаправит автора запроса к лучшему из известных DNS-серверов (то есть расположенных ближе всего к искомому доменному имени) – и при этом сам может оказаться таким сервером! Бедный иностранец может так и не добиться ответа. Такой вид неправильной настройки, которая заключается в делегировании зоны DNS-серверу, не являющемуся авторитативным для этой зоны, известен как *некорректное делегирование* (*lame delegation*).

Изменение значений TTL

Опытный администратор должен знать, каким образом выбирать значения времени жизни для данных зоны в целях получения наилучшего результата. Вспомним, что значение TTL для RR-записи – это период времени, в течение которого произвольному DNS-серверу разрешается кэшировать эту запись. Если значение TTL для отдельной RR-записи установлено в 3600 секунд и запись кэшируется сервером за пределами внутренней сети, этот сервер обязан удалить запись через час. Если по прошествии часа эти данные снова потребуются, внешнему серверу придется повторно послать запрос одному из внутренних DNS-серверов.

Рассказывая о значениях TTL, мы особо подчеркнули, что выбор TTL определяет скорость распространения данных, которая непосредственно связана с нагрузкой на DNS-серверы. Низкое значение TTL означает, что внешние DNS-серверы будут чаще обращаться к внутренним DNS-серверам, а значит, будут обладать наиболее свежими данными. При этом DNS-серверы внутренней сети будут в большей степени загружены запросами извне.

Но *необязательно* выбирать значения TTL раз и навсегда. Абсолютно допустимо – чем и пользуются опытные администраторы – периодически изменять значения TTL в зависимости от насущных нужд.

Предположим, нам известно, что один из наших узлов будет переведен в другую сеть. На этом узле хранится библиотека фильмов *movie.edu*, огромная коллекция файлов, которые наша площадка делает доступными пользователям сети Интернет. В процессе нормальной работы внешние DNS-серверы кэшировали адрес этого узла, учитывая значение TTL по умолчанию, определенное директивой $TTL, либо – для DNS-серверов более ранних, чем BIND версии 8.2, – SOA-записью. Время жизни по умолчанию для *movie.edu* в наших примерах устанавливалось в три часа. DNS-сервер, кэшировавший старую адресную запись непосредственно перед внесением изменений, в течение трех часов будет сообщать пользователям неправильный адрес. Недоступность узла в течение трех часов вполне приемлема. Что можно сделать, чтобы минимизировать подобные эффекты? Можно уменьшить значение TTL, чтобы внешние DNS-серверы кэшировали адресную запись на более короткие промежутки времени. Таким образом, мы заставляем внешние DNS-серверы обновлять данные более часто, поэтому любые вносимые нами изменения очень быстро дойдут до внешнего мира. Насколько большим можно сделать значение TTL? К сожалению, невозможно спокойно задать нулевое значение TTL, которое означает полное отсутствие кэширования. Некоторые из более старых DNS-серверов BIND 4 не умеют возвращать нулевые TTL и потому возвращают пустые ответы или ошибки SERVFAIL. Однако небольшие значения TTL, скажем 30 секунд, вполне допустимы. Самое простое изменение:

уменьшить значение TTL в директиве $TTL в файле *db.movie.edu*. Если время жизни не задано явным образом для RR-записей в файле данных зоны, DNS-сервер использует именно это *TTL по умолчанию* для каждой записи. Однако если уменьшить значение TTL по умолчанию, новое значение будет использовано для всех данных зоны, не только для адреса узла, который переезжает. Минус этого подхода в том, что DNS-сервер будет загружен гораздо большим числом запросов, поскольку *все* данные зоны будут кэшироваться DNS-серверами на гораздо более короткие периоды времени. Более приемлемая альтернатива – изменить TTL только для одной адресной записи.

Чтобы явно задать значение TTL для отдельной записи, следует поместить его перед идентификатором класса (IN). По умолчанию значение определяется в секундах, но существует возможность определить единицы измерения: *m* (минуты), *h* (часы), *d* (дни) и *w* (недели); точно так же, как в директиве $TTL. Вот пример явного указания значения TTL из файла *db.movie.edu*:

```
cujo  1h IN  A    192.253.253.5  ; явно определено, что TTL = 1 часу
```

Наблюдательные читатели, заглянувшие в документ RFC 1034, могут заметить потенциальную проблему, которая может возникнуть при загрузке этой записи более старым DNS-сервером: явно определенное значение TTL для адреса *cujo* – 1 час, но поле TTL SOA-записи, которое якобы определяет минимальное значение TTL для зоны в DNS-серверах более ранних, чем BIND 8.2, содержит большее значение. Какое из них используется?

Если бы старые DNS-серверы BIND скрупулезно следовали заветам исходного RFC-документа по DNS, поле TTL SOA-записи действительно определяло бы минимальное значение TTL для всех RR-записей зоны. Таким образом, можно было бы явным образом определять только те значения TTL, которые больше указанного минимума. Однако старые DNS-серверы BIND работают не совсем так. Иначе говоря, «минимум» в BIND – не совсем минимум. BIND интерпретирует поле TTL в SOA-записи как TTL «по умолчанию». (Разумеется, в более новых версиях BIND значение TTL по умолчанию устанавливается с помощью оператора $TTL.) Если значение TTL для записи не определено, используется значение по умолчанию. В противном случае используется определенное для записи значение, даже если оно меньше допустимого минимума. Так что в нашем случае одна запись включается в ответы с меньшим временем жизни, а все прочие – с минимальным, которое определяется SOA-записью.

Следует также знать, что вторичный DNS-сервер, возвращая ответы, использует те же значения TTL, что и первичный мастер-сервер DNS: если мастер-сервер DNS возвращает определенную запись со значением TTL в 1 час, так же будет поступать и вторичный. Вторичный DNS-сервер не уменьшает TTL в соответствии с тем, сколько времени прош-

ло с момента загрузки зоны. Так что если значение TTL для отдельной записи установлено меньше минимального, то и первичный и вторичный DNS-серверы будут возвращать эту запись с одинаковым, меньшим минимума, значением времени жизни. Если хранимая зона на вторичном DNS-сервере устаревает, то она перестает использоваться целиком. Отдельные записи в пределах зоны не могут устаревать.

Итак, BIND позволяет определять меньшие значения TTL для отдельных RR-записей, если известно, что данные скоро изменятся. Таким образом, любой DNS-сервер, кэширующий данные, запоминает их лишь на короткое время. К сожалению, немногие администраторы тратят время на использование этой возможности, поэтому часто, при изменении адреса узла, этот узел становится временно недоступен.

Чаще всего происходит смена обычного узла, а не одного из главных серверов площадки, поэтому перебой в работе затрагивает всего несколько человек. Если же речь идет о переезде крупного веб-сервера или FTP-архива (вроде библиотеки фильмов) – отсутствие связи с узлом в течение целого дня совершенно неприемлемо. В таких случаях администратор должен планировать изменения заранее и вовремя изменять значения TTL для данных, которым предстоит измениться.

Помните, что значение TTL для изменяемых данных следует понижать *до* внесения изменений. Уменьшение TTL адресной записи рабочей станции и параллельное изменение адреса не даст желаемого эффекта: адресная запись могла быть кэширована за несколько секунд до внесения изменений и будет бродить по свету, пока не истечет ее время жизни. *Кроме того*, не забудьте учесть время, которое потребуется для синхронизации вторичных серверов с первичным мастером. Так, если значение TTL по умолчанию установлено в 12 часов, а интервал обновления – 3 часа, следует уменьшить значение TTL по меньшей мере за 15 часов до времени внесения изменений, чтобы к моменту переезда узла все старые записи с более высокими значениями TTL устарели. Разумеется, если все дополнительные DNS-серверы – это DNS-серверы BIND 8 или 9, которые используют NOTIFY, дополнительные серверы синхронизируются гораздо раньше, чем это определено интервалом обновления.

Изменение прочих значений SOA-записи

Мы кратко упомянули увеличение интервала обновления в качестве способа разгрузить первичный мастер-сервер DNS. Обсудим обновления чуть более подробно и рассмотрим все остальные значения SOA-записи.

Как помнят читатели, значение интервала *обновления* (*refresh*) определяет, насколько часто вторичный узел проверяет актуальность данных хранимой зоны. Значение интервала *повторения попытки* (*retry*) становится интервалом обновления после того, как первая попытка вто-

ричного узла связаться с первичным мастером заканчивается неудачей. Значение интервала *устаревания* (*expire*) определяет, как долго могут храниться данные зоны перед их удалением в случае недоступности основного сервера. И наконец, для DNS-серверов до BIND 8.2 значение TTL *по умолчанию* определяет допустимый период кэширования информации. Более новые DNS-серверы интерпретируют последнее поле SOA-записи как значение TTL для отрицательных ответов.

Предположим, мы решили, что вторичные DNS-серверы должны синхронизироваться с первичным каждый час, а не каждые три часа. Поэтому мы изменили значение обновления на *1h* в каждом из файлов данных зоны (либо воспользовались ключом *-o* программы *h2n*). Поскольку интервал повторения попытки связан с интервалом обновления, его тоже следует уменьшить – примерно до 15 минут. Обычно интервал повторения меньше интервала обновления, но это необязательно.[1] Понижение значения обновления ускорит распространение новых данных, но увеличит нагрузку на DNS-сервер, распространяющий данные, поскольку вторичные DNS-серверы будут чаще производить проверку. Однако в данном случае дополнительная нагрузка не очень велика: каждый вторичный DNS-сервер делает один запрос SOA-записи в пределах интервала обновления каждой из зон, чтобы сравнить свою копию с копией, хранимой основным сервером. Для двух вторичных DNS-серверов уменьшение времени обновления с трех часов до одного приведет к созданию лишь четырех дополнительных запросов (для каждой зоны) к первичному мастер-серверу DNS на произвольный трехчасовой интервал времени.

Разумеется, если все дополнительные серверы являются серверами BIND 8 или 9 и при этом используется механизм NOTIFY, обновление не имеет такого значения. Но если хотя бы один вторичный DNS-сервер является сервером BIND 4, может истечь полный интервал обновления, прежде чем данные будут синхронизированы.

Некоторые из более ранних версий сервера BIND – при использовании их в качестве вторичных – перестают отвечать на запросы в процессе получения зоны. В результате код BIND был модифицирован с целью распределения нагрузки, создаваемой синхронизацией зон, и сокращения периодов недоступности дополнительных серверов. Поэтому даже если установить короткий интервал обновления, вторичный сервер может принять самостоятельное решение о более редких проверках актуальности хранимых данных. DNS-сервер BIND 4 совершает определенное число попыток получения зоны, а затем делает перерыв на 15 минут, прежде чем повторить серию. С другой стороны, обновление на DNS-серверах BIND версии 4.9 и более поздних может происхо-

[1] Вообще говоря, серверы имен BIND 8 выдают предупреждение, если установлен интервал повторения более чем десятикратно превышающий интервал обновления.

дить более часто, чем это определено интервалом обновления. Такие версии BIND перед сверкой серийных номеров делают паузу размером в случайное число секунд из интервала от половины до полного периода обновления.

Время устаревания порядка недели используется часто – более длинные интервалы имеют смысл, если случаются рецидивы проблем связи с источником обновлений. Время устаревания всегда должно быть намного больше, чем интервалы обновления и повторения; если время устаревания меньше интервала обновления, данные, хранимые вторичными серверами, будут устаревать еще до проверки их актуальности. BIND 8 выдаст сообщение, если время устаревания меньше суммы интервалов обновления и повторения, меньше удвоенного интервала повторения, меньше семи дней или больше шести месяцев. (BIND 9.1.0 в этом случае пока что молчит.) Выбор времени устаревания, соответствующего всем критериям BIND 8, в большинстве случаев является разумным решением.

Если данные зоны меняются редко, имеет смысл подумать об увеличении значения TTL по умолчанию. Значение TTL по умолчанию традиционно лежит в интервале от нескольких часов до суток, но можно использовать и большие значения. Одна неделя – практический предел, который имеет смысл для значения TTL. Более длительное время жизни может привести к тому, что некорректные кэшированные данные не будут удалены за разумное время.

Подготовка к бедствиям

Суровая правда жизни заключается в том, что в сети неизбежно возникают проблемы. Аппаратное обеспечение сбоит, программное обеспечение содержит дефекты, а люди время от времени совершают ошибки. Иногда это приводит к легким неудобствам, например, к потере соединений несколькими пользователями. А иногда – к катастрофическим результатам, связанным с потерей ценных данных и рабочих мест.

Поскольку DNS сильно зависит от сети, она уязвима для сбоев в сети. К счастью, несовершенство сетей было учтено при разработке DNS: система позволяет создавать избыточные DNS-серверы, ретранслировать запросы, повторять попытки получения зон и т. д.

Но DNS не защищена от всякого мыслимого бедствия. Существуют различные виды сетевых сбоев, причем довольно часто встречаются и такие, от которых система DNS не защищена. Но затратив немного времени и денег, можно свести к минимуму подобные опасности.

Перебои электроэнергии

Перебои электроэнергии довольно распространены во многих частях света. В некоторых районах США грозы и торнадо могут приводить к

потере электроснабжения площадки либо к перебоям электроэнергии на длительные периоды времени. В других местах к подобным эффектам могут приводить тайфуны, вулканы либо строительные работы.

Разумеется, если ни один узел внутренней сети не работает, служба доменных имен ни к чему. Однако довольно часто на площадке возникает проблема при *восстановлении* питания. Следуя нашим рекомендациям, администраторы устанавливают DNS-серверы на файл-серверы и крупные узлы с большим числом пользователей. И когда электроснабжение восстанавливается, эти машины, само собой, завершают загрузку последними, поскольку все эти огромные диски необходимо, прежде всего, проверить программой *fsck*! Так что все узлы на площадке, которые загрузятся быстро, будут вынуждены ждать, когда наконец заработает служба имен.

Это может приводить к разнообразным и забавным видам осложнений, в зависимости от того, как написаны загрузочные скрипты для системы, под управлением которой работает узел. Узлы Unix в целях настройки сетевого интерфейса и добавления маршрута по умолчанию зачастую выполняют следующие команды (с небольшими вариациями):

```
/usr/sbin/ifconfig lan0 inet `hostname` netmask 255.255.128.0 up
/usr/sbin/route add default site-router 1
```

Использование имен узлов в командах (вместо '*hostname*' подставляется локальное имя узла, а *site-router* – имя локального маршрутизатора) замечательно по двум причинам:

- Оно позволяет администраторам изменять IP-адрес маршрутизатора, не изменяя загрузочные файлы для всех машин площадки.
- Оно позволяет администраторам изменять IP-адрес узла путем изменения адреса в единственном файле.

К сожалению, команда *route* не работает в отсутствие службы имен. Команда *ifconfig* не работает только в случае, если имя локального узла и его IP-адрес отсутствуют в локальном файле */etc/hosts*, поэтому разумно оставлять в файлах */etc/hosts* различных узлов, по меньшей мере, эту информацию.

Ко времени, когда загрузка дойдет до выполнения команды *route*, сетевой интерфейс будет уже настроен, и узел попытается воспользоваться службой имен для преобразования имени маршрутизатора в IP-адрес. И поскольку маршрут по умолчанию отсутствует до выполнения команды *route*, узел сможет связаться только с DNS-серверами локальной подсети.

Если загружающийся узел успешно установил контакт с DNS-сервером в локальной подсети, команда *route* может быть выполнена успешно. Но чаще всего один или несколько DNS-серверов, с которыми мог бы связаться узел, еще не запущены. Что происходит в таком случае, зависит от содержимого файла *resolv.conf*.

DNS-клиенты BIND перейдут к использованию таблицы узлов, если в файле *resolv.conf* указан лишь один DNS-сервер (если DNS-серверы не определены в файле, клиент по умолчанию пытается обратиться к DNS-серверу локального узла). В таком случае клиент посылает запрос этому серверу, и при неоднократном получении ошибок переходит к поиску в таблице узлов. Перечислим ошибки, которые могут приводить к такому поведению:

- Получение ICMP-сообщения о недоступности порта (port unreachable).
- Получение ICMP-сообщения о недоступности сети (network unreachable).
- Отсутствие возможности послать пакет UDP (к примеру, сетевые службы локального узла еще не запущены).[1]

Если узел одного из DNS-серверов, указанных в файле *resolv.conf*, не работает вовсе, клиент не получает сообщений об ошибках. DNS-сервер в этом случае является черной дырой. Через 75 секунд попыток истекает интервал ожидания, и клиент возвращает приложению пустой ответ. Клиент может получить ICMP-сообщение о недоступности порта только в том случае, если на узле DNS-сервера уже запущены сетевые службы, но не DNS-сервер.

В итоге использование единственного DNS-сервера является вполне работоспособным вариантом, если в каждой сети существуют собственные DNS-серверы, но не столь элегантным, как нам хотелось бы. Если локальный DNS-сервер не был запущен после перезагрузки узла в той же сети, команда *route* не может быть успешно выполнена.

Это может казаться неудобным, но гораздо лучше варианта нескольких DNS-серверов. Если в файле *resolv.conf* перечислено несколько DNS-серверов, BIND *никогда* не перейдет к использованию таблицы узлов после того, как была выполнена команда *ifconfig* для основного сетевого интерфейса. Клиент просто перебирает DNS-серверы, посылая им запросы, пока не будет получен ответ либо достигнут конец интервала ожидания.

Это представляет особую проблему в процессе загрузки. Если ни один из перечисленных DNS-серверов не доступен, интервал ожидания клиента истекает без получения ответа, поэтому создания маршрута по умолчанию не происходит.

---

[1] Глава 6 «Конфигурирование узлов» содержит данные по дополнениям, существующим в различных системах, и вариантам алгоритма работы анализатора.

Советы

Как бы примитивно это не звучало, наш совет – явным образом определить IP-адрес маршрутизатора по умолчанию в файле загрузки либо в другом внешнем файле (во многих системах используется файл */etc/defaultrouter*). Это позволит произвести корректный запуск сетевых служб.

В качестве альтернативного варианта: можно указывать в файле *resolv.conf* лишь один, но очень надежный DNS-сервер в локальной для узла сети. Это позволит использовать в загрузочных файлах имя маршрутизатора по умолчанию, разумеется, при наличии имени маршрутизатора в */etc/hosts* (на тот случай, если надежный узел еще не работает при перезагрузке узла). Конечно, если хост надежного DNS-сервера еще не загружен при перезагрузке локального узла, можно считать, что песенка спета. Не будет никакого перехода к использованию */etc/hosts*, поскольку сетевые службы не смогут даже вернуть ошибку.

Если BIND, поставляемый в составе системы, позволяет определять порядок использования служб либо переходит к работе с файлом */etc/hosts* в случаях, когда ответ не может быть получен с помощью DNS, пользуйтесь этим обстоятельством! В первом случае можно настроить клиент таким образом, чтобы он прежде всего обращался к файлу */etc/hosts*, а затем создать «заглушку» */etc/hosts* на каждом узле, включив в файл имена стандартного маршрутизатора и локального узла. В последнем случае следует просто убедиться, что такой файл-заглушка существует; дополнительные настройки не нужны.

Есть еще одна многообещающая перспектива – избавиться от установки маршрута по умолчанию с помощью механизма *ICMP Router Discovery Messages*. Это расширение протокола ICMP, описанное в документе RFC 1256, использует броадкастовые (широковещательные) или мультикастовые пакеты для динамического обнаружения маршрутизаторов сети и распространения информации о них. Это расширение поддерживается системой Windows NT 4.0, но по умолчанию отключено. Чтобы привести его в рабочее состояние, следует воспользоваться инструкциями из статьи Q223756 базы знаний Microsoft. Sun включает реализацию этого протокола в последние версии системы Solaris в виде */usr/sbin/in.rdisc*; протокол также поддерживается операционной системой IOS (Internetwork Operating System) от Cisco.

Что произойдет в случае, если маршрут по умолчанию создан корректно, а DNS-серверы все еще не запущены? Это может вызвать проблемы у *sendmail*, NFS и массы других служб. Без DNS *sendmail* не сможет производить корректную канонизацию имен узлов, а монтирование по NFS не приведет к положительным результатам.

Лучшее решение – разместить DNS-сервер на узле с бесперебойным питанием. Если перебои в подаче электроэнергии происходят редко,

аккумуляторного резерва может вполне хватать. Если перебои более длительны, а работа службы имен жизненно необходима, следует задуматься об установке системы бесперебойного питания (UPS, Uninterruptible Power System) с генератором.

Если подобная роскошь недоступна, можно просто найти узел, который загружается быстрее прочих, и разместить DNS-сервер на нем. Узлы с журналируемой файловой системы должны загружаться особенно быстро, поскольку на таких узлах нет необходимости в работе *fsck*. Узлы с небольшими дисками также загружаются быстро, поскольку проверяемая файловая система гораздо меньше.

Обнаружив «правильный» узел, убедитесь, что IP-адрес этого узла присутствует в файле *resolv.conf* на всех узлах, которым круглосуточно необходима служба имен. При этом рекомендуется указывать резервный узел последним в списке, поскольку при нормальной работе узлы должны использовать наиболее близко расположенные DNS-серверы. В таком варианте после перебоев электроснабжения критические приложения будут иметь доступ к службе имен, пусть и ценой некоторого снижения производительности.

Борьба с бедствиями

Когда бедствие наносит удар, очень полезно знать, что именно следует делать. Если укрыться под крепким столом во время землетрясения, это будет способствовать тому, чтобы не быть придавленным упавшим монитором. Умение выключать газ может спасти дом от пожара.

Точно так же уверенные действия при бедствии в сети (пусть даже это мелкая неприятность) могут сохранить сеть в рабочем состоянии. Мы живем в Калифорнии, поэтому можем поделиться своим опытом и некоторыми соображениями.

Кратковременные перебои (на часы)

Если сеть отрезана от внешнего мира (где под «внешним миром» понимается сеть Интернет или остальные сети компании), DNS-серверы начинают испытывать проблемы при разрешении доменных имен. Скажем, если наш домен *corp.acme.com* отрезан от оставшейся части интернет-сети Acme, может отсутствовать связь с DNS-серверами родительской зоны (*acme.com*) или с корневыми DNS-серверами.

Казалось бы, это не должно повлиять на взаимодействие узлов в пределах локального домена, но дела обстоят несколько иначе. Если набрать команду:

```
% telnet selma.corp.acme.com
```

на узле с более старой версией клиента, первое доменное имя, для которого произведет поиск клиент, – *selma.corp.acme.com.corp.acme.com* (мы исходим из предположения, что на узле используется список поиска по умолчанию, – мы говорили об этой особенности в главе 6). Локальный DNS-сервер, если он является авторитативным для *corp.acme.com*, может сообщить, что доменное имя является неправильным. Но следующий поиск будет произведен для имени *selma.corp.acme.com.acme.com*. Это имя уже не принадлежит зоне *corp.acme.com*, поэтому запрос передается DNS-серверам *acme.com*. Вернее, локальный DNS-сервер *пытается* отправить им запрос, производя переключения, пока не истечет интервал ожидания.

Избежать этой проблемы можно, сделав так, чтобы первое доменное имя, для которого производится поиск, было правильным. Вместо команды:

```
% telnet selma.corp.acme.com
```

лучше набрать:

```
% telnet selma
```

либо:

```
% telnet selma.corp.acme.com.
```

(Обратите внимание на последнюю точку.) В результате первым именем, для которого производится поиск, будет *selma.corp.acme.com*.

По умолчанию в клиентах BIND 4.9 и более поздних версий эта проблема отсутствует. Начиная с версии 4.9, клиенты сначала производят буквальный поиск для введенного имени, если имя содержит хотя бы одну точку. В этом случае, если набрать:

```
% telnet selma.corp.acme.com
```

даже без точки в конце имени, будет сразу произведен поиск для *selma.corp.acme.com*.

Если приходится работать с клиентом версии 4.8.3 или более ранней, можно избежать поиска внешних имен с помощью настройки списка поиска. Для определения списка поиска, не содержащего доменного имени родительской зоны, можно воспользоваться инструкцией *search*. К примеру, чтобы решить затруднения для зоны *corp.acme.com*, можно временно определить список поиска следующим образом:

```
search corp.acme.com
```

В этом случае, если пользователь набирает:

```
% telnet selma.corp.acme.com
```

клиент производит поиск сначала для имени *selma.corp.acme.com.corp.acme.com* (на этот запрос способен ответить локальный

DNS-сервер), а затем для существующего доменного имени *selma.corp.acme.com*. Вот такой вариант тоже замечательно сработает:

```
% telnet selma
```

Перебои средней длительности (на дни)

При потере сетевого подключения на длительное время у DNS-серверов могут появляться проблемы иного рода. Если нет связи с корневыми DNS-серверами в течение долгого времени, DNS-серверы перестают производить разрешение запросов, которые не касаются данных в пределах авторитативности. Если дополнительные серверы не могут связаться с основным, рано или поздно произойдет устаревание хранимой зоны.

Если же служба имен разваливается при потере подключения, есть смысл иметь под рукой общий для площадки или рабочей группы файл */etc/hosts*. Во времена крайней нужды можно переименовать файл *resolv.conf* в *resolv.bak*, остановить локальный DNS-сервер (если таковой существует) и использовать только таблицу узлов – */etc/hosts*. Не особо модно, но в критической ситуации спасает.

Что касается вторичных DNS-серверов, их можно временно перенастроить на работу в качестве первичных мастеров, если связи с основными серверами нет. Достаточно отредактировать *named.conf* и изменить значение в предписании *type* в операторе *zone* со *slave* на *master*, а затем удалить предписание *masters*. Если «отрезанных» вторичных DNS-серверов для зоны несколько, можно временно сделать один из них первичным мастером, а все остальные настроить на синхронизацию с ним.

Как вариант можно увеличить время устаревания во всех резервных копиях зон вторичных DNS-серверов, а затем скомандовать серверам произвести перезагрузку файлов.

Долговременные перебои (на недели)

В случае долговременных (от недели) перебоев связи с сетью Интернет может возникать необходимость в искусственном восстановлении связи с корневыми DNS-серверами при приведении системы в рабочее состояние. Каждому DNS-серверу время от времени необходимо общаться с корневым DNS-сервером. Это своего рода терапия: DNS-сервер должен поговорить с корневым сервером, чтобы снова достичь устойчивого видения мира.

Чтобы обеспечить наличие корневых DNS-серверов во время длительных перебоев, можно создать собственные корневые DNS-серверы, *но только временно*. Как только подключение к сети Интернет будет восстановлено, администратор *должен* остановить свои корневые DNS-серверы. Самые неприятные паразиты сети Интернет – это DNS-серве-

ры, которые считают себя корневыми, но ничего не знают о большинстве доменов высшего уровня. На втором месте – сервер имен Интернет, настроенный на работу с поддельным набором корневых DNS-серверов.

Итак, мы обеспечили себе алиби и сказали вступительные слова, теперь следует настроить наши собственные корневые DNS-серверы. Во-первых следует создать файл *db.root*, содержащий данные корневой зоны. Файл *db.root* будет содержать информацию о делегировании зон высшего уровня для наших изолированных сетей. К примеру, если бы зона *movie.edu* была отрезана от сети Интернет, мы создали бы для узла *terminator* файл *db.root* следующего содержания:

```
$TTL 1d
. IN SOA terminator.movie.edu. al.robocop.movie.edu. (
                  1        ; Порядковый номер
                  3h       ; Обновление
                  1h       ; Повторение
                  1w       ; Устаревание
                  1h )     ; Отрицательное TTL

  IN NS terminator.movie.edu. ; terminator является временным ИО
                              ; корневого DNS-сервера

; Он знает только о movie.edu и паре
; доменов in-addr.arpa

movie.edu. IN NS terminator.movie.edu.
           IN NS wormhole.movie.edu.

249.249.192.in-addr.arpa. IN NS terminator.movie.edu.
                          IN NS wormhole.movie.edu.

253.253.192.in-addr.arpa. IN NS terminator.movie.edu.
                          IN NS wormhole.movie.edu.

terminator.movie.edu. IN A 192.249.249.3
wormhole.movie.edu.   IN A 192.249.249.1
                      IN A 192.253.253.1
```

Затем необходимо добавить соответствующие строки в файл *named.conf* узла *terminator*:

```
// Комментируем зону корневых указателей
// zone . {
//              type hint;
//                      file "db.cache";
//              };

zone "." {
                 type master;
                 file "db.root";
};
```

Или для BIND 4 – в файл *named.boot*:

```
; cache    .   db.cache  (комментируем инструкцию cache)
primary  .   db.root
```

Затем необходимо обновить файл *db.cache* на всех серверах (кроме нашего нового корневого) его новым вариантом, включающим только временный корневой DNS-сервер (предыдущие версии файлов корневых указателей лучше всего просто переименовать, поскольку они понадобятся нам позже, когда будет восстановлено подключение).

Вот содержимое файла *db.cache*:

```
.  99999999  IN  NS  terminator.movie.edu.

terminator.movie.edu.  99999999   IN  A  192.249.249.3
```

Такая настройка позволит производить разрешение имен в *movie.edu* во время перебоев. Когда связь с сетью Интернет восстановится, мы можем удалить новый оператор *zone* из *named.conf*, раскомментировать оператор *zone* для корневых указателей на узле *terminator*, а также восстановить исходные файлы корневых указателей на всех прочих DNS-серверах.

9

Материнство

- *Когда заводить детей*
- *Сколько детей?*
- *Какие имена давать детям*
- *Заводим детей: создание поддоменов*
- *Поддомены in-addr.arpa*
- *Заботливые родители*
- *Как справиться с переходом к поддоменам*
- *Жизнь родителя*

А знаешь, как Дина умывала своих котят? Одной лапой она хватала бедняжку за ухо и прижимала к полу, а другой терла ей всю мордочку, начиная с носа, против шерсти. Как я уже сказал, в это время она трудилась над Снежинкой, а та лежала смирно, не сопротивлялась, да еще пыталась мурлыкать – видно, понимала, что все это делается для ее же блага.

Когда домен достигает определенных размеров, администратор приходит к мысли, что управление сегментами домена имеет смысл распределить между различными объектами организации. Домен придется разделить на поддомены. Поддомены будут детьми существующего домена в пространстве имен; домен будет их родителем. Если администратор делегирует ответственность за поддомены другим организациям, каждый из них становится отдельной зоной. Мы будем называть сопровождение поддоменов-детей – *родительскими заботами*.

Заботливый родитель начинает с разумного разделения домена, выбора подходящих имен поддоменов и делегирования поддоменов с целью создания новых зон. Ответственный родитель старается изо всех сил, чтобы сохранить хорошие отношения между зоной и детьми, он следит за тем, чтобы делегирование от родителя ребенку было актуальным и корректным.

Заботливый администратор – ключевая фигура для процветания сети, особенно когда служба имен становится незаменимой для связи между площадками. Некорректное делегирование DNS-серверам порожденной зоны может сделать узлы этой зоны попросту недоступными, а потеря связи с DNS-серверами родительской зоны может отрезать поддомен от узлов, расположенных за пределами зоны.

В этой главе мы приводим свое мнение о том, когда следует создавать поддомены, и чуть более подробно рассматриваем процесс создания и делегирования. Помимо этого мы затронем сохранение отношений родителя и ребенка, и наконец минимизацию неудобств и проблем в процессе разбиения крупных доменов на более мелкие поддомены.

Когда заводить детей

У нас и в мыслях нет *учить* читателей, когда следует заводить детей, но мы возьмем на себя смелость предложить некоторые соображения на эту тему. Кое-кто, возможно, найдет вескую причину для создания поддоменов, которой нет в этом списке, но вот наиболее распространенные:

- Необходимость делегировать или разделить управление доменом между несколькими организациями.
- Излишнее укрупнение домена – разделение облегчит сопровождение и сократит нагрузку на авторитативные DNS-серверы.
- Необходимость различать принадлежность узлов различным организациям путем включения их в соответствующие поддомены.

Решив обзавестись детьми, мы – само собой – должны спросить себя: а сколько нужно детей?

Сколько детей?

Конечно же, нельзя просто сказать: «Хочу создать четыре поддомена». Определение числа доменов связано с их предназначением. К примеру, если у компании есть четыре местных офиса, то можно создать для каждого по отдельному поддомену.

Следует ли создавать отдельный домен для каждой площадки, для каждого отдела, для каждого факультета? Масштабируемость DNS создает определенную свободу выбора. Можно создать несколько крупных поддоменов или много маленьких. В любом варианте приходится идти на определенные компромиссы.

Делегирование нескольких крупных поддоменов – задача для родителя не очень трудоемкая, поскольку не так много информации о делегировании, которую необходимо сопровождать. Однако при этом приходится работать с крупными поддоменами, которые требуют больше памяти и большей скорости работы DNS-серверов, а также не позволяют разделять работу между большим числом администраторов. Если создать для каждой из существующих площадок отдельный поддомен, автономные или неродственные группы узлов этой площадки будут вынуждены проживать в одной зоне – с централизованным администрированием.

Многочисленные делегированные поддомены могут стать головной болью для администратора родительской зоны. Сопровождение информации о делегировании связано с отслеживанием узлов и работающих на них DNS-серверов, а также с вопросами авторитативности этих серверов. Каждый раз при появлении нового DNS-сервера в поддомене или при изменении адреса DNS-сервера эта информация изменяется. Если все поддомены находятся в ведении различных людей, увеличивается число людей, которых необходимо обучать, увеличивается число связей, поддерживаемых администратором родительской зоны, увеличивается организационная нагрузка в целом. С другой стороны, поддомены, поскольку они мельче, становятся проще в сопровождении, происходит рассредоточение административных прав, и данным каждой зоны в этом случае уделяется больше внимания.

Учитывая преимущества и недостатки обоих вариантов, может быть нелегко сделать выбор. Хотя в действительности в каждой организации наверняка существует какое-то естественное деление. Некоторые компании управляют компьютерами и сетями на уровне площадки, в других существуют децентрализованные, автономные рабочие группы, которые занимаются всеми вопросами самостоятельно. Вот несколько основных правил, которые помогут определиться с разделением пространства имен:

- Не пытайтесь запихнуть организацию в неестественную или неудобную структуру. Попытки поместить 50 независимых, ничем не связанных штатов США в четыре региональных поддомена могут уменьшить трудозатраты (администратора родительской зоны), но навряд ли положительно повлияют на репутацию. Децентрализация и автономная работа требует существования многих зон – такова государственная политика DNS.
- Структура домена должна отражать структуру организации, в особенности *опорную* структуру. Если факультеты сами создают свои сети, распределяют свои IP-адреса и занимаются сопровождением своих узлов, на факультеты и должна быть возложена ответственность за сопровождение поддоменов.
- Если нет полной уверенности относительно того, какой вид должно иметь пространство имен, постарайтесь создать нормативные принципы для случаев, когда группа в пределах организации желает выделиться в отдельный поддомен (скажем, минимальное число узлов для выделения в поддомен, уровень поддержки, который должна обеспечивать эта группа) – после чего позволить пространству имен органично расти – по мере необходимости.

Какие имена давать детям

Определившись с числом поддоменов и сущностями, которым поддомены соответствуют, следует подобрать удачные имена. Не очень веж-

ливо давать доменам имен в одностороннем порядке; лучше всего привлечь к решению этого вопроса администраторов будущих поддоменов и их подданных. Более того, можно предоставить им полную свободу в этом вопросе.

Однако такая политика может приводить к осложнениям. Наиболее эффективно использовать согласованную систему именования для всех поддоменов. Это облегчает угадывание и запоминание имен поддоменов, а также поиск узлов и пользователей в различных поддоменах – как для пользователей поддомена, так и для пользователей из внешнего мира.

Если имена придумываются местными властями, это может привести к хаосу с именами. Некоторым захочется зарегистрировать «географические» имена, другие будут настаивать на именах, отражающих название организации, с которой связан домен. Одни будут заниматься сокращением, прочие станут использовать полные имена.

Таким образом, перед тем как начать давать имена поддоменам, будет логично создать правила именования. Вот некоторые соображения из нашего собственного опыта:

- В динамичной компании имена организаций могут часто меняться. В этом случае привязка поддоменов к объектам компании может привести к катастрофе. В этом месяце группа «Относительно современные технологии» выглядит достаточно стабильно, а следующем может произойти поглощение этой группы организацией «Сомнительные компьютерные системы», а в следующем квартале обе они будут приобретены немецким конгломератом. Между тем, вы уже привязаны к конкретным узлам в поддоменах, имена которых более не имеют смысла.
- Географические имена более стабильны, но иногда не столь прозрачны. Администратору, быть может, известно, что его любимое «Бизнес-подразделение компьютерного евангелизма» находится в Паукипси (Poughkeepsie) или Уокигане (Waukegan), но люди, не работающие в компании, могут понятия не иметь, где это, и испытывать сложности при написании подобных названий.
- Не приносите читабельность в жертву удобству. Двухбуквенные имена поддоменов, конечно, проще набирать, но они совершенно ни о чем не говорят. Какой смысл сокращать «Italy» (Италию) до букв «it» и путать с Организацией информационных технологий (IT), когда ценой всего лишь трех дополнительных букв можно уничтожить всякую двусмысленность и пользоваться полным названием?
- Очень многие компании используют загадочные, неудобные имена. Общее правило примерно таково: чем больше компания, тем хуже поддаются расшифровке доменные имена. Не поддавайтесь тенденции – делайте имена поддоменов самоочевидными!

- Не используйте имена существующих или зарезервированных доменов высшего уровня в качестве имен поддоменов. Использование двухбуквенных кодов стран в качестве имен международных поддоменов или использование имени вроде *net* для организации, деятельность которой связана с сетями, может показаться неплохой идеей, но также может привести и к неприятностям. Если дать поддомену отдела коммуникаций имя *com* это может затруднить взаимодействие с узлами домена высшего уровня *com*. Представим, что администратор поддомена *com* назвал новую рабочую станцию Sun именем *sun*, а новый HP 9000 – *hp* (налицо некоторые проблемы с воображением). Пользователи домена, отправляющие почтовые сообщения своим друзьям из доменов *sun.com* и *hp.com*, могут обнаружить, что их письма неожиданно попали в поддомен *com*[1], поскольку доменное имя родительской зоны может присутствовать в списке поиска одного из узлов.

Заводим детей: создание поддоменов

Когда имена придуманы, остается только создать поддомены, что довольно просто. Но прежде необходимо решить, сколько автономии дать новым поддоменам. Странно, что приходится об этом думать *до* создания поддоменов...

До сих пор мы предполагали, что после создания поддомен будет делегирован другой организации, таким образом превращаясь в самостоятельную зону. Но всегда ли это так? Необязательно.

Принимая решение о делегировании поддомена, следует тщательно обдумать вопрос сопровождения машин и сетей этого поддомена. Нет смысла делегировать поддомен объекту, который не сопровождает собственные узлы и сети. К примеру, в крупной корпорации отдел кадров, видимо, не занимается сопровождением компьютеров; скорее всего, за это отвечает отдел информационных систем управления или отдел информационных технологий. Поэтому при создании поддомена для отдела кадров делегирование управления этим поддоменом – по всей видимости, потеря времени и сил.

Создание поддоменов в родительской зоне

Итак, можно создать поддомен, но не делегировать его. Как это сделать? С помощью RR-записей, относящихся к поддомену в родительской зоне. Предположим, в зоне *movie.edu* существует узел *brazil*, на

[1] Вообще говоря, эта проблема существует не во всех почтовых программах, но она присутствует в некоторых весьма распространенных версиях *sendmail*. Все зависит от используемой формы канонизации, как мы уже говорили в разделе «Электронная почта» главы 6 «Конфигурирование узлов».

котором хранится полная база данных по служащим и студентам. Чтобы поместить *brazil* в домен *personnel.movie.edu*, можно создать соответствующие записи в файле *db.movie.edu*.

Вот фрагмент *db.movie.edu*:

```
brazil.personnel      IN  A      192.253.253.10
                      IN  MX     10 brazil.personnel.movie.edu.
                      IN  MX     100 postmanrings2x.movie.edu.
employeedb.personnel  IN  CNAME  brazil.personnel.movie.edu.
db.personnel          IN  CNAME  brazil.personnel.movie.edu.
```

Теперь пользователи могут соединяться с узлом *db.personnel.movie.edu* для получения доступа к базе данных по служащим. Мы могли бы сделать это изменение особенно удобным для работников отдела кадров, добавив *personnel.movie.edu* в списки поиска их рабочих станций; это позволит набирать *telnet db* для получения доступа к узлу базы данных.

Мы можем упростить жизнь и себе, используя директиву $ORIGIN для изменения суффикса по умолчанию на *personnel.movie.edu*.

Фрагмент файла *db.movie.edu*:

```
$ORIGIN personnel.movie.edu.
brazil     IN A     192.253.253.10
           IN MX    10 brazil.personnel.movie.edu.
           IN MX    100 postmanrings2x.movie.edu.
employeedb IN CNAME brazil.personnel.movie.edu.
db         IN CNAME brazil.personnel.movie.edu.
```

Если бы записей было больше, мы могли бы вынести их в отдельный файл и включать его в *db.movie.edu* с помощью директивы $INCLUDE (в то же время меняя локально суффикс по умолчанию).

Обратили внимание, что SOA-запись для *personnel.movie.edu* отсутствует? В ней нет необходимости, поскольку SOA-запись *movie.edu* сообщает о начале авторитативности для всей зоны *movie.edu*. Поскольку нет делегирования *personnel.movie.edu*, поддомен является частью зоны *movie.edu*.

Создание и делегирование поддомена

Когда администратор принимает решение делегировать поддомены – дать детям путевку в жизнь, – требуется несколько иной подход. Поскольку мы находимся в процессе демонстрации, читатели могут к нам присоединяться.

Необходимо создать в зоне *movie.edu* поддомен для лаборатории специальных эффектов. Мы выбрали имя *fx.movie.edu* – краткое, прозрачное, недвусмысленное. Поскольку мы делегируем *fx.movie.edu* администраторам лаборатории, поддомен будет являться самостоятельной зоной. Узлы *bladerunner* и *outland*, расположенные в лаборатории, бу-

дут выступать в качестве DNS-серверов этой зоны (причем *bladerunner* будет первичным мастер-сервером DNS). Мы решили, в целях повышения надежности, установить два DNS-сервера для этой зоны – если выйдет из строя единственный DNS-сервер *fx.movie.edu*, вся лаборатория специальных эффектов будет, по сути дела, отрезана от внешнего мира. Но поскольку в лаборатории не так уж и много узлов, двух серверов будет вполне достаточно.

Лаборатория специальных эффектов расположена в новой сети *movie.edu* – 192.253.254/24 network.

Вот фрагмент файла */etc/hosts*:

```
192.253.254.1 movie-gw.movie.edu movie-gw
# fx: первичный
192.253.254.2 bladerunner.fx.movie.edu bladerunner br
# fx: вторичный
192.253.254.3 outland.fx.movie.edu outland
192.253.254.4 starwars.fx.movie.edu starwars
192.253.254.5 empire.fx.movie.edu empire
192.253.254.6 jedi.fx.movie.edu jedi
```

Сначала создадим файл данных зоны, который содержит записи для всех узлов *fx.movie.edu*.

Содержимое файла *db.fx.movie.edu*:

```
$TTL 1d
@  IN  SOA  bladerunner.fx.movie.edu. hostmaster.fx.movie.edu. (
            1       ; порядковый номер
            3h      ; обновление
            1h      ; повторение
            1w      ; устаревание
            1h )    ; отрицательное TTL

   IN  NS  bladerunner
   IN  NS  outland

; MX-записи fx.movie.edu
   IN  MX  10 starwars
   IN  MX  100 wormhole.movie.edu.

; узел starwars обрабатывает почту узла bladerunner
; wormhole - почтовый концентратор movie.edu

bladerunner  IN  A   192.253.254.2
             IN  MX  10 starwars
             IN  MX  100 wormhole.movie.edu.

br           IN    CNAME    bladerunner

outland      IN  A   192.253.254.3
             IN  MX  10 starwars
             IN  MX  100 wormhole.movie.edu.
```

```
starwars     IN  A   192.253.254.4
             IN  MX  10 starwars
             IN  MX  100 wormhole.movie.edu.

empire       IN  A   192.253.254.5
             IN  MX  10 starwars
             IN  MX  100 wormhole.movie.edu.

jedi         IN  A   192.253.254.6
             IN  MX  10 starwars
             IN  MX  100 wormhole.movie.edu.
```

Теперь следует создать файл *db.192.253.254*:

```
$TTL 1d
@  IN  SOA  bladerunner.fx.movie.edu. hostmaster.fx.movie.edu. (
             1       ; порядковый номер
             3h      ; обновление
             1h      ; повторение
             1w      ; устаревание
             1h )    ; отрицательное TTL

       IN    NS    bladerunner.fx.movie.edu.
       IN    NS    outland.fx.movie.edu.
1      IN    PTR   movie-gw.movie.edu.
2      IN    PTR   bladerunner.fx.movie.edu.
3      IN    PTR   outland.fx.movie.edu.
4      IN    PTR   starwars.fx.movie.edu.
5      IN    PTR   empire.fx.movie.edu.
6      IN    PTR   jedi.fx.movie.edu.
```

Обратите внимание, что PTR-запись для *1.254.253.192.in-addr.arpa* указывает на имя *movie-gw.movie.edu*. Это сделано умышленно. Этот маршрутизатор связан и с другими сетями *movie.edu* и в действительности не принадлежит *fx.movie.edu*; к тому же, не существует правила, по которому все PTR-записи в *254.253.192.in-addr.arpa* должны отображать адреса в единственную зону – хотя все они должны соответствовать каноническим именам узлов.

Затем мы создаем файл *named.conf* для первичного мастер-сервера DNS:

```
options {
                 directory "/var/named";
};

zone "0.0.127.in-addr.arpa" {
                type master;
                file "db.127.0.0";
};

zone "fx.movie.edu" {
                type master;
                file "db.fx.movie.edu";
};
```

```
zone "254.253.192.in-addr.arpa" {
                type master;
                file "db.192.253.254";
};

zone "." {
                type hint;
                file "db.cache";
};
```

Аналогичный файл *named.boot* для BIND 4:

```
directory     /var/named

primary       0.0.127.in-addr.arpa      db.127.0.0  ; loopback
primary       fx.movie.edu              db.fx.movie.edu
primary       254.253.192.in-addr.arpa  db.192.253.254

cache         .                         db.cache
```

Разумеется, если бы мы пользовались программой *h2n*, то могли бы просто выполнить команду:

```
% h2n -d fx.movie.edu -n 192.253.254 -s bladerunner -s outland \
-u hostmaster.fx.movie.edu -m 10:starwars -m 100:wormhole.movie.edu
```

и сэкономить время. В результате мы получили бы – по сути дела – такие же файлы *db.fx.movie.edu*, *db.192.253.254* и *named.boot*.

Теперь следует произвести настройку DNS-клиента узла *bladerunner*. Может оказаться, что нет необходимости создавать файл *resolv.conf*. Если мы установим значение *hostname* для узла *bladerunner* в новое доменное имя этого узла, *bladerunner.fx.movie.edu*, клиент сможет извлечь локальное доменное имя из абсолютного.

Теперь мы запускаем процесс *named* на узле *bladerunner* и проверяем log-файл *syslog* на наличие ошибок. Если *named* стартовал нормально, а в log-файле *syslog* нет ошибок, требующих немедленного исправления, используем *nslookup* для поиска данных для нескольких узлов в зонах *fx.movie.edu* и *254.253.192.in-addr.arpa* :

```
Default Server:  bladerunner.fx.movie.edu
Address:  192.253.254.2

> jedi
Server:  bladerunner.fx.movie.edu
Address:  192.253.254.2

Name:    jedi.fx.movie.edu
Address:  192.253.253.6

> set type=mx
> empire
Server:  bladerunner.fx.movie.edu
Address:  192.253.254.2
```

```
empire.fx.movie.edu     preference = 10,
                        mail exchanger = starwars.fx.movie.edu
empire.fx.movie.edu     preference = 100,
                        mail exchanger = wormhole.movie.edu
fx.movie.edu    nameserver = outland.fx.movie.edu
fx.movie.edu    nameserver = bladerunner.fx.movie.edu
starwars.fx.movie.edu   internet address = 192.253.254.4
wormhole.movie.edu      internet address = 192.249.249.1
wormhole.movie.edu      internet address = 192.253.253.1
bladerunner.fx.movie.edu        internet address = 192.253.254.2
outland.fx.movie.edu    internet address = 192.253.254.3
> ls -d fx.movie.edu
 [bladerunner.fx.movie.edu]
$ORIGIN fx.movie.edu.
@                       1D IN SOA       bladerunner hostmaster (
                                        1               ; порядковый номер
                                        3H              ; обновление
                                        1H              ; повторение
                                        1W              ; устаревание
                                        1H )            ; минимум

                        1D IN NS        bladerunner
                        1D IN NS        outland
                        1D IN MX        10 starwars
                        1D IN MX        100 wormhole.movie.edu.
bladerunner             1D IN A         192.253.254.2
                        1D IN MX        10 starwars
                        1D IN MX        100 wormhole.movie.edu.
br                      1D IN CNAME     bladerunner
empire                  1D IN A         192.253.254.5
                        1D IN MX        10 starwars
                        1D IN MX        100 wormhole.movie.edu.
jedi                    1D IN A         192.253.254.6
                        1D IN MX        10 starwars
                        1D IN MX        100 wormhole.movie.edu.
outland                 1D IN A         192.253.254.3
                        1D IN MX        10 starwars
                        1D IN MX        100 wormhole.movie.edu.
starwars                1D IN A         192.253.254.4
                        1D IN MX        10 starwars
                        1D IN MX        100 wormhole.movie.edu.
@                       1D IN SOA       bladerunner hostmaster (
                                        1               ; порядковый номер
                                        3H              ; обновление
                                        1H              ; повторение
                                        1W              ; устаревание
                                        1H )            ; минимум
> set type=ptr
> 192.253.254.3
Server:  bladerunner.fx.movie.edu
Address:  192.253.254.2
```

```
3.254.253.192.in-addr.arpa      name = outland.fx.movie.edu

> ls -d 254.253.192.in-addr.arpa.
 [bladerunner.fx.movie.edu]
$ORIGIN 254.253.192.in-addr.arpa.
@         1D IN SOA     bladerunner.fx.movie.edu. hostmaster.fx.movie.edu. (
                          1               ; порядковый номер
                          3H              ; обновление
                          1H              ; повторение
                          1W              ; устаревание
                          1H )            ; минимум

           1D IN NS       bladerunner.fx.movie.edu.
           1D IN NS       outland.fx.movie.edu.
1          1D IN PTR      movie-gw.movie.edu.
2          1D IN PTR      bladerunner.fx.movie.edu.
3          1D IN PTR      outland.fx.movie.edu.
4          1D IN PTR      starwars.fx.movie.edu.
5          1D IN PTR      empire.fx.movie.edu.
6          1D IN PTR      jedi.fx.movie.edu.
@         1D IN SOA     bladerunner.fx.movie.edu. hostmaster.fx.movie.edu. (
                          1               ; порядковый номер
                          3H              ; обновление
                          1H              ; повторение
                          1W              ; устаревание
                          1H )            ; минимум
> exit
```

Вывод выглядит нормально, поэтому можно приступать к установке вторичного DNS-сервера для зоны *fx.movie.edu*, а затем переходить к делегированию *fx.movie.edu*.

Вторичный DNS-сервер fx.movie.edu

Вторичный DNS-сервер для зоны *fx.movie.edu* устанавливается без сложностей: следует скопировать файлы *named.conf*, *db.127.0.0* и *db.cache* с узла *bladerunner*, а затем отредактировать *named.conf* и *db.127.0.0* в соответствии с инструкциями, приводимыми в главе 4 «Установка BIND».

Содержимое файла *named.conf*:

```
options {
                directory "/var/named";
};

zone "0.0.127.in-addr.arpa" {
                type master;
                file "db.127.0.0";
};

zone "fx.movie.edu" {
                type slave;
```

```
                masters { 192.253.254.2; };
                file "bak.fx.movie.edu";
};

zone "254.253.192.in-addr.arpa" {
                type slave;
                masters { 192.253.254.2; };
                file "bak.192.253.254";
};

zone "." {
                type hint;
                file "db.cache";
};
```

Эквивалентный файл *named.boot*:

```
directory  /var/named

primary    0.0.127.in-addr.arpa      db.127.0.0
secondary  fx.movie.edu              192.253.254.2  bak.fx.movie.edu
secondary  254.253.192.in-addr.arpa  192.253.254.2  bak.192.253.254

cache      .                         db.cache
```

Как и в случае с узлом *bladerunner*, на узле *outland* не нужен файл *resolv.conf* – если значение *hostname* установлено в *outland.fx.movie.edu*.

И снова – запускаем *named* и сверяемся с log-файлом *syslog* на предмет наличия в нем сообщений об ошибках. Если ошибок нет, можно переходить к поиску записей в *fx.movie.edu*.

Первичный DNS-сервер movie.edu

Осталось только делегировать поддомен *fx.movie.edu* новым DNS-серверам *fx.movie.edu*, работающим на узлах *bladerunner* и *outland*. Добавляем соответствующие NS-записи в файл *db.movie.edu*.

Фрагмент файла *db.movie.edu*:

```
fx    86400    IN    NS    bladerunner.fx.movie.edu.
      86400    IN    NS    outland.fx.movie.edu.
```

Документ RFC 1034 утверждает, что доменные имена в правой части записей в приводимых двух строках (*bladerunner.fx.movie.edu* и *outland.fx.movie.edu*) должны являться каноническими доменными именами DNS-серверов. Удаленный DNS-сервер, использующий для навигации информацию о делегировании, ожидает найти одну или несколько адресных записей, связанных с таким доменным именем, а не запись псевдонима (CNAME). Вообще говоря, этот RFC-документ распространяет данное ограничение на любой тип записей, включающий доменное имя в качестве значения – это значение должно являться каноническим доменным именем.

Но этих двух записей недостаточно. Видите, в чем проблема? Как может DNS-сервер за пределами *fx.movie.edu* производить поиск информации из *fx.movie.edu*? Ведь DNS-сервер *movie.edu* направит внешний сервер к DNS-серверам, авторитативным для зоны *fx.movie.edu*? Разумеется, но NS-записи в файле *db.movie.edu* содержат только *имена* DNS-серверов зоны *fx.movie.edu*. Серверу-иностранцу понадобятся IP-адреса DNS-серверов *fx.movie.edu*, чтобы послать им запросы. Кто может дать ему эти адреса? Только DNS-серверы зоны *fx.movie.edu*. Что было раньше – курица или яйцо?

Вот решение: адреса DNS-серверов *fx.movie.edu* следует включить в файл данных зоны *movie.edu*. Такая информация не принадлежит, строго говоря, зоне *movie.edu*, но она необходима, чтобы работало делегирование для *fx.movie.edu*. Разумеется, если DNS-серверы *fx.movie.edu* находились бы не в пределах *fx.movie.edu*, эта информация – называемая связующими записями (glue records) – не потребовалась бы. Сервер-иностранец нашел бы требуемые адреса, сделав несколько запросов к другим DNS-серверам.

Итак, в комплекте со связующими записями, фрагмент файла *db.movie.edu* выглядит следующим образом:

```
fx      86400    IN    NS    bladerunner.fx.movie.edu.
        86400    IN    NS    outland.fx.movie.edu.
bladerunner.fx.movie.edu.  86400  IN  A  192.253.254.2
outland.fx.movie.edu.      86400  IN  A  192.253.254.3
```

Не следует включать в файл лишние связующие записи. Более старые версии (до 4.9) загружают эти записи в кэш и выдают их в ответах другим DNS-серверам. Если DNS-сервер из адресной записи сменил IP-адрес, а администратор забыл обновить связующие записи, то DNS-сервер будет возвращать устаревшую адресную информацию, что приведет к понижению скорости разрешения для DNS-серверов, интересующихся данными из делегированной зоны, или просто лишит их возможности получать эти данные.

DNS-сервер BIND версии 4.9 или более поздней автоматически игнорирует связующие записи, которые не являются строго необходимыми, и заносит в log-файл *syslog* сообщение о том, что записи были проигнорированы. Так, если бы у нас была NS-запись для *movie.edu*, указывающая на внешний DNS-сервер, *ns-1.isp.net*, и мы сделали бы ошибку, включив адресную запись для этого сервера в файл *db.movie.edu* на первичном мастер-сервере DNS зоны *movie.edu*, то увидели бы примерно такое сообщение в выводе *syslog*:

```
Aug  9 14:23:41 terminator named[19626]: dns_master_load: db.movie.edu:55:
ignoring out-of-zone data
```

Если бы мы работали с DNS-сервером версии более ранней, чем 4.9, и он, по ошибке, включил бы ненужные связующие записи в данные

при передаче данных DNS-серверу более поздней версии, мы могли бы увидеть примерно следующее сообщение в резервной копии файла данных зоны:

```
; Ignoring info about ns-1.isp.net, not in zone movie.edu
; ns-1.isp.net 258983  IN      A       10.1.2.3
```

Обратите внимание, что лишняя адресная запись закомментирована.

И помните, что связующие записи должны быть актуальными. Если *bladerunner* обзаводится новым сетевым интерфейсом – а значит, и новым IP-адресом – следует добавить еще одну адресную запись в связующие данные.

Мы могли бы также создать псевдонимы для любых узлов, осуществляющих переезд из *movie.edu* в *fx.movie.edu*. К примеру, если необходимо переместить *plan9.movie.edu* (сервер, хранящий важную библиотеку свободно доступных алгоритмов специальных эффектов) в зону *fx.movie.edu*, следует создать псевдоним в *movie.edu*, связывающий старое доменное имя с новым:

```
plan9           IN      CNAME   plan9.fx.movie.edu.
```

Это позволит пользователям вне зоны *movie.edu* получать доступ к узлу *plan9*, используя даже прежнее доменное имя *plan9.movie.edu*.

Не следует размещать какую-либо информацию о доменных именах зоны *fx.movie.edu* в файле *db.movie.edu*. Псевдоним *plan9* в действительности принадлежит зоне *movie.edu* (запись связана с именем *plan9.movie.edu*), так что ему место в файле *db.movie.edu*. С другой стороны, псевдоним *p9.fx.movie.edu* для узла *plan9.fx.movie.edu* принадлежит зоне *fx.movie.edu* и должен содержаться в файле *db.fx.movie.edu*. Случись администратору поместить в файл данных зоны «чужую» запись, DNS-сервер BIND 4.9 или более поздней версии проигнорирует ее, как показано в примере с ненужными связующими записями. Более старый DNS-сервер может загрузить запись в кэш и даже поместить ее в свои авторитативные данные, но поскольку такое поведение приводит к непредсказуемым результатам и не реализовано в более новых версиях BIND, лучше всего придерживаться правильного способа, даже если этот способ не навязывается программным пакетом.

Делегирование зоны in-addr.arpa

Мы чуть не забыли делегировать зону 254.253.192.in-addr.arpa! Здесь сложностей чуть больше, чем при делегировании *fx.movie.edu*, поскольку администрирование родительской зоны нам недоступно.

Во-первых, необходимо узнать, в какую родительскую зону входит *254.253.192.in-addr.arpa* и кто занимается сопровождением этой зоны. Получение этой информации может быть связано с сыскным делом, но мы уже рассматривали этот вопрос в главе 3 «С чего начать?».

Выясняется, что родительской зоной для *254.253.192.in-addr.arpa* является *in-addr.arpa*. Если подумать, в этом есть определенный смысл. Администраторы *in-addr.arpa* не видят особого смысла в делегировании зон *253.192.in-addr.arpa* и *192.in-addr.arpa* отдельным инстанциям, поскольку сети вроде 192.253.253/24 и 192.253.254/24 не имеют ничего общего, если 192/8 или 192.253/16 не является одним большим CIDR-блоком. Ими могут заведовать совершенно не связанные между собой организации.

Возможно читатели помнят (из главы 3), что сопровождением зоны *in-addr.arpa* занимается организация ARIN (American Registry of Internet Numbers). Если бы вы этого не помнили, то могли бы воспользоваться программой *nslookup* для поиска контактного адреса из SOA-записи зоны *in-addr.arpa*, как мы демонстрировали в той же самой главе. Нам осталось только воспользоваться веб-интерфейсом «Modify Tool» (Инструмент регистрации изменений) по адресу *http://www.arin.net/cgi-bin/amt.pl*, чтобы запросить регистрацию зоны обратного отображения.

Еще один вторичный DNS-сервер movie.edu

Если лаборатория специальных эффектов станет достаточно большой, возможно будет иметь смысл разместить вторичный DNS-сервер *movie.edu* в сети 192.253.254/24. В этом случае разрешение большей доли DNS-запросов, создаваемых узлами *fx.movie.edu*, будет происходить локально. Кажется логичным сделать один из существующих DNS-серверов зоны *fx.movie.edu* вторичным сервером *movie.edu* – так мы сможем более полно использовать уже существующий сервер, вместо того чтобы создавать еще один.

Мы решили сделать вторичным DNS-сервером *movie.edu* узел *bladerunner*. Это не помешает работе узла *bladerunner* в качестве первичного мастер-сервера DNS зоны *fx.movie.edu*. Единственный DNS-сервер, если снабдить его достаточным количеством памяти, может быть авторитативным буквально для тысяч зон. Один и тот же DNS-сервер может выступать для одних зон в качестве первичного, а для других в качестве вторичного.[1]

Изменения в настройке минимальны: к файлу *named.conf* узла *bladerunner* добавляется один оператор, который сообщает серверу *named*, что следует загружать зону *movie.edu* с IP-адреса первичного мастер-сервера DNS *movie.edu*, который называется *terminator.movie.edu*.

Содержимое файла *named.conf*:

```
options {
```

[1] При этом очевидно, что сервер имен не может быть первичным мастером и вторичным для одной и той же зоны. DNS-сервер либо получает данные для зоны от другого сервера (и тогда он является вторичным), либо из локального файла данных зоны (и тогда он является первичным).

```
    directory "/var/named";
};

zone "0.0.127.in-addr.arpa" {
    type master;
    file "db.127.0.0";
};

zone "fx.movie.edu" {
    type master;
    file "db.fx.movie.edu";
};

zone "254.253.192.in-addr.arpa" {
    type master;
    file "db.192.253.254";
};

zone "movie.edu" {
    type slave;
    masters { 192.249.249.3; };
    file "bak.movie.edu";
};

zone "." {
    type hint;
    file "db.cache";
};
```

Если используется DNS-сервер BIND 4, то содержимое файла *named.boot*:

```
directory     /var/named

primary       0.0.127.in-addr.arpa      db.127.0.0  ; обратная связь
primary       fx.movie.edu              db.fx.movie.edu
primary       254.253.192.in-addr.arpa  db.192.253.254
secondary     movie.edu                 192.249.249.3     bak.movie.edu

cache         .                         db.cache
```

Поддомены доменов in-addr.arpa

Делить на поддомены и делегировать можно не только домены прямого отображения. Если сегмент пространства имен *in-addr.arpa* достаточно крупный, может возникнуть необходимость в его разделении. Обычно домен, соответствующий номеру сети, делится на поддомены, соответствующие подсетям. Механизм работы зависит от типа существующей сети и маски подсети.

Формирование подсети на границе октета

Поскольку в Университете кинематографии всего три сети /24 (класса C), по одной на сегмент, нет особой необходимости в формировании подсетей. Однако наш университет-побратим, Altered State, имеет сеть класса B, 172.20/16. Сеть этого университета разделена на подсети между третьим и четвертым октетом IP-адреса; то есть маска подсети – 255.255.255.0. Этот университет уже создал несколько поддоменов в своем домене *altered.edu*, в частности *fx.altered.edu* (признаемся, мы просто следовали их примеру), *makeup.altered.edu* и *foley.altered.edu*. Поскольку каждый из этих факультетов имеет собственную подсеть (факультет Spesial Effects – подсеть 172.20.2/24, факультет Makeup – 172.20.15/24, а Foley – 172.20.25/24), они хотели бы соответствующим образом разделить и пространство имен *in-addr.arpa*.

Делегирование поддоменов *in-addr.arpa* ничем не отличается от делегирования поддоменов доменов прямого отображения. В файле данных зоны *db.172.20* университету Altered State понадобится создать примерно такие записи:

```
2     86400    IN    NS    gump.fx.altered.edu.
2     86400    IN    NS    toystory.fx.altered.edu.
15    86400    IN    NS    prettywoman.makeup.altered.edu.
15    86400    IN    NS    priscilla.makeup.altered.edu.
25    86400    IN    NS    blowup.foley.altered.edu.
25    86400    IN    NS    muppetmovie.foley.altered.edu.
```

делегируя поддомены, соответствующие отдельным подсетям, DNS-серверам в различных поддоменах.

Несколько важных замечаний: администраторы Altered State могли использовать в поле имени владельца записи только третий октет подсети, поскольку суффикс по умолчанию для этого файла – *20.172.in-addr.arpa*. При этом им пришлось использовать в правой части NS-записей абсолютные доменные имена, чтобы избежать добавления к ним суффикса по умолчанию. И они *не* добавили связующие записи, поскольку имена DNS-серверов, которым делегируется зона, не заканчиваются доменным именем этой зоны.

Формирование подсети не на границе октета

Что делать с сетями, подсети в которых формируются не на границах октетов, как в случае сети /24 (сети класса C)? В таких случаях можно производить делегирование по границам подсетей. Это приводит к одной из двух ситуаций: существует несколько подсетей в каждой зоне *in-addr.arpa* либо существует много зон *in-addr.arpa* для каждой подсети. Оба варианта достаточно неприятные.

Сети классов A и B

Возьмем случай сети /8 (сеть класса A) – 15/8, подсети в которой формируются маской 255.255.248.0 (13-битное поле подсети и 11-битное поле узла, 8192 подсети по 2048 узлов). В таком случае, скажем, сеть 15.1.200.0 охватывает диапазон адресов с 15.1.200.0 по 15.1.207.255. Следовательно, делегирование для одного этого поддомена в *db.15*, файле данных зоны для *15.in-addr.arpa*, может выглядеть следующим образом:

```
200.1.15.in-addr.arpa.    86400    IN    NS    ns-1.cns.hp.com.
200.1.15.in-addr.arpa.    86400    IN    NS    ns-2.cns.hp.com.
201.1.15.in-addr.arpa.    86400    IN    NS    ns-1.cns.hp.com.
201.1.15.in-addr.arpa.    86400    IN    NS    ns-2.cns.hp.com.
202.1.15.in-addr.arpa.    86400    IN    NS    ns-1.cns.hp.com.
202.1.15.in-addr.arpa.    86400    IN    NS    ns-2.cns.hp.com.
203.1.15.in-addr.arpa.    86400    IN    NS    ns-1.cns.hp.com.
203.1.15.in-addr.arpa.    86400    IN    NS    ns-2.cns.hp.com.
204.1.15.in-addr.arpa.    86400    IN    NS    ns-1.cns.hp.com.
204.1.15.in-addr.arpa.    86400    IN    NS    ns-2.cns.hp.com.
205.1.15.in-addr.arpa.    86400    IN    NS    ns-1.cns.hp.com.
205.1.15.in-addr.arpa.    86400    IN    NS    ns-2.cns.hp.com.
206.1.15.in-addr.arpa.    86400    IN    NS    ns-1.cns.hp.com.
206.1.15.in-addr.arpa.    86400    IN    NS    ns-2.cns.hp.com.
207.1.15.in-addr.arpa.    86400    IN    NS    ns-1.cns.hp.com.
207.1.15.in-addr.arpa.    86400    IN    NS    ns-2.cns.hp.com.
```

Для одной подсети – довольно объемная информация о делегировании!

К счастью, начиная с версии 8.2 серверы BIND реализуют поддержку директивы $GENERATE. $GENERATE позволяет создавать группу RR-записей, которые отличаются только численным итератором. К примеру, 16 только что перечисленных NS-записей можно создать следующей парой директив $GENERATE:

```
$GENERATE 200-207 $.1.15.in-addr.arpa.  86400  IN  NS  ns-1.cns.hp.com.
$GENERATE 200-207 $.1.15.in-addr.arpa.  86400  IN  NS  ns-2.cns.hp.com.
```

Синтаксис оператора довольно прост: DNS-сервер создает набор записей из оператора $GENERATE, заменяя символ доллара ($) поочередно числами из диапазона, определенного в первом поле оператора.

Сети класса C

Рассмотрим случай сети /24 (класса C), например 192.253.254/24, в которой формирование сетей производится на основе маски 255.255.255.192. В данном случае существует единственная зона *in-addr.arpa*, *254.253.192.in-addr.arpa*, которая соответствует подсетям 192.253.254.0/26, 192.253.254.64/26, 192.253.254.128/26 и 192.253.254.192/26. Это может быть проблемой, если необходимо разрешить различным организациям сопровождение информации обрат-

ного отображения для этих сетей. Проблема решается одним из трех некрасивых способов.

Решение 1

Первое решение: администрировать зону *254.253.192.in-addr.arpa* в качестве единой сущности, даже не думая о делегировании. Для этого требуется сотрудничество администраторов четырех подсетей либо применение инструмента вроде Webmin (*http://www.webmin.com/webmin*), который позволит каждому из администраторов сопровождать собственный раздел данных.

Решение 2

Второе решение: произвести делегирование на *четвертом* октете. Это даже хуже, чем делегирование для сети /8, которое мы уже продемонстрировали. Нужна хотя бы пара NS-записей на каждый IP-адрес в файле *db.192.253.254*. Это выглядит примерно так:

```
1.254.253.192.in-addr.arpa.    86400    IN    NS    ns1.foo.com.
1.254.253.192.in-addr.arpa.    86400    IN    NS    ns2.foo.com.

2.254.253.192.in-addr.arpa.    86400    IN    NS    ns1.foo.com.
2.254.253.192.in-addr.arpa.    86400    IN    NS    ns2.foo.com.

...

65.254.253.192.in-addr.arpa.    86400    IN    NS    relay.bar.com.
65.254.253.192.in-addr.arpa.    86400    IN    NS    gw.bar.com.

66.254.253.192.in-addr.arpa.    86400    IN    NS    relay.bar.com.
66.254.253.192.in-addr.arpa.    86400    IN    NS    gw.bar.com.

...

129.254.253.192.in-addr.arpa.    86400    IN    NS    mail.baz.com.
129.254.253.192.in-addr.arpa.    86400    IN    NS    www.baz.com.

130.254.253.192.in-addr.arpa.    86400    IN    NS    mail.baz.com.
130.254.253.192.in-addr.arpa.    86400    IN    NS    www.baz.com.
```

и так далее, вплоть до *254.254.253.192.in-addr.arpa*.

Можно существенно сократить объем, воспользовавшись оператором $GENERATE:

```
$GENERATE 0-63 $.254.253.192.in-addr.arpa  86400  IN  NS  ns1.foo.com.
$GENERATE 0-63 $.254.253.192.in-addr.arpa  86400  IN  NS  ns2.foo.com.

$GENERATE 64-127 $.254.253.192.in-addr.arpa.  86400  IN  NS  relay.bar.com.
$GENERATE 64-127 $.254.253.192.in-addr.arpa.  86400  IN  NS  gw.bar.com.

$GENERATE 128-191 $.254.253.192.in-addr.arpa.  86400  IN  NS  mail.baz.com.
$GENERATE 128-191 $.254.253.192.in-addr.arpa.  86400  IN  NS  www.baz.com.
```

Конечно, подразумевается, что в файле *named.conf* на узле *ns1.foo.com* присутствует следующий фрагмент:

```
zone "1.254.253.192.in-addr.arpa" {
                type master;
                file "db.192.253.254.1";
};

zone "2.254.253.192.in-addr.arpa" {
                type master;
                file "db.192.253.254.2";
};
```

Если же на *ns1.foo.com* используется BIND 4, то следующие инструкции присутствуют в файле *named.boot*:

```
primary    1.254.253.192.in-addr.arpa    db.192.253.254.1
primary    2.254.253.192.in-addr.arpa    db.192.253.254.2
```

а в файле *db.192.253.254.1* – одна-единственная PTR-запись:

```
$TTL 1d
@    IN    SOA    ns1.foo.com.    root.ns1.foo.com.    (
                        1        ; Порядковый номер
                        3h       ; Обновление
                        1h       ; Повторение
                        1w       ; Устаревание
                        1h       ; Отрицательное TTL

     IN    NS    ns1.foo.com.
     IN    NS    ns2.foo.com.

     IN    PTR    thereitis.foo.com.
```

Обратите внимание, что эта PTR-запись связана с доменным именем зоны, поскольку доменное имя зоны соответствует всего лишь одному IP-адресу. Теперь, получая запрос PTR-записи для имени *1.254.253.192.in-addr.arpa*, DNS-сервер зоны *254.253.192.in-addr.arpa* направляет клиент к серверам *ns1.foo.com* и *ns2.foo.com*, которые возвращают именно эту, единственную в зоне, PTR-запись.

Решение 3

И наконец, существует более умный способ, избавляющий нас от необходимости сопровождать отдельный файл данных зоны для каждого IP-адреса.[1] Организация, отвечающая за всю сеть /24, создает CNAME-записи для каждого из доменных имен зоны; эти CNAME-записи указывают на доменные имена в новых поддоменах, которые, в свою оче-

[1] Впервые мы увидели эту идею в конференции *comp.protocols.tcp-ip.domains* в изложении Глена Херрмансфельдта (Glen Herrmansfeldt) из КалТех. В настоящее время способ стандартизирован в документе RFC 2317.

редь, делегируются различным DNS-серверам. Новые поддомены могут иметь практически любые имена, например *0-63*, *64-127*, *128-191* и *192-255*, четко показывающие диапазоны адресов, для которых производится обратное отображение в каждом из доменов. При этом каждый поддомен содержит только PTR-записи для указанного диапазона.

Фрагмент файла *db.192.253.254*:

```
1.254.253.192.in-addr.arpa.  IN  CNAME  1.0-63.254.253.192.in-addr.arpa.
2.254.253.192.in-addr.arpa.  IN  CNAME  2.0-63.254.253.192.in-addr.arpa.

...

0-63.254.253.192.in-addr.arpa.    86400    IN    NS    ns1.foo.com.
0-63.254.253.192.in-addr.arpa.    86400    IN    NS    ns2.foo.com.

65.254.253.192.in-addr.arpa. IN  CNAME 65.64-127.254.253.192.in-addr.arpa.
66.254.253.192.in-addr.arpa. IN  CNAME 66.64-127.254.253.192.in-addr.arpa.

...

64-127.254.253.192.in-addr.arpa.    86400    IN    NS    relay.bar.com.
64-127.254.253.192.in-addr.arpa.    86400    IN    NS    gw.bar.com.

129.254.253.192.in-addr.arpa.  IN  CNAME  129.128-191.254.253.192.in-addr.
arpa.
130.254.253.192.in-addr.arpa.  IN  CNAME  130.128-191.254.253.192.in-addr.
arpa.

...

128-191.254.253.192.in-addr.arpa.    86400    IN    NS    mail.baz.com.
128-191.254.253.192.in-addr.arpa.    86400    IN    NS    www.baz.com.
```

И опять можно использовать $GENERATE для сокращения:

```
$GENERATE 1-63 $ IN CNAME $.0-63.254.253.192.in-addr.arpa.

0-63.254.253.192.in-addr.arpa.    86400    IN    NS    ns1.foo.com.
0-63.254.253.192.in-addr.arpa.    86400    IN    NS    ns2.foo.com.

$GENERATE 65-127 $ IN CNAME $.64-127.254.253.192.in-addr.arpa.

64-127.254.253.192.in-addr.arpa.    86400    IN    NS    relay.bar.com.
64-127.254.253.192.in-addr.arpa.    86400    IN    NS    gw.bar.com.
```

Файл данных для зоны *0-63.254.253.192.in-addr.arpa* (*db.192.253.254.0-63*) вполне может содержать только PTR-записи для IP-адресов с 192.253.254.1 по 192.253.254.63.

Фрагмент файла *db.192.253.254.0-63*:

```
$TTL 1d
@    IN    SOA    ns1.foo.com.    root.ns1.foo.com.    (
                         1       ; Порядковый номер
                         3h      ; Обновление
                         1h      ; Повторение
```

```
                                1w      ; Устаревание
                                1h )    ; Отрицательное TTL

        IN    NS    ns1.foo.com.
        IN    NS    ns2.foo.com.

1       IN    PTR   thereitis.foo.com.
2       IN    PTR   setter.foo.com.
3       IN    PTR   mouse.foo.com.
...
```

Работа этого варианта не очень прозрачна, поэтому рассмотрим процесс несколько подробнее. Клиент запрашивает у локального DNS-сервера PTR-запись для *1.254.253.192.in-addr.arpa*. Локальный DNS-сервер в итоге добирается до DNS-сервера *254.253.192.in-addr.arpa*, который возвращает CNAME-запись, сообщающую, что *1.254.253.192.in-addr.arpa* в действительности является всего лишь псевдонимом для *1.0-63.254.253.192.in-addr.arpa* и что PTR-запись связана с последним именем. Ответ также содержит NS-записи, которые говорят локальному DNS-серверу, что авторитативными серверами для *0-63.254.253.192.in-addr.arpa* являются *ns1.foo.com* и *ns2.foo.com*. Локальный DNS-сервер посылает запрос PTR-записи для *1.0-63.254.253.192.in-addr.arpa* – DNS-серверу *ns1.foo.com* или *ns2.foo.com*, и получает искомое.

Заботливые родители

Теперь, разобравшись с делегированием DNS-серверам *fx.movie.edu*, мы – как заботливые родители – должны проверить делегирование с помощью программы *host*. Как? Мы еще не поделились с читателями программой *host*? Версия *host* для Unix-систем доступна для анонимного FTP-копирования с сервера *ftp.nikhef.nl* (имя файла – */pub/network/host.tar.Z*).

Чтобы собрать программу *host*, следует сначала распаковать архив:

```
% zcat host.tar.Z | tar -xvf -
```

А затем выполнить команду:

```
% make
```

host облегчает проверку делегирования. С помощью этого инструмента можно производить поиск NS-записей для зоны, используя DNS-серверы зоны-родителя. Если с этими записями все в порядке, можно использовать *host* для посылки запросов каждому из DNS-серверов, перечисленных для SOA-записи зоны. Запросы нерекурсивны, поэтому DNS-сервер, к которому произошло обращение, не контактирует с другими DNS-серверами в поисках SOA-записи. Когда DNS-сервер возвращает ответ, *host* проверяет ответ на предмет наличия установленного

бита *aa* – authoritative answer (авторитативный ответ). В случае положительного результата DNS-сервер проверяет ответное сообщение на присутствие собственно ответа. При выполнении этой пары условий DNS-сервер помечается как авторитативный для зоны. В противном случае сервер не является авторитативным, и *host* сообщает об ошибке.

Почему столько суматохи вокруг некорректного делегирования? Дело в том, что оно может замедлять разрешение имен или приводить к распространению устаревшей информации о корневых DNS-серверах. Когда сервер получает запрос данных из зоны, для которой он не является авторитативным, то прилагает все усилия, чтобы сообщить клиенту полезную информацию по теме. Эта «полезная информация» содержится в NS-записях ближайшей зоны-предка, которая известна DNS-серверу. (Мы вкратце поговорили об этом в главе 8 «Развитие домена», когда обсуждали, почему не следует регистрировать DNS-сервер, специализирующийся на кэшировании.)

Допустим, один из DNS-серверов *fx.movie.edu* по ошибке получает итеративный запрос адреса *carrie.horror.movie.edu*. Серверу ничего не известно о зоне *horror.movie.edu* (хотя возможно существуют какие-то данные в кэше), но, вполне вероятно, он кэшировал NS-записи для *movie.edu*, поскольку эти записи содержат информацию о родительской зоне. И эти записи DNS-сервер возвращает автору запроса.

В описанном сценарии возвращаемые NS-записи могут помочь DNS-серверу, от которого исходил запрос, найти ответ. Однако в сети Интернет правда жизни заключается в том, что не все администраторы следят за тем, чтобы файлы корневых указателей соответствовали реальности. Если один из наших DNS-серверов использует для навигации некорректную информацию о делегировании и запросит у удаленного DNS-сервера данные, которых у того нет, может произойти следующее:

```
% nslookup
Default Server:  terminator.movie.edu
Address:  192.249.249.3
> set type=ns
> .
Server:  terminator.movie.edu
Address:  192.249.249.3

Non-authoritative answer:
 (root)  nameserver = D.ROOT-SERVERS.NET
 (root)  nameserver = E.ROOT-SERVERS.NET
 (root)  nameserver = I.ROOT-SERVERS.NET
 (root)  nameserver = F.ROOT-SERVERS.NET
 (root)  nameserver = G.ROOT-SERVERS.NET
 (root)  nameserver = A.ROOT-SERVERS.NET
 (root)  nameserver = H.ROOT-SERVERS.NET
 (root)  nameserver = B.ROOT-SERVERS.NET
 (root)  nameserver = C.ROOT-SERVERS.NET
 (root)  nameserver = A.ISI.EDU          – Эти три сервера
```

```
(root)  nameserver = SRI-NIC.ARPA        — уже не являются
(root)  nameserver = GUNTER-ADAM.ARPA    — корневыми
```

Удаленный сервер попытался «помочь» нашему DNS-серверу, послав ему текущий перечень корневых серверов. К сожалению, удаленный DNS-сервер не шел в ногу со временем и поэтому вернул неправильные NS-записи. Наш локальный DNS-сервер, не имея лучшей альтернативы, кэшировал эти данные.

DNS-серверы BIND версии 4.9 и более поздних не поддаются на подобные провокации.

Запросы к некорректно настроеным DNS-серверам *in-addr.arpa* зачастую приводят к получению некорректных NS-записей для корневых DNS-серверов, поскольку зоны *in-addr.arpa* и *arpa* являются ближайшими предками большинства поддоменов *in-addr.arpa*, а DNS-серверы очень редко кэшируют NS-записи *in-addr.arpa* и *arpa*. (Корневые DNS-серверы редко возвращают эти записи, так как делегируют напрямую поддоменам более низких уровней.) Если DNS-сервер кэшировал некорректные NS-записи для корневых DNS-серверов, это может пагубно повлиять на разрешение имен.

Эти NS-записи могут привести к тому, что наш DNS-сервер будет посылать запрос корневому DNS-серверу, у которого изменился IP-адрес, или корневому DNS-серверу, который более не существует. Если у вас действительно тяжелый день, некорректные NS-записи корневых серверов могут указывать на существующий, не-корневой DNS-сервер, расположенный близко к вашей сети. Несмотря на то, что этот сервер не способен возвращать авторитативные ответы, ваш сервер будет оказывать ему предпочтение, исходя из высокой скорости реакции.

Используем host

Если наша лекция убедила читателей в важности соблюдения корректности делегирования, они, вероятно, не будут против узнать, как можно использовать *host*, чтобы не попасть в ряды злодеев.

Первый шаг: используем *host* для поиска NS-записей нашей зоны с помощью DNS-сервера родительской зоны и убедимся, что все в порядке. Следующая команда производит проверку для NS-записей *fx.movie.edu* с помощью одного из DNS-серверов зоны *movie.edu*:

```
% host -t ns fx.movie.edu. terminator.movie.edu.
```

Если все в порядке, NS-записи отображаются в выводе программы:

```
fx.movie.edu   NS   bladerunner.fx.movie.edu
fx.movie.edu   NS   outland.fx.movie.edu
```

Мы видим, что NS-записи, делегирующие зону *fx.movie.edu*, верны.

Теперь мы используем *host* для «SOA-теста» и запросим у каждого из DNS-серверов *fx.movie.edu* SOA-запись. Параллельно мы увидим, возвращается ли авторитативный ответ:

```
% host -C fx.movie.edu.
```

Обычно это приводит к отображению уже показанных NS-записей наряду с содержимым SOA-записи зоны *fx.movie.edu*:

```
fx.movie.edu      NS      bladerunner.fx.movie.edu
bladerunner.fx.movie.edu   hostmaster.fx.movie.edu  (1 10800 3600 608400 3600)
fx.movie.edu      NS      outland.fx.movie.edu
bladerunner.fx.movie.edu   hostmaster.fx.movie.edu  (1 10800 3600 608400 3600)
```

Если один из DNS-серверов *fx.movie.edu* – скажем, *outland* – был настроен неправильно, мы можем увидеть следующее:

```
fx.movie.edu   NS   bladerunner.fx.movie.edu
fx.movie.edu   NS   outland.fx.movie.edu
fx.movie.edu SOA record currently not present at outland.fx.movie.edu
fx.movie.edu has lame delegation to outland.fx.movie.edu
```

Смысл сообщения заключается в том, что DNS-сервер на узле *outland* работает, но не является авторитативным для зоны *fx.movie.edu*.

Если бы один из DNS-серверов *fx.movie.edu* не работал вовсе, мы увидели бы такое сообщение:

```
fx.movie.edu   NS   bladerunner.fx.movie.edu
bladerunner.fx.movie.edu   hostmaster.fx.movie.edu  (1 10800 3600 608400 3600)
fx.movie.edu   NS   outland.fx.movie.edu
fx.movie.edu SOA record not found at outland.fx.movie.edu, try again
```

В этом случае сообщение *try again* (повторите попытку) говорит о том, что программа *host* отправила серверу *outland* запрос, но не получила ответа за приемлемое время.

Разумеется, мы могли бы произвести проверку делегирования *fx.movie.edu* с помощью *nslookup*, но удобные ключи командной строки *host* позволяют решить эту задачу с особенной легкостью.

Управление делегированием

Если лаборатория специальных эффектов укрупнится, может оказаться, что необходимо увеличить число DNS-серверов. Мы уже описывали установку новых DNS-серверов в главе 8 и даже упомянули, какую информацию следует посылать администратору родительской зоны. Но мы не сказали, какие обязанности возлагаются на родителя.

Оказывается, что работа родителя в этом случае не очень сложна, особенно, если администраторы поддоменов присылают полную инфор-

мацию. Предположим, что произошло расширение лаборатории специальных эффектов в новую сеть, 192.254.20/24. В этой сети проживает стадо новых графических рабочих станций повышенной производительности. Одна из них, *alien.fx.movie.edu*, будет выступать в качестве нового DNS-сервера этой сети.

Администраторы зоны *fx.movie.edu* (которая была делегирована ребятам из лаборатории) посылают администраторам родительской зоны (то есть нам) короткое уведомление:

```
Привет!

Мы только что настроили alien.fx.movie.edu (192.254.20.3) в качестве
DNS-сервера для fx.movie.edu. Пожалуйста, обновите информацию
о делегировании. NS-записи, которые необходимо добавить, прилагаются.

Thanks,

Arty Segue
ajs@fx.movie.edu

----- cut here -----

fx.movie.edu.  86400  IN  NS  bladerunner.fx.movie.edu.
fx.movie.edu.  86400  IN  NS  outland.fx.movie.edu.
fx.movie.edu.  86400  IN  NS  alien.fx.movie.edu.

bladerunner.fx.movie.edu.  86400  IN  A  192.253.254.2
outland.fx.movie.edu.      86400  IN  A  192.253.254.3
alien.fx.movie.edu.        86400  IN  A  192.254.20.3
```

Наша задача – задача администраторов *movie.edu* довольно проста: необходимо добавить NS- и A-записи в файл *db.movie.edu*.

Что делать в случае, если мы используем программу *h2n* для создания данных DNS-сервера? Мы можем поместить информацию о делегировании в файл *spcl.movie*, который *h2n* включает с помощью директивы $INCLUDE в конец создаваемого файла *db.movie*.

Последнее действие администратора зоны *fx.movie.edu* – послать аналогичное уведомление по адресу *noc@netsol.com* (администратору зоны *in-addr.arpa*), запросив делегирование поддомена *20.254.192.in-addr.arpa* DNS-серверам *alien.fx.movie.edu*, *bladerunner.fx.movie.edu* и *outland.fx.movie.edu*.

Заглушки: еще один способ управления делегированием

Если вы работаете с DNS-сервером BIND версии 4.9 или более поздней, существует возможность избежать ручного сопровождения информации о делегировании. В DNS-серверах BIND начиная с версии 4.9 присутствует реализация экспериментального механизма *зон-заглушек*, который и позволяет DNS-серверу самостоятельно отслеживать изменения в информации, связанной с делегированием.

DNS-сервер, реализующий функциональность заглушки для зоны, выполняет дискретные запросы SOA- и NS-записей зоны, а также необходимых связующих записей. DNS-сервер использует полученные NS-записи для делегирования зоны, а SOA-записи определяют частоту выполнения подобных запросов. В этом случае, когда администраторы поддомена вносят изменения в данные DNS-серверов поддомена, они просто обновляют NS-записи, а авторитативные DNS-серверы родительской зоны запрашивают обновленные записи в пределах интервала обновления.

На DNS-серверах *movie.edu* мы добавили бы следующий оператор в файл *named.conf*:

```
zone "fx.movie.edu" {
                type stub;
                masters { 192.253.254.2; };
                file "stub.fx.movie.edu";
};
```

В случае DNS-сервера BIND 4.9 использовали бы такую инструкцию:

```
stub     fx.movie.edu    192.253.254.2     stub.fx.movie.edu
```

Обратите внимание, что следует настроить подобным образом все DNS-серверы *movie.edu*, поскольку при изменении информации о делегировании *fx.movie.edu* не изменяется порядковый номер зоны *movie.edu*.[1] Если все DNS-серверы *movie.edu* работают с зоной-заглушкой поддомена, они синхронизируются.

Как справиться с переходом к поддоменам

Мы не станем обманывать читателей – пример с поддоменом *fx.movie.edu* был не очень жизненным. Основная причина – волшебное появление узлов лаборатории специальных эффектов. В реальной жизни лаборатория началась бы с нескольких узлов, которые входили бы в зону *movie.edu*. После получения щедрого пожертвования, гранта NSF или корпоративного подарка лаборатория может немного подрасти и купить еще несколько машин. Рано или поздно в лаборатории будет достаточно узлов, чтобы гарантировать создание нового поддомена. Однако к тому моменту многие из узлов обретут известность под своими именами в домене *movie.edu*.

Мы коротко упоминали использование CNAME-записей в родительской зоне (в примере с узлом *plan9.movie.edu*), которые позволили бы пользователям безболезненно привыкнуть к переезду узла в другой домен. Но представьте себе, что целая сеть или подсеть переезжает в другой домен!

[1] Серверы имен BIND 9 обычно не передают NS-записи в родительскую зону, чтобы они не включались в данные при передаче зоны.

Стратегия, которую мы рекомендуем, связана с использованием CNAME-записей в аналогичном стиле, но в гораздо больших масштабах. Используя инструмент вроде *h2n*, можно создавать CNAME-записи для большого числа узлов сразу. Это позволяет пользователям продолжать пользоваться прежними доменными именами любых из переехавших узлов. Однако, когда они попытаются соединиться с любым из этих узлов по протоколу telnet или FTP (или еще какому-то), то получат сообщение, что подключились к узлу *fx.movie.edu*:

```
% telnet plan9
Trying...
Connected to plan9.fx.movie.edu.
Escape character is '^]'.

HP-UX plan9.fx.movie.edu A.09.05 C 9000/735 (ttyu1)

login:
```

Конечно, многие пользователи не замечают столь тонкой разницы, поэтому придется заняться рекламной деятельностью и уведомить народ о новостях.

На узлах *fx.movie.edu*, работающих со старыми версиями программы *sendmail*, необходимо настроить *sendmail* на прием почтовых сообщений для новых доменных имен. Современные версии *sendmail* производят канонизацию имен узлов из заголовков сообщений с помощью DNS-сервера, прежде чем посылать сообщения. Канонизация превратит псевдоним из *movie.edu* в каноническое имя из *fx.movie.edu*. Однако если на принимающем узле работает старая версия *sendmail*, в которой жестко закодировано доменное имя локального узла, придется изменить это имя на новое вручную. Это обычно требует внесения простых изменений в класс *w* или файловый класс *w* в файле *sendmail.cf*; более подробная информация содержится в разделе «MX-алгоритм» главы 5 «DNS и электронная почта».

Как создать все эти псевдонимы? Достаточно просто сказать *h2n*, что следует создать псевдонимы для узлов в сетях *fx.movie.edu* (192.253.254/24 и 192.254.20/24) и задать (в файле */etc/hosts*) новые доменные имена для этих узлов. К примеру, используя таблицу узлов *fx.movie.edu*, мы можем с легкостью создать псевдонимы в *movie.edu* для всех узлов *fx.movie.edu*.

Фрагмент файла */etc/hosts*:

```
192.253.254.1 movie-gw.movie.edu movie-gw
# fx: первичный
192.253.254.2 bladerunner.fx.movie.edu bladerunner br
# fx: вторичный
192.253.254.3 outland.fx.movie.edu outland
192.253.254.4 starwars.fx.movie.edu starwars
192.253.254.5 empire.fx.movie.edu empire
192.253.254.6 jedi.fx.movie.edu jedi
192.254.20.3  alien.fx.movie.edu alien
```

Ключ *-c* программы *h2n* в качестве аргумента принимает доменное имя зоны. Когда *h2n* обнаруживает узел из этой зоны в сети, для которой производится генерация данных, то создает псевдонимы в текущей зоне (которая указывается с помощью ключа *-d*). Поэтому, выполнив команду:

```
% h2n -d movie.edu -n 192.253.254 -n 192.254.20 \
-c fx.movie.edu -f options
```

(где файл *options* содержит прочие ключи командной строки для создания данных, относящихся к прочим сетям *movie.edu*), мы можем создать в *movie.edu* псевдонимы для всех узлов *fx.movie.edu*.

Удаление псевдонимов из родительской зоны

Псевдонимы в родительской зоне полезны в плане минимизации отрицательных последствий при перемещении узлов, но они, по существу, являются костылями. Как и любые костыли, они ограничивают свободу движений. Они замусоривают пространство имен родительской зоны, и это при том, что одна из причин создания поддоменов – разгрузка и уменьшение зоны. Помимо этого, псевдонимы исключают использование в родительской зоне узлов с именами, совпадающими с именами узлов поддомена.

После тактичной паузы, о наличии которой следует обязательно проинформировать пользователей, все псевдонимы должны быть удалены, за исключением псевдонимов для широко известных в сети Интернет узлов. Во время этой паузы пользователи могут привыкнуть к новым доменными именам, отредактировать сценарии, файлы *.rhosts* и т. д. Пусть вас ничто не введет в заблуждение – сохранение жизни псевдонимам в родительской зоне противоречит самой идее DNS, поскольку псевдонимы мешают администраторам родительской зоны и администраторам поддоменов производить автономное именование узлов.

Возможно, придется оставить CNAME-записи для широко известных узлов Интернет или центральных ресурсов сети нетронутыми, из-за возможных последствий утраты связи. С другой стороны, прежде чем производить перевод широко известного узла или важного ресурса в поддомен, стоит как следует подумать – быть может, этот узел следует оставить в родительской зоне.

h2n предоставляет простой способ удалить псевдонимы, столь же просто созданные с помощью ключа *-c*, даже если записи для узлов поддомена подмешаны в таблицу узлов или в ту же сеть, что узлы других зон. Ключ *-e* принимает доменное имя зоны в качестве аргумента и предписывает *h2n* исключить (*e* от слова *exclude*) все записи, содержащие указанное доменное имя для всех сетей, для которых будет происходить создание данных. К примеру, следующая команда удалит все ранее созданные записи CNAME для узлов *fx.movie.edu*, но при этом

все равно создаст адресную запись для узла *movie-gw.movie.edu* (принадлежащего сети 192.253.254/24):

```
% h2n -d movie.edu -n 192.253.254 -n 192.254.20 \
-e fx.movie.edu -f options
```

Жизнь родителя

Всю эту информацию о жизни родителя нелегко переварить за один прием, поэтому мы резюмируем основные пункты нашей дискуссии. Жизненный цикл типичного родителя выглядит примерно так:

1. Единственная зона, которая содержит все узлы.
2. Зона разбивается на поддомены, некоторые из которых входят в ту же зону, что и родитель – при необходимости. Для широко известных узлов, совершивших переезд, в родительской зоне могут быть созданы CNAME-записи.
3. После тактичной паузы все существующие CNAME-записи удаляются.
4. Администратор обновляет информацию о делегировании поддоменов вручную либо при помощи зон-заглушек и периодически проверяет работоспособность делегирования.

Теперь, когда мы рассказали все о родителях и детях, можно переходить к разговору о более серьезных возможностях DNS-серверов. Некоторые из них могут пригодиться при воспитании детей.

По договору между издательством «Символ-Плюс» и Интернет-магазином «Books.Ru – Книги России» единственный легальный способ получения данного файла с книгой ISBN 5-93286-035-9, название «DNS и BIND, 4-е издание» – покупка в Интернет-магазине «Books.Ru – Книги России». Если Вы получили данный файл каким-либо другим образом, Вы нарушили международное законодательство и законодательство Российской Федерации об охране авторского права. Вам необходимо удалить данный файл, а также сообщить издательству «Символ-Плюс» (piracy@symbol.ru), где именно Вы получили данный файл.

10

Дополнительные возможности

- *Списки отбора адресов и управления доступом*
- *DNS: динамические обновления*
- *DNS NOTIFY (уведомления об изменениях зоны)*
- *Инкрементальная передача зоны (IXFR)*
- *Ретрансляция*
- *Виды*
- *Round Robin: распределение нагрузки*
- *Сортировка адресов DNS-сервером*
- *DNS-серверы: предпочтения*
- *Нерекурсивный DNS-сервер*
- *Борьба с фальшивыми DNS-серверами*
- *Настройка системы*
- *Совместимость*
- *Основы адресации в IPv6*

–...Но я могу вам сказать, как их зовут.
– А они, конечно, идут, когда их зовут? – небрежно заметил Комар.
– Нет, кажется, не идут.
– Тогда зачем же их звать, если они не идут?

В последних версиях DNS-серверов BIND 8.2.3 и 9.1.0 *огромное* число новых возможностей. Из наиболее выдающихся нововведений можно отметить динамические обновления, асинхронные уведомления об изменениях зон («NOTIFY») и инкрементальную передачу зональных данных. Из прочих новшеств самые важные связаны с безопасностью: они позволяют объяснить DNS-серверу, на чьи запросы следует отвечать, кому разрешать получение зоны и от кого допускать динамические обновления. Многие из механизмов обеспечения безопасности не имеют применения в корпоративных сетях, а иные будут полезны администратору любого DNS-сервера.

В данной главе мы рассмотрим все эти возможности и их потенциальные применения в вашей инфраструктуре DNS. (За исключением под-

робного материала по брандмауэрам, который мы приберегли для следующей главы.)

Списки отбора адресов и управления доступом

Но прежде чем перейти к новым возможностям, следует рассказать о списках отбора адресов (address match list). Практически каждый механизм обеспечения безопасности в BIND 8 и 9 (а также некоторые механизмы, вовсе не связанные с защитой) работает с применением списков отбора адресов.

Список отбора адресов – это список (а что же еще?) элементов, определяющий набор из одного или более IP-адресов. Элементы списка могут являться отдельными IP-адресами, IP-префиксами, либо именованными списками отбора адресов (подробнее чуть позже).[1] IP-префикс имеет следующий формат:

```
сеть в формате октетов через точку/число битов в маске сети
```

К примеру, сеть 15.0.0.0 с маской сети 255.0.0.0 (восемь единиц подряд) записывается как 15/8. В традиционной терминологии это сеть 15 класса A. Сеть, состоящая из IP-адресов с 192.168.1.192 по 192.168.1.255 может быть представлена в виде 192.168.1.192/26 (сеть 192.168.1.192 с маской сети 255.255.255.192, в которой 26 единиц подряд). Вот список отбора адресов, состоящий из двух сетей:

```
15/8; 192.168.1.192/26;
```

Именованный список отбора адресов – это точно такой же список, которому присвоено имя. Чтобы использоваться в другом списке отбора адресов, именованный список должен быть предварительно определен в файле *named.conf* с помощью оператора *acl* (*от access control list*). Оператор *acl* имеет примитивный синтаксис:

```
acl name { address_match_list; };
```

Он делает определенное имя (*name*) эквивалентом указанного списка отбора адресов. Хотя имя оператора, *acl*, наводит на мысли о списке управления доступом («access control list»), именованные списки отбора адресов можно использовать в любом месте, где может присутствовать такой список, включая и случаи, которые никоим образом не относятся к разграничению доступа.

Если одни и те же элементы списка применяются во многих местах, разумно будет связать их с определенным именем с помощью операто-

[1] В случае использования BIND 9 список отбора адресов может также включать IPv6-адреса и IPv6-префиксы, описанные позже в данной главе.

ра *acl*. Затем по имени можно ссылаться на созданный список. К примеру, назовем сеть 15/8 ее действительным именем: «HP-NET». А сети 192.168.1.192/26 дадим имя «internal»:

```
acl "HP-NET" { 15/8; };

acl "internal" { 192.168.1.192/26; };
```

Теперь мы можем ссылаться на эти списки отбора адресов из других списков отбора адресов по именам. Это не только сокращает набор, но и делает конечный файл *named.conf* более читаемым.

Мы предусмотрительно поместили имена ACL-списков в кавычки, чтобы избежать конфликтов с именами, зарезервированными BIND для собственных нужд. Если вы уверены, что имена списков не конфликтуют с зарезервированными словами, можно обойтись и без кавычек.

Существует четыре предопределенных именованных списка отбора адресов:

none

Пустой список. В него не входит ни один IP-адрес

any

Любые IP-адреса

localhost

Любые IP-адреса локального узла (узла, на котором работает DNS-сервер)

localnets

Любая из сетей, в которой есть сетевой интерфейс у локального узла (вычисление сетей происходит путем удаления битов узла из адресов интерфейсов с помощью соответствующих масок).

DNS: динамические обновления

Мир сети Интернет и сетей TCP/IP в целом стал гораздо более динамичным за последнее время. Большинство крупных корпораций используют DHCP для динамического распределения IP-адресов. Практически все интернет-провайдеры используют динамическое распределение адресов DHCP для клиентов, использующих коммутируемые соединения и кабельные модемы. Чтобы не отстать, система DNS должна реализовывать поддержку динамического добавления и удаления записей. Этот механизм, получивший название DNS Dynamic Update (динамические обновления DNS), описан в документе RFC 2136.

BIND 8 и 9 реализуют поддержку механизма динамических обновлений, описанных в RFC 2136. Это позволяет авторизованным клиентам производить добавление и удаление RR-записей зоны, для которой

DNS-сервер является авторитативным. Автор обновления может определить авторитативные DNS-серверы зон путем поиска NS-записей. Если DNS-сервер, получивший сообщение обновления от авторизованного клиента, не является первичным DNS-мастер-сервером зоны, он передает обновление «вверх по течению» – своему мастеру. Этот процесс называется «ретрансляцией обновлений». Если следующий DNS-сервер оказывается, в свою очередь, вторичным для зоны, он также передает обновление вверх по течению. В конце концов, только первичный DNS-мастер-сервер зоны владеет изменяемой копией данных; все вторичные DNS-серверы получают свои копии данных зоны от первичного мастера, прямо или косвенно (через другие вторичные серверы). Когда первичный DNS-сервер обработал динамическое обновление и внес изменение в зону, вторичные серверы могут получить новую копию путем синхронизации.

Динамические обновления предусматривают более сложные действия, чем удаление и добавление записей. Обновление может затрагивать добавление или удаление отдельных RR-записей, удаление RRset-наборов (наборов RR-записей, имеющих общего владельца, одинаковый класс и тип, например, все адресные записи для *www.movie.edu*) и даже удаление всех записей, связанных с определенным доменным именем. Выполнение обновления также может требовать выполнения специальных условий, например, существования или отсутствия определенных записей в данных зоны. Так, обновление может добавить адресную запись:

```
armageddon.fx.movie.edu.  300  IN  A  192.253.253.15
```

только в случае, если доменное имя *armageddon.fx.movie.edu* в этот момент не используется, либо если для *armageddon.fx.movie.edu* не существует адресных записей.

Замечание по ретрансляции обновлений: реализация этого механизма в DNS-серверах BIND отсутствовала до версии 9.1.0, поэтому при использовании DNS-серверов более ранних версий следует проверять, что обновление посылается напрямую первичному DNS-мастер-серверу зоны, для которой оно предназначено. Вопрос можно снять, указав первичный DNS-мастер-сервер для зоны в поле MNAME записи SOA. Большинство функций, связанных с динамическими обновлениями, используют поле MNAME в качестве источника информации о том, каким авторитативным DNS-серверам следует посылать обновления.

По большей части, функциональность динамических обновлений находит применение в программах вроде серверов DHCP, которые автоматически присваивают адреса компьютерам, что связано с необходимостью регистрации соответствующих отображений имен в адреса и адресов в имена. Некоторые из этих программ используют новую

функцию DNS-клиента, *ns_update( )*, для создания сообщений обновления и отправки их авторитативному серверу для зоны, владеющей доменным именем.

Помимо этого, обновления могут создаваться вручную, с помощью программы *nsupdate*, которая входит в стандартный дистрибутив BIND. *nsupdate* принимает однострочные команды и преобразует их в сообщения обновлений. Команды могут поступать со стандартного ввода (по умолчанию), либо из файла, имя которого должно быть указано в качестве аргумента *nsupdate*. Команды, которые не разделены пустыми строками, записываются в одно сообщение, пока не кончится место.

nsupdate понимает следующие команды:

prereq yxrrset domain name type [rdata]

Создание предварительного условия для выполнения команд обновления. Должен существовать RRset-набор типа *type*, связанный с доменным именем (*domain name*). Если указано поле *rdata*, эти данные также должны существовать.

prereq nxrrset

Создание предварительного условия для выполнения команд обновления. Должен отсутствовать Rrset-набор типа *type* для доменного имени *domain name*.

prereq yxdomain domain name

Должно существовать указанное доменное имя.

prereq nxdomain

Указанное доменное имя должно быть несуществующим.

update delete domain name [type] [rdata]

Удаление указанного доменного имени либо, при указании поля *type*, удаление указанного RRset-набора, а в случае указания поля *rdata* – удаление записей, соответствующих параметрам *domain name*, *type* и *rdata*.

update add domain name ttl [class] type rdata

Добавление к зоне указанной записи. Обратите внимание, что значение TTL должно быть указано, как значения *type* и *rdata*, а поле класса является необязательным – по умолчанию принимается класс IN.

К примеру, следующая команда:

```
% nsupdate
> prereq nxdomain mib.fx.movie.edu.
> update add mib.fx.movie.edu. 300 A 192.253.253.16
>
```

предписывает серверу добавить адресную запись для *mib.fx.movie.edu*, но только в том случае, если доменное имя еще не существует. Обрати-

те внимания на последнюю пустую строку – она говорит программе *nsupdate*, что обновление можно посылать. Хитро, а?

Команда:

```
% nsupdate
> prereq yxrrset mib.fx.movie.edu. MX
> update delete mib.fx.movie.edu. MX
> update add mib.fx.movie.edu. 600 MX 10 mib.fx.movie.edu.
> update add mib.fx.movie.edu. 600 MX 50 postmanrings2x.movie.edu.
>
```

проверяет, существуют ли MX-записи для *mib.fx.movie.edu*, удаляет их, если существуют, и заменяет двумя новыми.

Возможности динамических обновлений ограничены: невозможно удалить всю зону (хотя можно удалить все ее данные, за исключением SOA-записи и одной NS-записи), и невозможно создать новую зону.

Динамические обновления и порядковые номера

Когда DNS-сервер обрабатывает поступившее динамическое обновление, данные зоны изменяются, поэтому порядковый номер для зоны должен быть увеличен, чтобы изменения были восприняты вторичными DNS-серверами. Это происходит автоматически. Однако DNS-сервер увеличивает порядковый номер не после всякого динамического изменения.

DNS-серверы BIND 8 откладывают увеличение порядкового номера на пять минут, либо на 100 обновлений, в зависимости от того, какое условие будет выполнено раньше. Такая пауза предназначена для сопряжения способности DNS-сервера обрабатывать динамические обновления с его способностью передавать зоны: последняя требует гораздо больших временных ресурсов в случае крупных зон. Когда DNS-сервер наконец увеличивает порядковый номер зоны, то посылает NOTIFY-уведомление (о котором мы расскажем чуть позже в этой главе) вторичным DNS-серверам зоны, чтобы сообщить, что порядковый номер изменился.

DNS-серверы BIND 9 увеличивают порядковый номер после каждого обработанного динамического обновления.

Динамические обновления и файлы данных зон

Поскольку динамические обновления вносят долговременные изменения в зону, информацию о них следует сохранять на диске. Но перезапись файла данных зоны при каждом добавлении или удалении записи может оказаться для DNS-сервера чрезвычайно затруднительным занятием. Запись файла данных зон требует времени, а DNS-сервер вполне может получать десятки или сотни динамических обновлений каждую секунду.

Поэтому при получении динамических обновлений DNS-серверы BIND 8 и 9 просто добавляют короткую запись обновления в файл журнала.[1] Изменения, разумеется, немедленно отражаются на копии зоны, которая хранится в памяти сервера. Но DNS-серверы могут записывать зону на диск только в определенные моменты времени (обычно это происходит каждый час). После этого DNS-серверы BIND 8 удаляют log-файл, поскольку он больше не нужен. (В этот момент копия зоны в памяти идентична тому, что записано в файле данных зоны.) DNS-серверы BIND 9 оставляют log-файл, поскольку он используется также для инкрементальной передачи зоны, о которой мы поговорим чуть позже в этой главе. (DNS-серверы BIND 8 хранят информацию, необходимую для инкрементальной передачи зоны, в отдельном файле.)

В случае DNS-серверов BIND 8 имя log-файла создается путем добавления суффикса *.log* к имени файла данных зоны. В случае DNS-серверов BIND 9 используется суффикс *.jnl*. Поэтому, начав использовать динамические обновления, не удивляйтесь появлению этих файлов рядом с файлами данных зон – это совершенно естественно.

В случае DNS-сервера BIND 8 log-файлы должны исчезать каждый час (хотя могут практически сразу появляться снова, если DNS-сервер получает много динамических обновлений), а также при нормальном завершении работы DNS-сервера. В случае DNS-сервера BIND 9 log-файлы не исчезают никогда. Сервер любой из этих версий вносит изменения из log-файлов в зоны, если log-файл существует в момент запуска DNS-сервера.

Если кому-то интересно, log-файлы в BIND 8 поддаются прочтению людьми и содержат записи следующего вида:

```
;BIND LOG V8
[DYNAMIC_UPDATE] id 8761 from [192.249.249.3].1148 at 971389102 (named pid 17602):
zone:   origin movie.edu class IN serial 2000010957
update: {add} almostfamous.movie.edu. 600 IN A 192.249.249.215
```

Нельзя сказать того же о log-файлах BIND 9. По крайней мере, если речь идет о таких, как мы с вами, людях.

Списки управления доступом для обновлений

Учитывая ужасающие возможности, которые попадают в руки автора обновлений, очевидно, следует ограничивать доступ к ним, если они вообще используются. По умолчанию, DNS-серверы BIND 8 и BIND 9 не разрешают выполнять динамические обновления зон, для которых являются авторитативными. Чтобы воспользоваться динамическими

[1] Идея покажется знакомой всем, кому приходилось использовать журналируемую файловую систему.

обновлениями, следует добавить предписание *allow-update* или *update-policy* к оператору *zone* зоны, для которой необходимо разрешить динамические обновления.

allow-update в качестве аргумента принимает список отбора адресов. Обновления разрешается выполнять только адресу или адресам из этого списка. Разумной политикой является максимальное сокращение этого списка управления доступом:

```
zone "fx.movie.edu" {
    type master;
    file "db.fx.movie.edu";
    allow-update { 192.253.253.100; }; // только наш
                                       // DHCP-сервер
};
```

Обновления с TSIG-подписями

DNS-сервер BIND 9.1.0 и более поздних версий умеет производить ретрансляцию обновлений, так что возникает вопрос: какой смысл в списке управления доступом для IP-адресов? Если первичный DNS-мастер-сервер разрешает обновления с адресов вторичных DNS-серверов, значит, любые ретранслированные обновления будут разрешены, вне зависимости от того, кто являлся изначальным их автором. Это плохо.[1]

Ну, во-первых, существует возможность определить, *какие* обновления ретранслируются. Предписание *allow-update-forwarding* в качестве аргумента принимает список отбора адресов. Ретрансляция будет выполняться только для обновлений, поступающих с перечисленных IP-адресов. Так, следующий оператор *zone* разрешает ретрансляцию только тех обновлений, которые поступают из подсети факультета Special Effects:

```
zone "fx.movie.edu" {
    type slave;
    file "bak.fx.movie.edu";
    allow-update-forwarding { 192.253.254/24; };
};
```

Тем не менее, при использовании ретрансляции обновлений следует работать с динамическими обновлениями с TSIG-подписями. Подробно механизм TSIG будет рассмотрен только в главе 11 «Безопасность», а сейчас читателям необходимо знать лишь, что динамические обновления с TSIG-подписями содержат криптографическую подпись автора. Если происходит ретрансляция таких сообщений, то подпись сохраняется. В процессе проверки определяется имя ключа, который использовался для создания подписи обновления. Имя ключа похоже на

[1] BIND 9.1.0 даже предупреждает, что небезопасно использовать списки управления доступом, состоящие из IP-адресов.

доменное и довольно часто совпадает с доменным именем использующего этот ключ узла.

В DNS-серверах BIND 8.2 и более поздних версий список отбора адресов может содержать имена одного или нескольких TSIG-ключей:

```
zone "fx.movie.edu"
    type master;
    file "db.fx.movie.edu";
    allow-update { key dhcp-server.fx.movie.edu.; }; // разрешить только
                                                      // обновления, подписанные
                                                     // TSIG-ключом DHCP-сервера
};
```

Этот оператор позволяет автору обновления вносить любые изменения в зону *fx.movie.edu* – используя TSIG-ключ *dhcp-server.fx.movie.edu*. К сожалению, не существует способа ограничить автора с определенным TSIG-ключом набором исходных IP-адресов.

В BIND 9 реализован более совершенный, чем *allow-update*, механизм управления доступом, основанный также на TSIG-подписях. Использование этого механизма связано с новым предписанием оператора *zone*, *update-policy*. *update-policy* позволяет указать ключи, которым разрешено обновление, и записи, которые могут обновляться конкретными ключами. Это имеет смысл только для первичных DNS-мастер-серверов зоны, поскольку вторичные DNS-серверы должны просто ретранслировать обновления.

Обновление определяется именем ключа, используемого для подписи, а также доменным именем и типом записей, для которых это обновление предназначено. Синтаксис предписания *update-policy*:

```
(grant | deny) identity nametype name [types]
```

Значения вариантов *grant* и *deny* очевидны: разрешить или запретить конкретное динамическое обновление. Параметр *identity* определяет имя ключа, который используется для создания подписи. Параметр *nametype* может принимать значения:

name

Соответствие доменного имени, для которого производится обновление, имени в поле аргумента *name*.

subdomain

Соответствие доменного имени, для которого производится обновление, поддомену имени в поле аргумента *name* (обновляемое имя заканчивается значением из этого поля). (Разумеется, доменное имя должно входить в контекстную зону.)

wildcard

Соответствие доменного имени, для которого производится обновление, маске, определенной в поле аргумента *name*.

self

Соответствие доменного имени, для которого производится обновление, имени в поле *identity* (а не *name*), то есть совпадение доменного имени с именем ключа, который использовался для создания подписи обновления. Когда *nametype* принимает значение *self*, поле *name* игнорируется. Но несмотря на очевидную избыточность (мы сейчас увидим ее в примере), поле *name* должно присутствовать и в этом случае.

Естественно, *name* – доменное имя, соответствующее указанному варианту *nametype*. Если указать *wildcard* в качестве значения *nametype*, поле *name* должно содержать метку-маску.

Поле *types* является необязательным, и может содержать любой существующий тип записи (либо перечисление, с пробелом в качестве разделителя), кроме NXT. (Тип ANY является удобным сокращением для «всех типов, кроме NXT».) Если поле *types* отсутствует, происходит отбор всех типов записей, кроме SOA, NS, SIG и NXT.

Замечание по старшинству правил предписания *update-policy*: к динамическому обновлению применяется первое соответствие (не ближайшее, а точное).

Итак, если *mummy.fx.movie.edu* использует ключ *mummy.fx.movie.edu*, чтобы подписывать свои динамические обновления, мы можем запретить *mummy.fx.movie.edu* обновлять любые записи, кроме собственных, с помощью следующего оператора:

```
zone "fx.movie.edu" {
    type master;
    file "db.fx.movie.edu";
    update-policy { grant mummy.fx.movie.edu. self mummy.fx.movie.edu.; };
};
```

либо – запретить обновлять любые записи, кроме собственных адресных:

```
zone "fx.movie.edu" {
    type master;
    file "db.fx.movie.edu";
    update-policy { grant mummy.fx.movie.edu. self mummy.fx.movie.edu. A; };
};
```

В общем случае мы можем запретить всем клиентам обновлять что-либо, кроме собственных адресных записей, следующим образом:

```
zone "fx.movie.edu" {
    type master;
    file "db.fx.movie.edu";
    update-policy { grant *.fx.movie.edu. self fx.movie.edu. A; };
};
```

Вот более сложный пример: мы разрешаем всем клиентам изменять произвольные записи, кроме SRV-записей, которые принадлежат доменному имени, совпадающему с именем ключа, но при этом разрешаем узлу *matrix.fx.movie.edu* обновлять SRV-записи, связанные с Windows 2000 (в поддоменах *_udp.fx.movie.edu*, *_tcp.fx.movie.edu*, *_sites.fx.movie.edu* и *_msdcs.fx.movie.edu*).

```
zone "fx.movie.edu" {
    type master;
    file "db.fx.movie.edu";
    update-policy {
        deny *.fx.movie.edu. self *.fx.movie.edu. SRV;
        grant *.fx.movie.edu. self *.fx.movie.edu. ANY;
        grant matrix.fx.movie.edu. subdomain _udp.fx.movie.edu. SRV;
        grant matrix.fx.movie.edu. subdomain _tcp.fx.movie.edu. SRV;
        grant matrix.fx.movie.edu. subdomain _sites.fx.movie.edu. SRV;
        grant matrix.fx.movie.edu. subdomain _msdcs.fx.movie.edu. SRV;
    };
};
```

Поскольку правила в предписании *update-policy* проверяются в порядке следования, клиенты не могут обновлять свои SRV-записи, хотя могут обновлять собственные записи любого другого типа.

Для тех, кому интересно, разница между:

```
grant identity subdomain fx.movie.edu
```

и:

```
grant identity wildcard *.fx.movie.edu:
```

в том, что первое правило разрешает ключу, указанному в поле *identity* изменять записи, связанные с *fx.movie.edu* (скажем, NS-записи этой зоны), а второе – нет.

Если существует необходимость воспользоваться преимуществами динамических обновлений с TSIG-подписями, но нет соответствующего программного обеспечения, можно применить свежую версию программы *nsupdate* – как именно, рассказано в следующей главе.

DNS NOTIFY (уведомления об изменениях зоны)

Традиционно вторичные DNS-серверы BIND самостоятельно опрашивали DNS-мастер-серверы, чтобы определить, не пора ли произвести получение зоны. Интервал опроса получил название *интервала обновления*. Прочие поля SOA-записи зоны также влияют на различные аспекты работы механизма опросов.

В случае использования схемы самостоятельных опросов может пройти полный интервал обновления до того момента, как вторичный узел обнаружит и получит новые данные зоны от основного DNS-сервера. Задержки такого рода могут привести к катастрофе в динамически меняющейся среде. Разве не было бы здорово, если бы первичный DNS-мастер-сервер мог *говорить* своим вторичным серверам, что данные зоны изменились? В конце концов, первичный DNS-мастер-сервер *знает*, что данные изменились; данные были перезагружены, а сервер изучил параметр *mtime* (время изменения файла в файловой системе Unix) всех файлов данных зон, чтобы определить, какие именно изменились[1], либо сервер получил и обработал динамическое обновление. Первичный DNS-мастер-сервер мог бы посылать уведомления непосредственно после перезагрузки или завершения обработки обновления, вместо того чтобы ждать, когда закончится интервал обновления, и вторичные DNS-серверы самостоятельно произведут синхронизацию.

В документе RFC 1996 был предложен механизм, который позволяет первичным DNS-мастер-серверам посылать вторичным серверам уведомления об изменениях данных зон. Этот механизм, получивший название DNS NOTIFY, реализован в DNS-серверах BIND 8 и 9.

DNS NOTIFY работает следующим образом: когда первичный DNS-мастер-сервер замечает, что порядковый номер зоны изменился, то посылает специальное объявление всем DNS-серверам, которые являются для этой зоны вторичными. Перечень вторичных серверов для зоны определяется путем выборки из NS-записей тех, которые указывают на DNS-сервер из поля MNAME SOA-записи зоны либо на доменное имя локального узла.

Когда DNS-сервер замечает изменение? Перезапуск первичного DNS-мастер-сервера приводит к посылке текущих порядковых номеров зон всем вторичным серверам этих зон, поскольку первичный DNS-мастер-сервер не в состоянии определить, были ли изменены файлы данных перед его запуском. Перезагрузка одной или нескольких зон с обновленными порядковыми номерами приводит к отправке уведомлений вторичным серверам этих зон. Уведомление также посылается в результате обработки динамического обновления, в результате которой увеличивается порядковый номер зоны.

Специальное NOTIFY-объявление определяется кодом операции и заголовком DNS-сообщения. Код операций для большинства запросов – QUERY. NOTIFY-сообщения, включая объявления и ответы, имеют специальный код операции, NOTIFY (сюрприз!). Во всем остальном сообщения NOTIFY очень похожи на запрос SOA-записи для зоны: в них указывается доменное имя зоны, порядковый номер которой изменил-

[1] Исключением является случай перезагрузки единственной зоны, когда сервер проверяет время модификации только для файла перезагружаемой зоны.

ся, ее класс, а также тип SOA. Помимо этого, устанавливается бит авторитативности.

Когда вторичный сервер получает NOTIFY-уведомление для зоны от одного из DNS-серверов, который считается первичным мастером, то посылает ответное NOTIFY-сообщение. Ответное сообщение говорит мастер-серверу о том, что уведомление для зоны получено, его не следует посылать повторно. После этого вторичный DNS-сервер работает точно так же, как в случае срабатывания таймера обновления: запрашивает SOA-запись зоны, которая изменилась, у основного DNS-сервера. Если порядковый номер больше хранимого, происходит получение новой копии зоны.

Почему вторичный сервер не верит основному на слово – в том, что зона действительно изменилась? Вполне возможно, что какой-то злодей подделал NOTIFY-уведомления, полученные узлами, чтобы перегрузить основной DNS-сервер ненужными процессами передачи зон, произведя атаку DoS (denial-of-service, отказ от обслуживания).

Документ RFC 1996 предписывает вторичному серверу – в случае, если получение зоны произошло, – послать собственные уведомления NOTIFY прочим авторитативным серверам зоны. Идея в основе этого предписания такова: первичный DNS-мастер-сервер мог уведомить не все вторичные DNS-серверы, поскольку некоторые из них могут не иметь прямой связи с основным, общаясь только с другими вторичными серверами. Такое поведение реализовано в BIND 8.2.3 и BIND 9, но не в более ранних версиях BIND 8. Вторичные DNS-серверы более ранних версий BIND 8 не посылают NOTIFY-сообщений, если не произведена специальная настройка.

Вот как это работает на практике. В нашей сети первичным DNS-мастер-сервером для зоны *movie.edu* является *terminator.movie.edu*, а *wormhole.movie.edu* и *zardoz.movie.edu* – вторичные DNS-серверы (рис. 10.1).

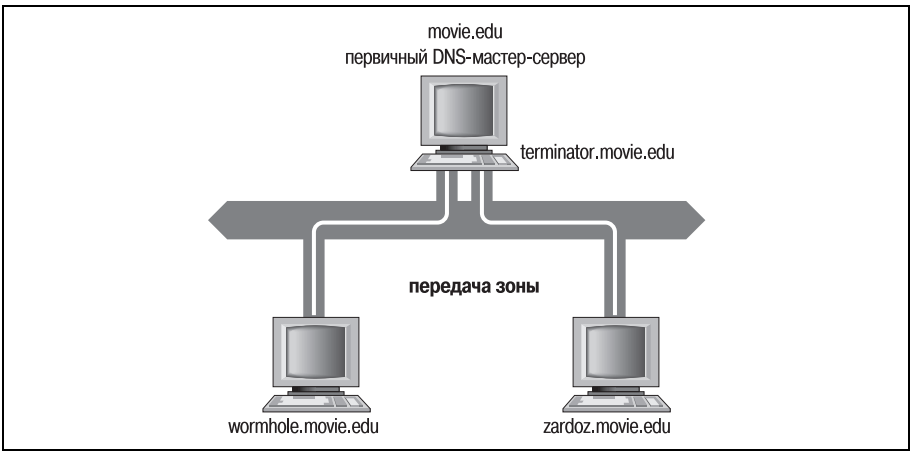

Рис. 10.1. *movie.edu, передача зоны*

Когда копия зоны *movie.edu* на узле *terminator.movie.edu* подвергается редактированию и перезагрузке либо для нее выполняется динамическое обновление, *terminator.movie.edu* посылает NOTIFY-объявления узлам *wormhole.movie.edu* и *zardoz.movie.edu*. Оба вторичных сервера отвечают узлу *terminator.movie.edu*, что информация получена. Затем они проверяют, увеличился ли порядковый номер зоны *movie.edu*, и если это так, производят получение зоны. Если *wormhole.movie.edu* и *zardoz.movie.edu* работают под управлением DNS-серверов BIND 8.2.3 или BIND 9, то после получения новой версии зоны они посылают NOTIFY-объявления друг другу, сообщая об изменениях. Но поскольку *wormhole.movie.edu* не является DNS-мастер-сервером для *zardoz.movie.edu* (в контексте зоны *movie.edu*) и обратное также неверно, каждый из серверов игнорирует NOTIFY-сообщение, полученное от второго.

DNS-сервер BIND 8 заносит информацию о сообщениях NOTIFY в log-файл *syslog*. Эти сообщения были записаны в log-файл на узле *terminator.movie.edu* после перезагрузки зоны *movie.edu*:

```
Oct 14 22:56:34 terminator named[18764]: Sent NOTIFY for "movie.edu IN SOA
2000010958" (movie.edu); 2 NS, 2 A
Oct 14 22:56:34 terminator named[18764]: Received NOTIFY answer (AA) from
192.249.249.1 for "movie.edu IN SOA"
Oct 14 22:56:34 terminator named[18764]: Received NOTIFY answer (AA) from
192.249.249.9 for "movie.edu IN SOA"
```

Первая запись отражает первое NOTIFY-объявление, посланное сервером *terminator.movie.edu* двум узлам (2 NS), смысл которого в том, что порядковый номер зоны *movie.edu* теперь 2000010958. Следующие две строки отражают получение подтверждений от вторичных DNS-серверов. (DNS-серверы BIND 9 обычно не регистрируют в log-файле NOTIFY-события.)

Взглянем теперь на более сложную схему синхронизации зоны. В этом примере *a* является первичным DNS-мастер-сервером зоны и мастер-сервером для *b*, при этом *b* является мастер-сервером для *c*. Помимо этого, у *b* есть два сетевых интерфейса (рис. 10.2).

В этом варианте *a* уведомляет узлы *b* и *c* о том, что зона обновилась. Затем *b* проверяет, действительно ли увеличился порядковый номер зоны, и инициирует получение зоны. Однако *c* игнорирует NOTIFY-сообщение от *a*, поскольку *a* не сконфигурирован как DNS-мастер-сервер для *c* (в отличие от *b*). Если узел *b* работает под управлением DNS-сервера BIND 8.2.3 или BIND 9 либо явным образом настроен уведомлять узел *c*, то после завершения процесса передачи зоны *b* посылает уведомление NOTIFY узлу *c*, которое приводит к проверке узлом *c* порядкового номера, хранимого для зоны сервером *b*. Если узел *c* также работает под управлением BIND 8.2.3 или BIND 9, то после получения зоны *c* отправляет NOTIFY-объявление узлу *b*, которое тот, разумеется, игнорирует.

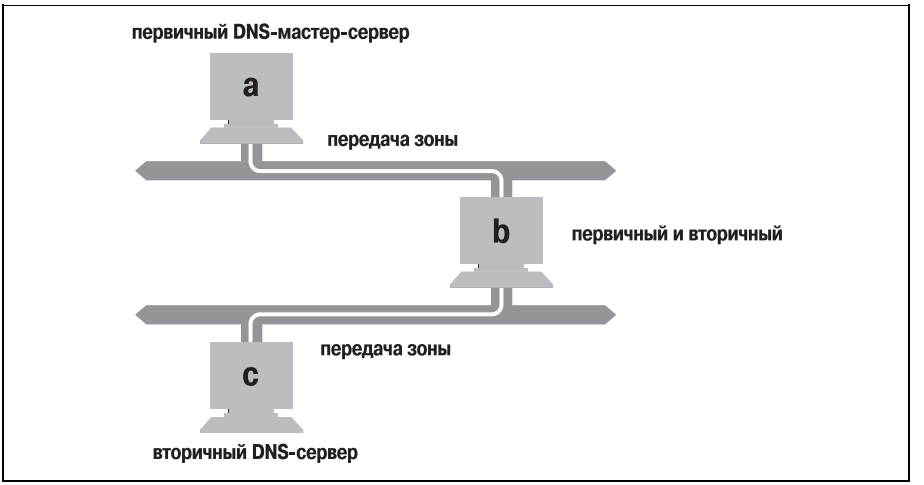

Рис. 10.2. *Более сложный пример синхронизации зоны*

Отметим, что если существует хоть малейшая вероятность того, что узел *c* получит NOTIFY-уведомление с другого сетевого интерфейса *b*, в соответствующем предписании *masters* для DNS-сервера *c* должны быть указаны адреса обоих сетевых интерфейсов. В противном случае *c* будет игнорировать NOTIFY-сообщения, получаемые через неизвестный сетевой интерфейс.

Вторичные серверы BIND 4 (и прочие, не поддерживающие механизм NOTIFY) будут возвращать ошибку Not Implemented (NOTIMP, реализация отсутствует). Обратите внимание, что сервер Microsoft DNS *поддерживает* работу с механизмом DNS NOTIFY.

DNS NOTIFY по умолчанию работает в BIND 8 и 9, но можно глобально отключить этот механизм с помощью предписания *notify*:

```
options {
    notify no;
};
```

Можно также включать или выключать NOTIFY для отдельных зон. К примеру, нам известно, что все вторичные серверы зоны *fx.movie.edu* работают под управлением BIND 4, а значит не понимают NOTIFY-объявлений. Следующий оператор *zone*:

```
zone "fx.movie.edu" {
    type master;
    file "db.fx.movie.edu";
    notify no;
};
```

предотвращает посылку бесполезных NOTIFY-сообщений вторичным DNS-серверам зоны *fx.movie.edu*. Частные настройки NOTIFY для зоны имеют более высокий приоритет, чем глобальные. К сожалению,

ни в BIND 8, ни в BIND 9 нет возможности выключать механизм NOTIFY для отдельных серверов.

В BIND 8 и 9 существует даже возможность добавлять в «NOTIFY-список» DNS-серверы помимо тех, что указаны в NS-записях зоны. К примеру, у нас может быть один или несколько незарегистрированных вторичных DNS-серверов (случай описан в главе 8 «Развитие домена») и мы хотим, чтобы они быстро реагировали на изменение зоны. Как вариант речь может идти о более старом вторичном DNS-сервере BIND 8, который является мастер-сервером для другого вторичного и должен передавать этому вторичному узлу NOTIFY-сообщения.

Чтобы добавить сервер в список рассылки NOTIFY-объявлений, следует воспользоваться предписанием *also-notify* оператора *zone*:

```
zone "fx.movie.edu" {
    type slave;
    file "bak.fx.movie.edu";
    notify yes;
    also-notify { 15.255.152.4; };  // Это вторичный сервер BIND 8, который
                                    // следует настроить на посылку уведомлений
                                    // своему вторичному DNS-серверу
};
```

Начиная с BIND 8.2.2, *also-notify* может использоваться также и в качестве предписания в операторе *options*. Такое предписание будет действовать для всех зон со включенным механизмом NOTIFY (и для которых отсутствуют частные предписания *also-notify*).

Начиная с BIND 9.1.0, в качестве аргумента предписания *notify* можно указать ключевое слово *explicit*; что приводит к подавлению посылки NOTIFY-сообщений для всех DNS-серверов, *кроме* тех, что перечислены в списке *also-notify*. Помимо этого, можно использовать предписание *allow-notify*, чтобы DNS-сервер принимал NOTIFY-сообщения не только от DNS-мастер-серверов зоны:

```
options {
    allow-notify { 192.249.249.17; }; // let 192.249.249.17 send NOTIFY msgs
};
```

В качестве предписания оператора *options allow-notify* относится ко всем вторичным зонам. В качестве частного предписания оператора *zone allow-notify* имеет приоритет больший, чем глобальное предписание *allow-notify* и действует в пределах текущей зоны.

Инкрементальная передача зоны (IXFR)

Итак, мы используем динамические обновления и механизм NOTIFY, и когда мы обновляем зоны – в соответствии с изменениями в сети – обновления быстро распространяются на данные, хранимые всеми авторитативными DNS-серверами для этих зон. Чего еще можно желать?

Оказывается, есть чего. Представим себе, что крупная зона динамически обновляется с ужасающей частотой. Такое вполне может случиться: администратор заведует крупной зоной, у которой тысячи клиентов, и вдруг руководство посещает идея оборудовать клиентов системой Windows 2000 и начать использовать DHCP. Теперь каждый из клиентов будет обновлять свои адресные данные в зоне, а контроллеры домена будут обновлять записи, предоставляющие клиентам информацию о существующих службах. (Гораздо больше материала по Windows 2000 содержится в главе 16 «Обо всем понемногу».)

Каждый раз, когда первичный DNS-мастер-сервер производит обновление, связанное с увеличением порядкового номера зоны, он посылает NOTIFY-объявления вторичным DNS-серверам. Каждый раз при получении уведомлений вторичные серверы проверяют, не увеличился ли порядковый номер зоны на основном сервере, и, возможно, производят синхронизацию. Если зона крупная, синхронизация займет определенное время, а за это время она снова может обновиться. И так вторичные серверы будут бесконечно долго заниматься исключительно синхронизацией! В лучшем случае DNS-серверы потратят много времени на передачу данных зоны, причем весьма вероятно, что изменения были довольно незначительными по размеру (скажем, добавилась адресная запись для клиента).

Инкрементальная передача зоны (incremental zone transfer или IXFR) решает эту проблему, позволяя вторичным DNS-серверам сообщать основным, какие версии зон ими хранятся, и запрашивать только изменения зоны между хранимыми версиями и текущими. Это позволяет весьма значительно сократить объем передаваемых данных и время передачи.

Тип запроса для инкрементальной передачи зон – IXFR вместо AXFR (тип запроса для полной передачи зоны), при этом запрос содержит хранимую на вторичном сервере SOA-запись зоны в разделе авторитативности. Когда основной DNS-сервер получает запрос инкрементальной передачи зоны, он производит поиск записей об изменениях зоны, то есть разницы между существующей зоной и копией, хранимой вторичным DNS-сервером. Если информация отсутствует, происходит полная передача зоны. В противном случае передается только найденная разница.

Ограничения IXFR

Звучит неплохо? И работает отлично! Но IXFR имеет несколько ограничений, о которых читателям следует знать. Во-первых, этот механизм толком заработал начиная с BIND 8.2.3. Все DNS-серверы BIND 9 содержат реализацию IXFR, которая хорошо сосуществует с реализацией BIND 8.2.3.

Кроме того, IXFR лучше всего работает, когда данные зоны изменяются только с помощью динамических обновлений. Выполнение динамических обновлений приводит к появлению записей о внесенных изменениях и порядковом номере, которому эти изменения соответствуют, – это именно та информация, которую основной сервер должен послать вторичному, получив запрос IXFR. Но первичный DNS-мастер-сервер BIND, перезагрузивший весь файл данных зоны, неспособен вычислить разницу между этой зоной и предыдущей ее копией. Вторичный DNS-сервер BIND, при проведении полной передачи зоны, также не в состоянии понять, в чем разница между этой копией зоны и предыдущей.

Таким образом, чтобы по максимуму использовать преимущества IXFR, следует изменять зону только с помощью динамических обновлений и никогда не редактировать файл данных вручную.

Файлы IXFR

DNS-серверы BIND 8 ведут журнал изменений IXFR, отдельный от файла динамических обновлений. Как и файл журнала динамических обновлений, журнал IXFR обновляется при каждом получении обновления серверов. В отличие от журнала динамических обновлений, журнал IXFR никогда не удаляется, хотя DNS-сервер можно настроить на усечение этого файла при достижении им определенного размера. По умолчанию файл IXFR-журнала получает имя файла данных зоны с добавленным суффиксом *.ixfr*.

DNS-серверы BIND 9 используют файл журнала динамических обновлений для сборки ответных IXFR-сообщений и поддержки целостности данных зоны. Поскольку первичный DNS-мастер-сервер не знает, когда именно может понадобиться запись об определенной модификации зоны, то не удаляет файл журнала. Вторичный DNS-сервер BIND 9 удаляет файл журнала при получении AXFR-запроса для зоны, поскольку после передачи полной копии зоны можно перестать отслеживать изменения, приведшие к получению этой копии.

Настройка IXFR в BIND 8

Настройка IXFR в BIND 8 достаточно прямолинейна. Во-первых, на основном DNS-сервере следует воспользоваться предписанием *maintain-ixfr-base* оператора *options*, чтобы предписать хранение файлов IXFR-журналов для всех зон – даже тех, для которых DNS-сервер является вторичным, поскольку могут существовать прочие вторичные DNS-серверы, способные посылать ему IXFR-запросы:

```
options {
    directory "/var/named";
    maintain-ixfr-base yes;
};
```

Теперь следует объяснить вторичным серверам, что IXFR-обновления доступны при работе с основным сервером. Это делается с помощью нового предписания *support-ixfr*:

```
server 192.249.249.3 {
    support-ixfr yes;
};
```

И это практически все, если нет необходимости изменять имя файла IXFR-журнала на первичном DNS-мастер-сервере. Если такая необходимость существует, воспользуйтесь предписанием *ixfr-base* оператора *zone*:

```
zone "movie.edu" {
    type master;
    file "db.movie.edu";
    ixfr-base "ixfr.movie.edu";
};
```

Ах да, можно настроить DNS-сервер на усечение файла IXFR-журнала в момент превышения определенного размера:[1]

```
options {
    directory "/var/named";
    maintain-ixfr-base yes;
    max-ixfr-log-size 1M;    // усечение IXFR-журнала до 1 мегабайта
};
```

Как только размер IXFR-журнала превысит указанный на 100 килобайт, произойдет усечение до указанного размера. Буфер в 100 килобайт предотвращает выполнение усечения после каждого успешного обновления.

Эффективность синхронизации зон можно увеличить еще больше, используя формат передачи *many-answers*. Подробности в разделе «Повышение эффективности при передаче зон».

Настройка IXFR в BIND 9

Настройка IXFR для основного DNS-сервера BIND 9 еще проще, поскольку делать вообще ничего не надо: механизм включен по умолчанию. Если существует необходимость отключить механизм для определенного вторичного узла (что маловероятно, поскольку вторичный сервер должен *запросить* инкрементальную передачу зоны), воспользуйтесь предписанием *provide-ixfr* оператора *server*, для которого по умолчанию устанавливается значение *yes*:

```
server 192.249.249.1 {
```

[1] В версиях до BIND 8.2.3 размер может указываться только в байтах (вместо «1M») из-за существующей ошибки в коде.

```
        provide-ixfr no;
    };
```

provide-ixfr можно использовать также в качестве предписания оператора *options*, в этом случае оно относится ко всем вторичным DNS-серверам, для которых не существует частных предписаний *provide-ixfr* в операторах *server*.

Поскольку основные DNS-серверы BIND 9 выполняют передачу зоны в формате *many-answers* по умолчанию, нет необходимости дополнительно настраивать этот аспект с помощью *transfer-format*.

Представляет интерес предписание *request-ixfr*, которое можно указывать в операторах *options* и *server*. Для смешанного набора IXFR-ориентированных и неIXFR-ориентированных DNS-мастер-серверов можно настроить вторичные DNS-серверы на использование существующих способностей основных серверов:

```
options {
    directory "/var/named";
    request-ixfr no;
};

server 192.249.249.3 {
    request-ixfr yes;   // из всех основных только terminator поддерживает IXFR
};
```

В BIND 9 не реализована поддержка предписания *max-ixfr-log-size*.

Ретрансляция

В определенных случаях крупные объемы внешнего трафика нежелательны либо из-за значительной суммы оплаты канала, напрямую связанной с этими объемами, либо из-за медленного канала связи с большими задержками, например, при использовании удаленным офисом компании спутникового канала для подключения к корпоративной сети. В таких случаях внешний DNS-трафик желательно свести к минимуму. В BIND существует механизм, позволяющий это сделать: *forwarders* (ретрансляторы).

Ретрансляторы могут также использоваться при необходимости возложить ответственность за разрешение имен на конкретный сервер. К примеру, если только один узел сети подключен к Интернету, и на этом узле работает DNS-сервер, все прочие DNS-серверы можно настроить на использование этого узла в качестве ретранслятора при поиске данных для доменных имен. (Более подробно такое применение ретрансляторов мы обсудим в главе 11, когда речь пойдет о брандмауэрах.)

Если сделать один или несколько серверов площадки ретрансляторами, DNS-серверы будут посылать внешние запросы прежде всего этим серверам. Идея состоит в том, что ретранслятор обрабатывает все внешние запросы для площадки, параллельно занимаясь накоплени-

ем кэшированной информации. Для произвольного запроса данных из внешней зоны существует высокая вероятность того, что ретранслятор в состоянии ответить на запрос данными из кэша, что избавляет прочие серверы от необходимости посылать внешние запросы. Чтобы сделать конкретный DNS-сервер ретранслятором, не нужны какие-либо специальные действия – настраивать следует все *прочие серверы* площадки, чтобы они направляли свои запросы ретрансляторам.

Рабочий процесс несколько изменяется для первичного или вторичного DNS-сервера, который настроен на использование ретранслятора. Если клиент запрашивает записи, которые входят в авторитативность DNS-сервера, либо содержатся в кэшированных данных, DNS-сервер возвращает информацию самостоятельно: в этой части ничего не изменилось. Но если записи отсутствуют в базе данных, DNS-сервер посылает запрос ретранслятору и в течение небольшого промежутка времени ожидает ответа, после чего продолжает нормальную работу и самостоятельно посылает запрос удаленным DNS-серверам. Разница в данном случае лишь в том, что DNS-сервер посылает ретранслятору *рекурсивные* запросы, ожидая, что ретранслятор самостоятельно найдет ответ. Во всех прочих случаях DNS-сервер посылает другим серверам *нерекурсивные* запросы и самостоятельно работает со ссылками, направляющими его к другим DNS-серверам.

Ниже приводится предписание *forwarders* для BIND 8 и 9, а также эквивалентная инструкция загрузочного файла BIND 4 – для DNS-серверов зоны *movie.edu*. *wormhole.movie.edu* и *terminator.movie.edu* являются ретрансляторами площадки. Следующее предписание *forwarders* добавляется в файлы настройки всех DNS-серверов, за исключением тех, которые являются ретрансляторами:

```
options {
    forwarders { 192.249.249.1; 192.249.249.3; };
};
```

Соответствующая инструкция BIND 4:

```
forwarders 192.249.249.1 192.249.249.3
```

При использовании ретрансляторов старайтесь максимально упрощать настройки, иначе можно очень сильно запутаться в полученной структуре.

Избегайте объединения ретрансляторов в цепочки. Не стоит делать так, чтобы DNS-сервер A передавал запросы серверу B, а сервер B – серверу C (или, того хуже, обратно A). Это может привести к длительным паузам в процессе разрешения имен, а также делает хрупкой всю структуру: если произойдет сбой на любом из ретрансляторов цепи, разрешение имен будет затруднено либо станет попросту невозможным.

Более ограниченный DNS-сервер

Можно ограничить DNS-серверы еще более – запретив даже *пытаться* посылать запросы внешним DNS-серверам, если ретрансляторы не работают или не отвечают. Это можно сделать, настроив DNS-серверы на работу в режиме *forward-only*. DNS-сервер в режиме forward-only – это вариация на тему DNS-сервера, использующего ретрансляторы. Он по-прежнему отвечает на запросы, исходя из авторитативных и кэшированных данных, но во всем остальном *полностью* полагается на ретрансляторы, не пытаясь связаться с другими DNS-серверами в поисках ответа. Вот пример файла настройки для DNS-сервера, работающего в режиме эксклюзивной ретрансляции:

```
options {
    forwarders { 192.249.249.1; 192.249.249.3; };
    forward only;
};
```

То же для DNS-сервера BIND 4:

```
forwarders 192.249.249.1 192.249.249.3
options forward-only
```

DNS-серверы BIND до версии 4.9 предоставляют ту же функциональность с помощью инструкции *slave* (вместо *options forward-only*):

```
forwarders 192.249.249.1 192.249.249.3
slave
```

Не следует путать старое значение термина «slave» с имеющим хождение в наше время. В DNS-серверах BIND 4 слово «slave» являлось синонимом для «forward-only». Сегодня «slave» – это вторичный DNS-сервер, получающий данные зоны от первичного мастер-сервера.

В случае использования режима forward-only должно присутствовать предписание *forwarders*. Иначе нет смысла устанавливать режим forward-only. Если настроить DNS-сервер BIND версии более ранней, чем 8.2.3, на работу в режиме forward-only, возможно, имеет смысл указать IP-адреса ретрансляторов по несколько раз. Для DNS-сервера BIND 8 это будет выглядеть так:

```
options {
    forwarders { 192.249.249.1; 192.249.249.3;
                 192.249.249.1; 192.249.249.3; };
    forward only;
};
```

Для DNS-сервера BIND 4 так:

```
forwarders 192.249.249.1 192.249.249.3 192.249.249.1 192.249.249.3
options forward-only
```

Этот DNS-сервер вступает в контакт с каждым из ретрансляторов лишь единожды и в течение короткого промежутка времени ожидает ответа. Повторное включение ретрансляторов в список позволяет DNS-серверу *повторно посылать* запросы ретрансляторам и увеличивает суммарное время ожидания ответа.

Однако следует задуматься, действительно ли использование DNS-сервера в режиме forward-only имеет *какой-то* смысл. Такой DNS-сервер полностью зависим от ретрансляторов. Практически такого же результата можно добиться и не используя DNS-сервера вовсе: создайте файл *resolv.conf*, содержащий инструкции *nameserver* для используемых ретрансляторов. В этом случае зависимость от ретрансляторов сохраняется, но теперь приложения посылают запросы ретрансляторам напрямую, и нет необходимости содержать DNS-сервер, который будет передавать эти запросы по поручению приложений. При этом теряется локальное кэширование и сортировка адресов, но уменьшается суммарная сложность настройки структуры DNS-площадки в результате сокращения числа DNS-серверов.

Зоны ретрансляции

Традиционно использование ретрансляторов работало по формуле «все или ничего»: либо следовало использовать ретрансляторы для разрешения всех запросов, которые не могут быть разрешены отдельным DNS-сервером, либо не использовать ретрансляторы вовсе. Однако существуют случаи, когда было бы удобно иметь более тонкое управление ретрансляцией. К примеру, можно было бы производить разрешение определенных доменных имен с помощью определенного ретранслятора, а для всех прочих имен – итеративное.

В BIND 8.2 появился новый механизм – *зоны ретрансляции*, который позволяет настроить DNS-сервер на использование ретрансляторов только при поиске для определенных доменных имен. (Поддержка зон ретрансляции в BIND 9 появилась в версии 9.1.0.) К примеру, DNS-сервер может быть настроен на передачу всех запросов для доменных имен с суффиксом *pixar.com* паре DNS-серверов компании Pixar:

```
zone "pixar.com" {
    type forward;
    forwarders { 138.72.10.20; 138.72.30.28; };
};
```

Почему возникает необходимость явным образом указывать DNS-серверы, когда настраиваемый DNS-сервер мог бы самостоятельно найти DNS-серверы зоны *pixar.com*, пользуясь информацией о делегировании из зоны *com*? Представьте себе, что у нас есть прямой канал в компанию Pixar и необходимо использовать специальный набор DNS-серверов, доступных только в пределах нашей сети, которые будут заниматься разрешением доменных имен зоны *pixar.com*.

Правила ретрансляции содержатся в операторе *zone*, но выполняются для всех доменных имен, которые заканчиваются именем указанной зоны. Вне зависимости от того, входит ли доменное имя, *foo.bar.pixar.com*, в зону *pixar.com*, для него выполняется правило ретрансляции, поскольку имя заканчивается на *pixar.com* (или входит в домен *pixar.com* – кому что больше нравится).

Существует еще одна разновидность зоны ретрансляции, в некотором смысле противоположная только что описанной. Эта разновидность позволяет указывать, какие запросы *не* передаются ретранслятору. Следовательно, правило применимо только для DNS-серверов, ретрансляторы для которых указаны в операторе, *options*, то есть распространяется на все запросы.

Такие зоны ретрансляции настраиваются с помощью оператора *zone*, но не типа *forward*. Вместо этого используются предписания *forwarders* для обычных зон – основных, вторичных или зон-заглушек. Чтобы произвести отказ от ретрансляции, настройка которой произведена в операторе *options*, можно указать пустой список ретрансляторов:

```
options {
    directory "/var/named";
    forwarders { 192.249.249.3; 192.249.249.1; };
};

zone "movie.edu" {
    type slave;
    masters { 192.249.249.3; };
    file "bak.movie.edu";
    forwarders {};
};
```

Минуточку – но зачем отключать ретрансляцию в зоне, для которой сервер является авторитативным? Разве DNS-сервер не ответит на запрос без использования ретранслятора?

Вспомним, что правила ретрансляции применяются ко всем запросам для всех доменных имен, которые заканчиваются доменным именем зоны. Так что данное правило ретрансляции касается только запросов для доменных имен из делегированных поддоменов *movie.edu*, например *fx.movie.edu*. Без такого правила ретрансляции данный DNS-сервер просто передал бы запрос для имени *matrix.fx.movie.edu* DNS-серверу 192.249.249.3 или 192.249.249.1. В присутствии такого правила используются NS-записи поддомена из зоны *movie.edu*, и запросы направляются DNS-серверам зоны *fx.movie.edu*.

Зоны ретрансляции невероятно полезны при работе с брандмауэрами Интернета, как мы увидим в следующей главе.

Выбор ретранслятора

В случае использования DNS-серверов BIND 8.2.3 нет необходимости упоминать ретрансляторы более одного раза. Эти DNS-серверы необязательно опрашивают ретрансляторы в порядке их перечисления; они считают DNS-серверы из списка «кандидатами» для ретрансляции и выбирают сервер на основе времени передачи сигнала, то есть времени получения ответов на предшествующие запросы.

Это является преимуществом для случаев, когда один из ретрансляторов выходит из строя, особенно если он первый в списке. Более старые версии BIND продолжали попытки получить ответ от неработающего ретранслятора, делая паузу, прежде чем перейти к следующему ретранслятору из списка. BIND 8.2.3 очень быстро понимает, что ретранслятор не отвечает, и в следующий раз в качестве первого выбирает другой DNS-сервер.

К сожалению, в BIND 9 пока еще не реализован этот, более интеллектуальный, способ выбора ретрансляторов, хотя происходит переключение между ретрансляторами при необходимости.

Виды

В BIND 9 появились *виды* (*views*), еще один механизм, исключительно полезный в сетях, защищенных брандмауэрами. Виды позволяют использовать различные настройки DNS-сервера при общении с различными наборами узлов. Это в особенности удобно, если DNS-сервер работает на узле, получающем запросы как от внутренних узлов, так и от узлов сети Интернет (этот вариант будет рассмотрен в следующей главе).

Если отсутствует явная настройка видов, BIND 9 автоматически создает единственный неявный вид, который и доступен всем клиентам, посылающим запросы. Чтобы создать вид явным образом, следует воспользоваться оператором *view*, аргументом которого является имя вида:

```
view "internal" {
};
```

Имя вида может быть практически любым, но хорошей практикой является использование описательных имен. Кавычки, заключающие в себя имя вида, не являются обязательными, но полезно их использовать в целях предотвращения конфликтов имен видов с зарезервированными ключевыми словами BIND (такими как «internal», например). Оператор *view* может следовать за любым оператором *options*, хотя и необязательно непосредственно за таковым.

Перечисление узлов, которые могут «видеть» конкретный вид, производится с помощью предписания *match-clients view*, принимающего

список отбора адресов в качестве аргумента. Если набор узлов не указан с помощью *match-clients*, вид доступен для всех узлов.

Предположим, мы создаем специальный вид зоны *fx.movie.edu*, который будет доступен только факультету Special Effects. Мы можем создать вид, доступный только узлам нашей подсети:

```
view "internal" {
    match-clients { 192.253.254/24; };
};
```

Чтобы сделать настройки более читаемыми, можно воспользоваться оператором *acl*:

```
acl "fx-subnet" { 192.253.254/24; };

view "internal" {
    match-clients { "fx-subnet"; };
};
```

Однако следует убедиться, что ACL-список определен *вне* вида, поскольку оператор *acl* пока еще не может использоваться в операторе *view*.

Что же может использоваться в операторе *view*? Практически все остальные операторы. Можно определять зоны с помощью операторов *zone*, описывать DNS-серверы с помощью операторов *server*, а также настраивать ключи TSIG с помощью операторов *key*. В пределах вида можно использовать большинство предписаний оператора *options*, не заключая их в этот оператор:

```
acl "fx-subnet" { 192.253.254/24; };

view "internal" {
    match-clients { "fx-subnet"; };
    recursion yes;  // включить рекурсию для этого вида
                    // (рекурсия отключена глобально, в операторе options)
};
```

Значения параметров настройки, указываемые в пределах вида, замещают глобально определенные значения с идентичными именами (скажем, определенные в операторе *options*) – для узлов из списка *match-clients*.

Полный перечень инструкций, допустимых в пределах оператора *view* для используемой версии BIND 9 (перечень меняется от версии к версии), содержится в файле *doc/misc/options* дистрибутива BIND.

Вот полный файл *named.conf* лаборатории специальных эффектов, который даст читателям представления о потенциале видов:

```
options {
    directory "/var/named";
};

acl "fx-subnet" { 192.253.254/24; };
```

```
view "internal" {  // внутренний вид наших зон

    match-clients { "fx-subnet"; };

    zone "fx.movie.edu" {
        type master;
        file "db.fx.movie.edu";
    };

    zone "254.253.192.in-addr.arpa" {
        type master;
        file "db.192.253.254";
    };
};

view "external" {  // вид наших зон, доступный внешнему миру

    match-clients { any; };  // неявно определено
    recursion no;            // рекурсия не должна запрашиваться извне
    zone "fx.movie.edu" {
        type master;
        file "db.fx.movie.edu.external";  // внешний файл данных зоны
    };

    zone "254.254.192.in-addr.arpa" {
        type master;
        file "db.192.253.254.external";   // внешний файл данных зоны
    };
};
```

Обратите внимание, что в каждом виде существуют зоны *fx.movie.edu* и *254.253.192.in-addr.arpa*, но файлы данных зон для «внутреннего» и «внешнего» видов используются различные. Это позволяет показывать внешнему миру не то «лицо», которое доступно нам.

Порядок следования операторов *view* важен, поскольку клиент с определенным IP-адресом будет наблюдать первый вид, разрешенный к наблюдению для этого адреса. Если бы «external» предшествовал виду «internal», вид «internal» никогда бы не использовался, поскольку внешний вид разрешен к наблюдению всеми адресами.

И последнее замечание по видам (до того, как мы снова к ним вернемся в следующей главе): если создан хотя бы один оператор *view*, все операторы *zone* должны быть явно включены в один из видов.

Round Robin: распределение нагрузки

DNS-серверы, появившиеся после BIND 4.9, официально включают функциональность, связанную с распределением нагрузки, которая прежде существовала только в виде заплат к BIND. Брайан Бичер (Bryan Beecher) создал заплаты к BIND 4.8.3, реализующие – как он это назвал – «перетасовку адресных записей». Речь шла об адресных

записях специального типа, которые DNS-сервер возвращал в различном порядке в различных ответных сообщениях. К примеру, доменное имя *foo.bar.baz* имело три «тасуемых» IP-адреса, 192.168.1.1, 192.168.1.2 и 192.1.168.3, и соответствующим образом обновленный DNS-сервер мог возвращать их сначала в таком порядке:

```
192.168.1.1 192.168.1.2 192.168.1.3
```

затем в таком:

```
192.168.1.2 192.168.1.3 192.168.1.1
```

и наконец – в таком:

```
192.168.1.3 192.168.1.1 192.168.1.2
```

прежде чем снова перейти к первому варианту перечисления и затем продолжать перестановки до бесконечности.

Такая функциональность невероятно полезна в случаях, когда существует набор эквивалентных сетевых ресурсов, например, зеркальных FTP-серверов, веб-серверов, терминальных серверов, а также необходимость в распределении нагрузки между этими серверами. Одно доменное имя является указателем на группу ресурсов: клиенты получают доступ к ресурсу по этому доменному имени, а DNS-сервер распределяет запросы между несколькими перечисленными IP-адресами.

Начиная с BIND 4.9 тасуемые адресные записи прекратили существование в виде самостоятельного типа данных, требующего специальной обработки. Вместо этого современный DNS-сервер производит перестановку адресов для любого доменного имени, с которым связано более одной A-записи. (Вообще говоря, DNS-сервер производит перестановки для записей любого типа, если таких записей для доменного имени существует больше одной.[1]) Поэтому существование записей:

```
foo.bar.baz.    60    IN    A    192.168.1.1
foo.bar.baz.    60    IN    A    192.168.1.2
foo.bar.baz.    60    IN    A    192.168.1.3
```

приводит для DNS-сервера версии 4.9 или более поздней к такому же результату, как в случае обновленного для работы с тасуемыми адресными записями сервера версии 4.8.3. В документации BIND процесс перестановки записей носит название *round robin*.

Хорошей идеей будет уменьшить время жизни для записей, как мы сделали в последнем примере. В этом случае, если адреса будут кэшированы промежуточным DNS-сервером, который не поддерживает перестановку адресов, они быстро устареют. Промежуточный DNS-сер-

[1] До появления BIND 9 PTR-записи не подвергались перестановке. В BIND 9 выполняется перестановка для всех типов записей.

вер будет вынужден снова запросить адрес для имени, и снова получит адресные записи в другом порядке.

Заметим, речь идет именно о распределении нагрузки, а не о ее балансировке, поскольку DNS-серверы выдают адреса в совершенно предсказуемом порядке, вне зависимости от реально существующей нагрузки или мощности серверов, обслуживающих запросы. В нашем примере сервер по адресу 192.168.1.3 может быть машиной 486DX33, работающей под управлением системы Linux, а два других сервера – суперкомпьютерами HP9000; но Linux-машина все равно будет получать треть всех запросов. Многократное перечисление адресов более мощных серверов ни к чему не приведет, поскольку BIND удаляет дублирующиеся записи.

Множественные CNAME-записи

Во времена расцвета DNS-серверов BIND 4 некоторым приходила в голову мысль обеспечивать перестановку с помощью множественных CNAME-записей (вместо множественных адресных):

```
foo1.bar.baz.   60   IN   A   192.168.1.1
foo2.bar.baz.   60   IN   A   192.168.1.2
foo3.bar.baz.   60   IN   A   192.168.1.3
foo.bar.baz.    60   IN   CNAME   foo1.bar.baz.
foo.bar.baz.    60   IN   CNAME   foo2.bar.baz.
foo.bar.baz.    60   IN   CNAME   foo3.bar.baz.
```

Вероятно, читателям это покажется странным, ведь мы постоянно твердим, что не следует использовать CNAME-записи не по назначению. DNS-серверы BIND 4 не считали такие данные ошибочными (а они таковыми являются) и просто возвращали CNAME-записи для *foo.bar.baz* в порядке перестановки *round robin*.

С другой стороны, DNS-серверы BIND 8 более бдительны и немедленно сообщают об ошибке. Тем не менее, их можно явным образом настроить, разрешив использование множественных CNAME-записей для одного доменного имени с помощью следующей конструкции:

```
options {
    multiple-cnames yes;
};
```

DNS-серверы BIND 9 не замечают этой CNAME-ошибки вплоть до версии 9.1.0. BIND 9.1.0 способен обнаружить ошибку, но не поддерживает предписание *multiple-cnames*.

Предписание rrset-order

В определенных ситуациях предпочтительнее, чтобы DNS-сервер не использовал механизм round robin. К примеру, существует необходимость сделать один веб-сервер резервным для второго. В таком случае DNS-

сервер должен всегда возвращать адрес резервного веб-сервера после адреса основного. Но в случае перестановки адресов это невозможно, порядок адресов в различных ответах будет постоянно меняться.

DNS-серверы, начиная с BIND 8.2 (но не BIND 9 на момент существования версии 9.1.0), позволяют отключать действие механизма round robin для отдельных доменных имен и типов записей. Так, если необходимо обеспечить стабильный порядок возвращаемых адресных записей для *www.movie.edu*, можно воспользоваться предписанием *rrset-order*:

```
options {
    rrset-order {
        class IN type A name "www.movie.edu" order fixed;
    };
};
```

В этом случае, вероятно, следует понизить значение TTL для адресных записей *www.movie.edu*, чтобы DNS-сервер, кэшировавший эти записи, не слишком долго возвращал их в различном порядке.

Параметры *class*, *type* и *name* определяют записи, для которых используется указанный порядок. Класс по умолчанию принимает значение IN, тип – значение ANY, а имя – *, другими словами речь идет о произвольных записях. Поэтому оператор:

```
options {
    rrset-order {
        order random;
    };
};
```

предписывает использовать случайный порядок для всех записей, возвращаемых DNS-сервером. Параметр имени может содержать маску вместо самой первой метки:

```
options {
    rrset-order {
        type A name "*.movie.edu" order cyclic;
    };
};
```

Допускается использование только одного предписания *rrset-order*, но оно может содержать несколько спецификаций порядка. Имеет силу первая спецификация для наборов записей в ответах.

rrset-order позволяет выбрать один из трех (сосчитайте-ка, из трех!) вариантов порядка:

fixed

Результаты поиска записей всегда возвращаются в одном и том же порядке.

random

Результаты поиска записей возвращаются в случайном порядке.

cyclic

Результаты поиска записей возвращаются в циклическом порядке (*round robin*).

Поведение по умолчанию определяется следующим образом:

```
options {
    rrset-order {
        class IN type ANY name "*" order cyclic;
    };
};
```

К сожалению, настройка с помощью *rrset-order* не является окончательным решением, поскольку ее работе могут мешать клиенты и кэширование DNS-серверов. Более правильным и удачным решением является использование SRV-записей, которые мы изучим в главе 16.

Сортировка адресов DNS-сервером

Иногда администратора не устраивает ни порядок перестановки round robin, ни какой-либо иной. При необходимости связаться с узлом, имеющим многочисленные сетевые интерфейсы, выбор определенного адреса на основе адреса вашего узла может привести к повышению производительности. Но добиться этого с помощью предписания *rrset-order* невозможно.

Если узел, смотрящий в несколько сетей, является локальным, и входит в сеть или подсеть исходного узла, один из его адресов является «более близким». Если узел удаленный, при выборе различных интерфейсов производительность также будет меняться, но часто нет разницы, какой именно интерфейс использовать. В давние времена сеть 10 (бывшая основа ARPAnet) всегда была ближе любого другого удаленного адреса. С тех пор Интернет значительно усовершенствовался, поэтому выбор той или иной сети при контакте с внешними узлами не приводит к особенному повышению производительности, хотя мы все равно рассмотрим этот случай.

Прежде чем мы начнем говорить о сортировке адресов DNS-сервером, читателям следует взглянуть на главу 6, а именно на раздел «Инструкция sortlist», и подумать – возможно, сортировка адресов с помощью клиента более соответствует вашим нуждам. Поскольку клиент и DNS-сервер могут находиться в разных сетях, зачастую имеет больший смысл выполнять сортировку адресов с помощью клиента конкретного узла – способом, для этого узла оптимальным. Сортировка адресов DNS-сервером работает достаточно хорошо, но ее бывает трудно оптимизировать для всех обслуживаемых клиентов. Сортировка ад-

ресов клиентом появилась в BIND 4.9, поэтому если используется клиент (не DNS-сервер) версии более ранней, чем 4.9, либо не клиент BIND, выбирать особо не приходится. Придется обойтись сортировкой адресов DNS-сервером, которая появилась в версии 4.8.3.

В результате невероятного поворота событий механизм сортировки адресов *не был* включен в более поздние версии BIND, в основном потому, что разработчики утверждали «Этому механизму не место в DNS-сервере». Статус-кво был восстановлен – даже с некоторыми улучшениями – в BIND 8.2. BIND 9.1.0 – первая версия BIND 9, поддерживающая сортировку адресов.

Сортировка адресов в BIND 4

Сортировка адресов BIND 4 проще в настройке, чем аналогичный механизм BIND 8, но сложнее для описания, поскольку довольно многое происходит автоматически, безо всяких настроек. Оставим BIND 8 на потом.

Локальные мультиинтерфейсные узлы

Рассмотрим сначала локальные мультиинтерфейсные узлы. Предположим, у нас есть узел исходных текстов, на котором хранятся их мастер-копии; этот узел входит в две сети, которые носят весьма оригинальные названия – сеть A и сеть B. На узле применяется NFS для экспорта файловых систем для узлов в обеих сетях. Узлы сети A получают бо́льшую производительность, если используют сетевой интерфейс этого узла с сетью A. Аналогично узлы сети B получают бо́льшую производительность, используя интерфейс этого узла с сетью B – при поиске адреса NFS-сервера.

В главе 4 «Установка BIND» мы упоминали, что BIND возвращает все адреса для мультиинтерфейсного узла. Для возвращаемых адресов не гарантирован какой-либо заданный порядок, поэтому нам пришлось создать псевдонимы (*wh249.movie.edu* и *wh253.movie.edu* для *wormhole.movie.edu*) отдельных интерфейсов. Если какой-либо из интерфейсов являлся более предпочтительным, мы (вернее, клиент DNS) могли выбрать соответствующий псевдоним и получить нужный адрес. Можно использовать псевдонимы для выбора «более близких» интерфейсов (скажем, для работы с NFS), но это не всегда имеет смысл, поскольку существует сортировка адресов.

DNS-серверы BIND 4 по умолчанию производят сортировку адресов в случае выполнения следующего условия: узел, являющийся автором запроса к DNS-серверу, находится в одной сети с узлом DNS-сервера (скажем, оба узла принадлежат сети A). Как BIND определяет, что находится в одной сети с клиентом? При запуске BIND находит все адреса интерфейсов узла, на котором работает. Из этих адресов BIND извлекает номера сетей и использует их для создания списка сортировки

по умолчанию. При получении запроса BIND проверяет, не входит ли адрес отправителя в одну из сетей, перечисленных в списке поиска по умолчанию. Если это так, запрос является локальным и BIND производит сортировку адресов в ответе.

Предположим (рис. 10.3), что на узле *notorious* работает DNS-сервер под управлением BIND 4. Список поиска этого DNS-сервера по умолчанию содержит сеть A и сеть B. Когда узел *spellbound* посылает DNS-серверу запрос для адресов *notorious*, то получает ответ, в котором адрес *notorious* в сети A представлен первым. Это происходит потому, что *notorious* и *spellbound* принадлежат одной сети, A. Когда узел *charade* посылает DNS-серверу запрос для адресов *notorious*, то получает ответ, в котором первым представлен адрес узла *notorious* в сети B, поскольку в данном случае оба узла принадлежат сети B. В обоих случаях DNS-сервер производит сортировку адресов в запросе, потому что узел разделяет сеть с узлом DNS-сервера. В сортированном списке адресов «более близкий» интерфейс приводится первым.

Теперь немного изменим условия. Допустим, DNS-сервер работает на узле *gaslight*. Когда *spellbound* посылает DNS-серверу *gaslight* запрос адреса *notorious*, то получает тот же ответ, что в предыдущем случае, поскольку *spellbound* и *gaslight* находятся в одной сети, A, что и приводит к сортировке адресов DNS-сервером. Однако узел *charade* может получить ответ с другим порядком адресов, поскольку не входит в од-

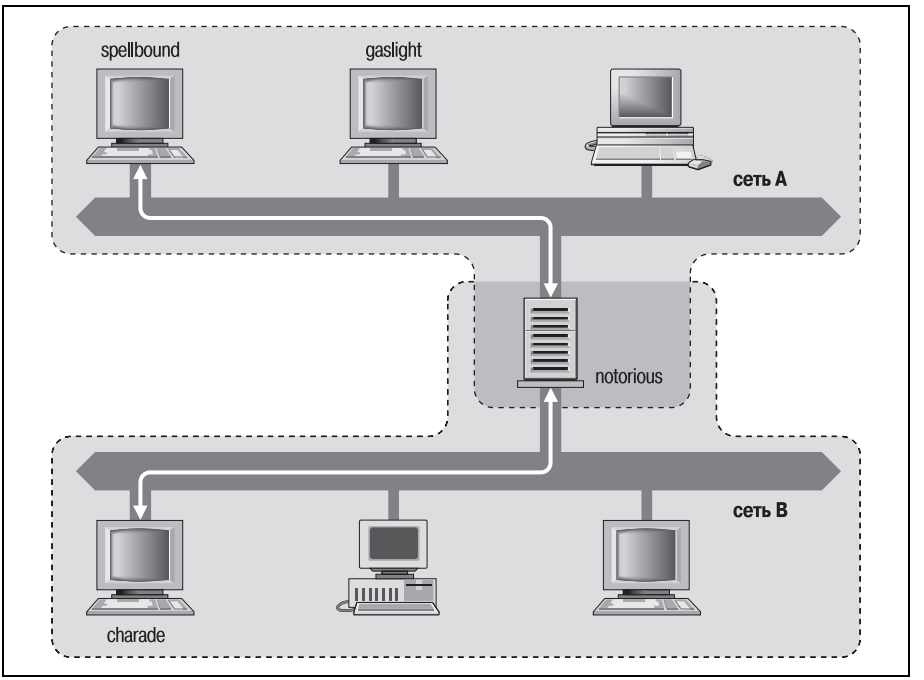

Рис. 10.3. *Связь с локальным мультиинтерфейсным узлом*

ну сеть с узлом *gaslight*. Более близкий адрес *notorious* при этом может оказаться первым в ответе для *charade*, но только по случайности, а не из-за сортировки адресов DNS-сервером. В данном случае, чтобы воспользоваться преимуществами сортировки адресов имен BIND 4 на узле *charade*, должен существовать вторичный DNS-сервер в сети B.

Как можно видеть, в существовании DNS-сервера для каждой отдельной сети есть определенные преимущества: DNS-сервер доступен, если маршрутизатор не работает, да при этом еще и выполняет сортировку адресов мультиинтерфейсных узлов. Поскольку сортировка адресов выполняется DNS-сервером, отпадает необходимость указывать псевдонимы при монтировании NFS или для прочих сетевых соединений, чтобы получить лучшую скорость работы.

Удаленные мультиинтерфейсные узлы

Предположим, наша площадка часто вступает в контакт с определенной удаленной площадкой либо «далекой» локальной площадкой, а лучшая производительность получается при выборе тех или иных адресов удаленной сети. К примеру, зона *movie.edu* охватывает сети 192.249.249/24 и 192.253.253/24. Добавим подключение к сети 10/8 (старая ARPAnet). Удаленный узел, с которым происходит общение, имеет два сетевых подключения – в сети 10/8 и в сети 26/8. Узел не производит маршрутизацию в сеть 26/8, но по какой-то особой причине связан с ней. Поскольку маршрутизатор для 26/8 постоянно перегружен, более высокая производительность получается при использовании адреса сети 10/8 для удаленного узла (рис. 10.4).

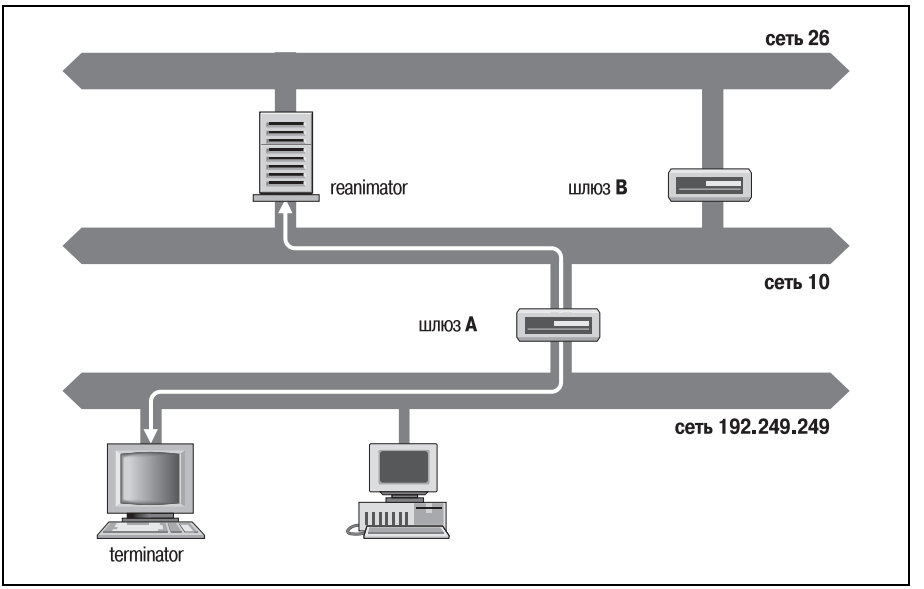

Рис. 10.4. *Связь с удаленным мультиинтерфейсным узлом*

Если пользователь узла *terminator.movie.edu* устанавливает связь с узлом *reanimator.movie.edu*, предпочтительно воспользоваться адресом в сети 10/8, поскольку доступ через шлюз B к адресу 26/8 будет более медленным, чем при использовании прямого маршрута. К сожалению, DNS-сервер *terminator.movie.edu* не станет специально возвращать адрес 10/8 первым в списке адресов *reanimator.movie.edu*; дело в том, что *terminator.movie.edu* входит только в сеть 192.249.249/24, соответственно ничего не знает о «расстояниях» до адресов 10/8 и 26/8. На сцене появляется инструкция *sortlist*. Предпочтительность адресов сети 10/8 определяется следующей строкой файла *named.boot*:

```
sortlist 10.0.0.0
```

Аргументы *sortlist добавляются* к списку поиска по умолчанию. При применении приведенной инструкции *sortlist* список поиска сервера *terminator.movie.edu* содержит сети с номерами 192.249.249/24 и 10/8. Теперь, когда пользователь узла *terminator.movie.edu* делает запрос к DNS-серверу *terminator.movie.edu*, а DNS-сервер производит сортировку запроса, поскольку запрос локальный, первыми в ответе будут следовать адреса сети 192.249.249/24. Если в ответе нет адресов сети 192.249.249/24, первыми будут адреса сети 10/8. Это решает описанную проблему; теперь при поиске адреса *reanimator.movie.edu* первым в ответе будет следовать адрес узла в сети 10/8.

Сортировка адресов для подсетей

Подсети не сильно влияют на сортировку адресов. Когда DNS-сервер создает список поиска по умолчанию, в него добавляется номер подсети, а также номер сети этой подсети. Как и прежде, если запрос является локальным, DNS-сервер производит сортировку адресов, и адрес в общей подсети приводится в ответе первым. К сожалению, не все здесь идеально: нет возможности добавлять в список *sortlist* элементы для прочих подсетей сети. И вот почему. DNS-сервер предполагает, что элементы списка *sortlist* представляют номера сетей (а не подсетей), а номер сети уже входит в список сортировки. Поскольку номер сети уже входит в список сортировки, явно определенные элементы *sortlist*, соответствующие той же сети, просто удаляются.

Элементы sortlist

И последнее – чтобы добавить несколько элементов в список *sortlist*, следует перечислить их все в одной строке, следующим образом:

```
sortlist 10.0.0.0 26.0.0.0
```

Сортировка адресов в BIND 8 и 9

DNS-серверы BIND 8.2 и более поздних версий (а также 9.1.0 и более поздних версий) также умеют производить сортировку адресов. Однако особой автоматизации процесса не наблюдается, а настройка меха-

низма не особенно проста. Соответствующее предписание оператора *options*, само собой, носит имя *sortlist*.

Предписание *sortlist* в качестве аргумента принимает список отбора адресов. Однако *sortlist* особым образом интерпретирует списки. Каждый элемент списка отбора адресов считается списком, который содержит один, либо два элемента.

Если список содержит один элемент, он используется для проверки IP-адреса клиента. Если адрес клиента соответствует элементу, происходит сортировка адресов в ответе клиенту, причем адреса, соответствующие элементу, возвращаются первыми. Запутались? Вот пример:

```
options {
    sortlist {
        { 192.249.249/24; };
    };
};
```

Единственная запись в списке сортировки состоит из одного элемента. Этот список сортировки производит сортировку адресов сети 192.249.249/24 для запросов, поступивших также из этой сети.

Если запись содержит два элемента, первый элемент используется для проверки IP-адреса клиента. Если адрес клиента соответствует элементу, DNS-сервер производит сортировку адресов в ответе этому клиенту, причем адреса, соответствующие второму элементу, возвращаются первыми. Второй элемент на деле может являться списком отбора адресов, состоящим из нескольких значений, и в этом случае первым в ответ добавляется тот адрес, который соответствует первому значению из такого списка. Простой пример:

```
options {
    sortlist {
        { 192.249.249/24; { 192.249.249/24; 192.253.253/24; }; };
    };
};
```

Этот список сортировки применяется для клиентов с адресами 192.249.249/24 и возвращает адреса, отдавая предпочтение адресам из той же сети и из сети 192.253.253/24.

Элементы в списке поиска могут определять как подсети, так и отдельные узлы:

```
options {
    sortlist {
        { 15.1.200/21;        // если клиент - из подсети 15.1.200/21
            { 15.1.200/21;    // отдавать предпочтение адресам этой подсети
            15/8; };          // или из сети 15/8
        };
    };
};
```

DNS-серверы: предпочтения

Механизм топологии в BIND 8 несколько схож с механизмом *sortlist*, но используется в процессе выбора DNS-серверов. (Топология не поддерживается в BIND 9 на момент существования версии 9.1.0.) Ранее мы описывали процесс выбора из серверов, являющихся авторитативными для одной зоны, при котором выбирается DNS-сервер с минимальным временем передачи сигнала (RTT). Но мы слукавили – совсем чуть-чуть. В действительности BIND 8 помещает удаленные DNS-серверы в интервалы длительностью 64 миллисекунды при сравнении показателей RTT. Первый интервал – на самом деле – имеет ширину всего в 32 миллисекунды (ну вот! опять мы обманули читателей!), от нулевой до 32 миллисекунды. Следующий интервал охватывает миллисекунды с 33 по 96, и т. д. Построение интервалов гарантирует, что DNS-серверы, расположенные на различных континентах, попадают в разные интервалы.

Идея в том, чтобы отдавать предпочтение серверам, попадающим в интервалы, ближе расположенные к точке отсчета, но при этом считать серверы в пределах одного интервала эквивалентными. Если при сравнении RTT-показателей двух удаленных DNS-серверов один из них попадает в более ранний интервал, серверу, связанному с этим показателем, отдается предпочтение. Но если удаленные DNS-серверы попадают в один интервал, локальный сервер проводит расследование на предмет выяснения, какой из этих серверов топологически ближе.

Итак, топология позволяет некоторым образом влиять на процесс выбора DNS-сервера, которому посылается запрос. Топология позволяет отдавать предпочтение DNS-серверам определенной сети. В качестве аргумента принимается список отбора адресов, элементы которого определяют сети и перечислены в порядке предпочтения для локального DNS-сервера. Поэтому конструкция:

```
topology {
    15/8;
    172.88/16;
};
```

предписывает локальному DNS-серверу отдавать предпочтение серверам сети 15/8, а затем серверам сети 172.88/16. Если DNS-сервер выбирает между DNS-сервером сети 15/8, DNS-сервером сети 172.88/16 и DNS-сервером сети 192.168.1/24, в ситуации, когда значения RTT для всех трех попадают в один интервал, будет сделан выбор в пользу DNS-сервера сети 15/8.

Можно указывать отрицание для элементов списка отбора адресов топологии, штрафуя таким образом DNS-серверы в определенных сетях. Чем раньше в списке поиска будет найдено соответствие с отрицанием, тем выше штраф. Можно использовать подобный механизм, чтобы

удержать DNS-сервер от посылки запросов удаленным DNS-серверам, расположенным, к примеру, в сетях, подверженных частым сбоям.

Нерекурсивный DNS-сервер

По умолчанию клиенты BIND посылают рекурсивные запросы, а DNS-серверы по умолчанию выполняют работу, которая требуется, чтобы ответить на эти запросы. (Если вы уже подзабыли, как работает рекурсия, загляните в главу 2 «Как работает DNS».) В процессе поиска ответов на рекурсивные запросы DNS-сервер создает кэш неавторитативной информации из других зон.

В некоторых случаях нежелательно, чтобы DNS-серверы выполняли лишнюю работу, связанную с выполнением рекурсивных запросов наполнением кэша данными. Такова, например, ситуация с корневыми DNS-серверами. Корневые DNS-серверы настолько загружены, что не могут позволить себе тратить силы на разрешение рекурсивных запросов. Вместо этого они возвращают ответы, исходя исключительно из данных, которые лежат в пределах их авторитативности. Ответное сообщение может содержать готовый ответ, но чаще всего содержит ссылки на другие DNS-серверы. И, поскольку корневые DNS-серверы не поддерживают рекурсивные запросы, на них не происходит заполнение кэша неавторитативными данными, что очень правильно, поскольку в ином случае кэши разрастались бы до гигантских размеров.[1]

С помощью следующего оператора файла настройки можно предписать DNS-серверу BIND работать в нерекурсивном режиме:

```
options {
    recursion no;
};
```

Для случая DNS-сервера BIND 4.9 следует использовать инструкцию:

```
options no-recursion
```

Теперь сервер будет отвечать на рекурсивные запросы так, как если бы они были нерекурсивными.

Чтобы предотвратить заполнение кэша, совместно с предписанием *recursion no* следует использовать еще одну дополнительную настройку:

```
options {
```

[1] Заметим, что корневой сервер имен в обычной ситуации не получает рекурсивных запросов, если только администратор определенного сервера имен не настроил его на использование корневого сервера в качестве ретранслятора. Варианты: администратор настроил анализатор на использование корневого сервера имен, пользователь указал корневой сервер имен в качестве рабочего для *nslookup*. Такие вещи случаются гораздо чаще, чем можно подумать.

```
    fetch-glue no;
};
```

Либо, для BIND 4.9:

```
options no-fetch-glue
```

Данная конструкция запрещает DNS-серверу производить поиск связующих записей при создании дополнительного раздела ответа. DNS-серверы BIND 9 не производят поиск связующих записей, поэтому предписание *fetch-glue* в BIND 9 не используется.

Не следует упоминать нерекурсивный DNS-сервер в файле *resolv.conf*. DNS-сервер можно сделать нерекурсивным, но не существует настройки, которая позволяла бы клиенту работать с таким сервером.[1] Если DNS-сервер должен продолжать обслуживать один или несколько клиентов, можно воспользоваться предписанием *allow-recursion*, которое появилось в BIND 8.2.1 (и существует в BIND 9). *allow-recursion* в качестве аргумента принимает список отбора адресов; запросы с этих адресов могут быть рекурсивными, но все прочие запросы обрабатываются так, как будто рекурсия выключена:

```
options {
     allow-recursion { 192.253.254/24; };  // Посылка рекурсивных запросов
                                           // разрешена только клиентам подсети FX
};
```

По умолчанию *allow-recursion* разрешает рекурсию для всех IP-адресов.

Кроме того, не следует упоминать нерекурсивный DNS-сервер в качестве ретранслятора. Когда DNS-сервер использует другой сервер в качестве ретранслятора, то передает ретранслятору *рекурсивные* запросы. Чтобы разрешить авторизованным DNS-серверам передавать рекурсивные запросы нерекурсивному ретранслятору, воспользуйтесь предписанием *allow-recursion*.

Допускается упоминание нерекурсивного DNS-сервера в качестве одного из серверов, авторитативных для зоны (то есть, можно разрешить DNS-серверу родительской зоны перенаправлять клиенты к этому DNS-серверу в целях получения данных зоны). Это работает, потому что DNS-серверы между собой обмениваются нерекурсивными запросами.

Борьба с фальшивыми DNS-серверами

Администратор зоны в процессе работы может обнаружить удаленный DNS-сервер, возвращающий неправильную информацию – устаревшую, несоответствующую действительности, некорректно форматиро-

[1] Это верно в общем случае. Разумеется, программы, посылающие нерекурсивные запросы, либо программы, которые могут быть настроены таким образом, скажем, *nslookup*, по-прежнему будут работать.

ванную либо преднамеренно ложную. Можно попытаться связаться с администратором этого сервера и решить проблему. А можно поберечь свои нервы и настроить собственный DNS-сервер таким образом, чтобы он не посылал запросы фальшивому; это можно сделать в DNS-серверах BIND 4.9, BIND 8, а также BIND 9, начиная с версии 9.1.0. Вот так выглядит оператор настройки:

```
server 10.0.0.2 {
    bogus yes;
};
```

Или инструкция для сервера BIND 4.9:

```
bogusns 10.0.0.2
```

Разумеется, следует указывать IP-адрес фальшивого сервера.

Если администратор запретил DNS-серверу общаться с другим сервером, который является единственным для своей зоны, можно забыть о поиске информации для имен этой зоны. Можно надеяться, что существуют другие DNS-серверы для этой зоны, и они возвращают корректную информацию.

Наиболее эффективный способ перекрыть взаимодействие с удаленным DNS-сервером – поместить его в список *blackhole*. DNS-сервер не посылает запросы DNS-серверам из этого списка и не отвечает на запросы, получаемые от этих серверов.[1] *blackhole* – это предписание оператора *options*, аргументом которого является список отбора адресов:

```
options {

    /* Не стоит тратить время, отвечая на запросы с внутренних
       адресов (RFC 1918) */

    blackhole {
        10/8;
        172.16/12;
        192.168/16;
    };
};
```

Эта конструкция запретит DNS-серверу отвечать на запросы, получаемые с внутренних адресов (см. документ RFC 1918). В сети Интернет отсутствуют маршруты к этим адресам, и попытка отвечать – пустая трата процессорного времени и излишняя загрузка канала.

Предписание *blackhole* поддерживается версиями BIND 8 начиная с 8.2, а также версиями BIND 9 после 9.1.0.

---

[1] И мы действительно имеем в виду *отсутствие ответа*. Для запросов, запрещенных списком *allow-query*, будет сгенерирован ответ, сообщающий, что в выполнении запроса отказано. Запросы, запрещенные списком *blackhole*, не получат никакого ответа. Вообще.

Настройка системы

Для большинства DNS-серверов стандартные настройки BIND вполне подходят для работы, но иногда необходимо внести дополнительные коррективы. В этом разделе мы обсудим разнообразные ручки и переключатели, позволяющие настраивать DNS-сервер.

Передача зон

Передача зон может быть связана с большой нагрузкой на DNS-сервер. В DNS-серверах BIND 4 для исходящих передач зон (в случае, когда сервер является основным для зоны) требуется *fork( )* процесса *named*, а значит – значительные объемы памяти. В BIND 4.9 появились способы ограничивать нагрузку, связанную с передачей зон основными серверами. В BIND 8 и 9 помимо этих механизмов существуют дополнительные возможности.

Ограничение числа запросов для отдельных DNS–серверов

Начиная с BIND 4.9 существует возможность ограничить число передач зон, одновременно запрашиваемых DNS-сервером с конкретного удаленного сервера. Это счастье для администратора удаленного сервера, поскольку в случае изменения всех зон его сервер не будет завален запросами на передачу – а это немаловажно, если речь идет о сотнях зон.

В BIND 8 и 9 соответствующий оператор настройки выглядит так:

```
options {
    transfers-per-ns 2;
};
```

Эквивалентная инструкция загрузочного файла BIND 4:

```
limit transfers-per-ns 2
```

В BIND 9 предел может быть определен либо частным образом для отдельных DNS-серверов, либо глобально. Частное определение производится с помощью предписания *transfers* оператора *server*:

```
server 192.168.1.2 {
    transfers 2;
};
```

Частное предписание имеет более высокий приоритет, чем глобальное, которое присутствует в операторе *options*. По умолчанию предел установлен в два активных процесса передачи зоны на DNS-сервер. Может показаться, что ограничение слишком суровое, но такая настройка работает. Происходит следующее. Предположим, DNS-серверу нужно

произвести загрузку четырех зон с удаленного сервера. DNS-сервер начинает получение первых двух зон, оставляя третью и четвертую в очереди. После завершения одного из первых двух процессов, DNS-сервер начинает получение третьей зоны. После завершения еще одного процесса, DNS-сервер инициирует процесс получения четвертой зоны. Конечный результат такой же, как и до появления ограничений, – все зоны получены; но при этом процесс длился чуть дольше.

Когда может возникнуть необходимость увеличить это ограничение? Если вы заметили, синхронизация с удаленным DNS-сервером занимает слишком много времени, и очевидно, причина именно в последовательном выполнении запросов, а не в скорости сетевого обмена между узлами. Вероятно, этот вопрос имеет значение, если DNS-сервер работает с сотнями или тысячами зон. Помимо прочего, следует убедиться, что удаленный DNS-сервер и сетевой маршрут между узлами способны выдержать нагрузку, создаваемую многочисленными параллельными процессами передачи зональных данных.

Ограничение общего числа запросов на передачу зон

Последнее ограничение было связано с получением зон с удаленного DNS-сервера. Описываемое в данном разделе позволяет установить ограничение для нескольких удаленных DNS-серверов. Начиная с версии 4.9 BIND позволяет ограничивать общее число параллельно запрашиваемых DNS-сервером зон. По умолчанию установлен предел в 10 зон. Как мы только что говорили, DNS-сервер по умолчанию параллельно может получать только две зоны от каждого из удаленных DNS-серверов. Если DNS-сервер производит получение пар зон с пяти удаленных DNS-серверов, значит он достиг предела и отложит все прочие запросы, пока не будет завершен один из текущих процессов.

Оператор настройки для BIND 8 и 9:

```
options {
    transfers-in 10;
};
```

Эквивалентная инструкция загрузочного файла BIND 4:

```
limit transfers-in 10
```

Если узел или сеть не справляется с десятью параллельными процессами передачи зон, следует уменьшить это число. Когда же речь идет о сервере, обслуживающем сотни и тысячи зон, возможно, имеет смысл увеличить предел, если узел и сеть способны справиться с такой нагрузкой. При увеличении этого предела возможно понадобится увеличить и число разрешенных параллельных процессов на каждый DNS-сервер. (К примеру, если DNS-сервер производит получение данных от всего лишь четырех удаленных серверов, то по умолчанию сможет параллельно производить получение лишь восьми зон. Увеличение пре-

дела общего числа процессов в таком случае не имеет смысла, если не ослабляются ограничения для отдельных DNS-серверов.)

Ограничение общего числа исходящих передач зон

В DNS-серверах BIND 9 существует возможность ограничивать число параллельных процессов для передачи зон другим серверам. Это едва ли не более полезно, чем ограничение числа запрашиваемых передач, поскольку последнее заставляет полагаться на доброту администраторов вторичных DNS-серверов и на то, что они не будут перегружать наш основной сервер. Оператор BIND 9:

```
options {
    transfers-out 10;
};
```

По умолчанию принимается значение 10.

Ограничение длительности передачи зоны

DNS-серверы BIND 8 и 9 позволяют ограничить продолжительность процесса получения зоны. По умолчанию устанавливается порог в 120 минут (два часа). Смысл заключается в том, что процесс передачи зоны, который длится более двух часов, вероятнее всего, просто «повис» и уже никогда не завершится, потребляя, между тем, драгоценные ресурсы. Уменьшить или увеличить этот временной интервал (скажем, если DNS-сервер является вторичным для зоны, получение копии которой обычно занимает больше двух часов) можно с помощью оператора следующего вида:

```
options {
    max-transfer-time-in 180;
};
```

Ограничить время получения для отдельной зоны можно с помощью предписания *max-transfer-time-in* в операторе *zone*. К примеру, если известно, что получение зоны *rinkydink.com* всегда занимает много времени (допустим, три часа) из-за размеров зоны либо из-за медленного канала связи с основным сервером, но для всех прочих зон нужно установить более низкий порог (в районе часа), можно использовать такие операторы:

```
options {
    max-transfer-time-in 60;
};

zone "rinkydink.com" {
    type slave;
    file "bak.rinkydink.com";
    masters { 192.168.1.2; };
    max-transfer-time-in 180;
};
```

В BIND 9 существует предписание *max-transfer-time-out*, которое может использоваться таким же образом (в операторе *options* или *zone*). Оно позволяет определить максимальную длительность исходящей передачи зоны (то есть передачи данных вторичному DNS-серверу); по умолчанию принимается такое же, как и для *max-transfer-time-in*, пороговое значение – 120 минут.

BIND 9 позволяет также определять максимально допустимую длину интервала времени простоя при передаче зоны. Два предписания настройки, *max-transfer-idle-in* и *max-transfer-idle-out*, позволяют определять этот параметр для входящих и исходящих передач зон, соответственно. Оба предписания могут быть использованы в операторах *options* и *zone*. По умолчанию принимается пороговое значение в 60 минут.

Ограничение частоты передачи зон

Интервал обновления для зоны может быть установлен в столь малое значение, что вторичные DNS-серверы зоны будут выполнять ненужную работу. К примеру, если DNS-сервер является вторичным для многих тысяч зон, а администраторы некоторых из этих зон устанавливают значения интервалов обновления в очень низкие значения, DNS-сервер может просто не успевать выполнять синхронизацию для всех зон. (Если речь идет о DNS-сервере, который является вторичным для действительно большого числа зон, обязательно прочтите раздел «Ограничение числа SOA-запросов»; может появиться необходимость в ограничении числа выполняемых SOA-запросов.) С другой стороны, неопытный администратор может установить слишком большой интервал обновления, который будет приводить к рассинхронизации первичного DNS-мастер-сервера зоны и вторичных DNS-серверов на длительные промежутки времени.

Начиная с версии 9.1.0, в BIND появилась возможность ограничивать интервалы обновления с помощью предписаний *max-refresh-time* и *min-refresh-time*. Эти предписания определяют множество допустимых значений для всех основных и вторичных зон и зон-заглушек при использовании в операторе *options* либо для конкретной зоны – при использовании в операторе *zone*. В качестве аргументов выступает количество секунд:

```
options {
    max-refresh-time 86400;  // обновления не реже раза в сутки
    min-refresh-time 1800;   // не чаще раза в 30 минут
};
```

Помимо этого, начиная с версии 9.1.0, DNS-серверы позволяют ограничивать допустимые значения для интервала повторения с помощью предписаний *max-retry-time* и *min-retry-time*, которые имеют тот же синтаксис.

Повышение эффективности при передаче зон

Передача зоны, как мы уже говорили, состоит из многочисленных сообщений, которыми стороны обмениваются через TCP-соединение. При традиционной передаче зоны каждое сообщение DNS содержит только одну запись. Это очень неэффективно: каждое сообщение должно содержать полный заголовок, даже если оно инкапсулирует единственную запись. Это все равно что возить одного человека в пассажирском автобусе. DNS-сообщение, передаваемое через TCP-соединение, может содержать гораздо больше записей, поскольку его размер ограничен порогом в 64 килобайта!

DNS-серверы BIND 8 и 9 понимают новый формат передачи данных зоны, который носит название *many-answers*. В формате *many-answers* в одно сообщение DNS помещается максимальное число записей. В результате передача зоны в формате *many-answers* намного меньше загружает канал, за счет сокращения издержек транспортировки, и процессор, за счет сокращения времени, уходящего на разбор сообщений DNS.

Формат, используемый DNS-сервером для передачи зон, для которых он является первичным мастером, может быть определен с помощью предписания *transfer-format*. То есть это предписание определяет формат, используемый для передачи зоны вторичным DNS-серверам. *transfer-format* может являться предписанием оператора *options* или *server*; в качестве предписания *options transfer-format* определяет глобально используемый формат передачи зон. В BIND 8 по умолчанию используется старый формат передачи зон, *one-answer*, который обеспечивает совместимость с DNS-серверами BIND 4. В BIND 9 по умолчанию используется формат *many-answers*. Оператор:

```
options {
    transfer-format many-answers;
};
```

предписывает DNS-серверу использовать для передачи зон всем вторичным DNS-серверам формат *many-answers*, за исключением случаев, когда оператор *server*, подобный приводимому ниже, предписывает обратное:

```
server 192.168.1.2 {
    transfer-format one-answer;
};
```

У формата *many-answers* есть один недостаток – передача зоны может на деле занимать больше времени, чем при применении старого формата, несмотря на повышение эффективности загрузки канала и использования процессора.

Чтобы воспользоваться преимуществами нового способа передачи зональных данных, воспользуйтесь одним из следующих пунктов:

- Установите глобальный формат передачи зон в *many-answers* (либо не делайте этого, если используется BIND 9), если большинство вторичных DNS-серверов работает под управлением BIND 8, BIND 9 или сервера Microsoft DNS, который также реализует поддержку нового формата.
- Установите глобальный формат передачи зон в *one-answer*, если большинство вторичных DNS-серверов работает под управлением BIND 4. Затем воспользуйтесь предписанием *transfer-format* оператора *server*, чтобы частным образом произвести настройку для серверов, которые представляют собой исключения.

Помните, что при использовании BIND 9 потребуется добавить явный оператор *server* в файлы настройки всех вторичных DNS-серверов BIND 4 с целью выбора для них формата *one-answer*.

Ограничение ресурсов

Иногда просто требуется сказать DNS-серверу, чтобы он не жадничал: использовал меньше памяти, открывал меньше файлов. Возможности определения подобных ограничений появились в BIND 4.9, а в BIND 8 и 9 – как и для многих других новых возможностей – существуют вариации на тему таких настроек.

Изменение ограничения размера сегмента данных

Некоторые операционные системы ограничивают объем памяти, доступный отдельному процессу. Если операционная система не позволяет DNS-серверу получить дополнительную память, это может привести к останову или завершению работы DNS-сервера. Предел доступной для использования памяти может быть достигнут, если сервер работает с очень большими объемами данных, либо если установлено низкое ограничение. Для таких случаев в BIND 4.9, BIND 8 и BIND 9, начиная с версии 9.1.0, предусмотрено несколько параметров, которые позволяют изменять умолчание системного ограничения для размера сегмента данных. С помощью этих настроек можно установить для *named* большее пороговое значение, чем предлагается системой.

В BIND 8 и 9 оператор выглядит так:

```
options {
    datasize size
};
```

Эквивалентная инструкция BIND 4:

```
limit datasize size
```

size (размер) – целочисленное значение, по умолчанию в байтах. Можно указывать альтернативные единицы измерения, добавляя к размеру соответствующую букву: k – килобайты, m – мегабайты, g – гигабайты. К примеру, «64m» – это 64 мегабайта.

Не во всех операционных системах реализована поддержка увеличения размера сегментов данных для отдельных процессов. В случае отсутствия такой поддержки DNS-сервер записывает сообщение *syslog* с приоритетом LOG_WARNING, чтобы уведомить администратора о проблеме.

Изменение ограничения размера стека

DNS-серверы BIND 8 и BIND 9, начиная с версии 9.1.0, позволяют изменять не только ограничение размера сегмента данных, но и производить подстройку объема памяти, выделяемого операционной системой под стек процесса *named*. Синтаксис оператора следующий:

```
options {
    stacksize size;
};
```

Размер *size* имеет тот же формат, что и аргумент предписания *datasize*. Как и *datasize*, *stacksize* работает только в системах, которые позволяют процессам изменять размер стека.

Изменение ограничения размера файла образа

Если вам не нравится, что *named* оставляет за собой гигантские файлы образов (core files), можно сократить их размер с помощью предписания *coresize*. И наоборот, если процесс *named* не смог создать полный файл образа из-за слишком жестких ограничений, накладываемых операционной системой, предел можно увеличить с помощью все того же предписания. Синтаксис *coresize*:

```
options {
    coresize size;
};
```

Как и в случае *datasize*, данное ограничение работает только в операционных системах, которые позволяют процессам изменять размеры файлов образов, а также не реализовано в BIND 9 до версии 9.1.0.

Изменение ограничения для числа открытых файлов

Если DNS-сервер является авторитативным для большого числа зон, процесс *named* при запуске открывает большое число файлов – по одному на авторитативную зону; предполагается, что существуют резервные копии файлов зон, для которых DNS-сервер является вторичным. Аналогично, если на узле DNS-сервера существует большое число виртуальных сетевых интерфейсов[1], *named* требует наличия одного

[1] В главе 14 «Разрешение проблем DNS и BIND» описаны более разумные решения для проблемы слишком большого числа открытых файлов.

файлового дескриптора на каждый интерфейс. В большинстве операционных систем Unix существуют ограничения для числа файлов, которые единовременно могут быть открыты одним процессом. Если DNS-сервер попытается открыть больше файлов, чем разрешено, в выводе *syslog* появится следующее сообщение:

```
named[pid]: socket(SOCK_RAW): Too many open files
```

Если операционная система позволяет изменять это ограничение для отдельных процессов, можно увеличить соответствующее значение с помощью предписания *files*:

```
options {
    files number;
};
```

По умолчанию используется (вполне допустимое) значение *unlimited* (без ограничений), хотя оно просто означает, что DNS-сервер не ограничивает число одновременно открытых файлов, хотя это число может ограничивать операционная система. Мы знаем, что читателей уже тошнит от этой фразы, но – в BIND 9 это предписание не поддерживается до версии 9.1.0.

Ограничение числа клиентов

BIND 9 предоставляет администратору возможность ограничить число клиентов, параллельно обслуживаемых DNS-сервером. Число рекурсивных клиентов (это клиенты и DNS-серверы, одновременно использующие данный сервер в качестве ретранслятора) можно ограничить с помощью предписания *recursive-clients*:

```
options {
    recursive-clients 10;
};
```

По умолчанию принимается значение 1000. Если окажется, что DNS-сервер отказывается выполнять рекурсивные запросы и заносит в log-файл ошибок сообщения следующего рода:

```
Sep 22 02:26:11 terminator named[13979]: client 192.249.249.151#1677: no more
recursive clients: quota reached
```

можно увеличить пороговое значение. И наоборот, если DNS-сервер с трудом успевает выполнять все получаемые рекурсивные запросы, следует это значение уменьшить.

Помимо этого, существует возможность ограничить число параллельно существующих TCP-соединений (для передачи зон и TCP-запросов) с помощью предписания *tcp-clients*. TCP-соединения потребляют гораздо больше ресурсов, чем UDP-соединения, поскольку узел должен отслеживать состояние каждого TCP-соединения. Значение по умолчанию – 100.

Ограничение числа SOA-запросов

Начиная с BIND 8.2.2, DNS-серверы позволяют ограничивать число ожидающих обработки SOA-запросов. Если сервер является вторичным для многих тысяч зон, в каждый отдельный момент времени в очереди запросов может находиться большое число запросов SOA-записей. Отслеживание каждого такого запроса требует небольшого, но фиксированного количества памяти, которая имеет обыкновение кончаться, поэтому в DNS-серверах BIND 8 число таких запросов ограничено до четырех. Если DNS-сервер не успевает в таком темпе выполнять свои обязанности вторичного, предел можно увеличить с помощью предписания *serial-queries*:

```
options {
    serial-queries 1000;
};
```

Предписание *serial-queries* вышло из употребления в BIND 9. BIND 9 ограничивает скорость посылки SOA-запросов (не более 20 в секунду), но не число ожидающих обработки.

Интервалы служебных операций

DNS-серверы BIND традиционно выполняли периодические хозяйственные работы, такие как обновление зон, для которых являются вторичными. В BIND 8 и 9 появилась возможность управления частотой выполнения таких работ.

Интервал чистки

DNS-серверы BIND до версии 4.9 удаляют старые записи из кэша в пассивном режиме. Прежде чем вернуть запись клиенту, DNS-сервер проверяет, не истекло ли время жизни этой записи. Если значение TTL обнулилось, DNS-сервер начинает процесс разрешения, чтобы найти более актуальные данные. Это означает, что DNS-сервер BIND 4 может кэшировать множество записей в суматохе разрешения имен, после чего эти записи будут спокойно портиться в кэше, занимая драгоценную память даже в случае устаревания.

DNS-серверы BIND 8 и 9 производят активное исследование кэша и удаление устаревших записей каждый интервал чистки. Это означает, что DNS-серверы BIND 8 и 9 обычно занимают меньше памяти под кэшируемые данные, чем аналогичный DNS-сервер BIND 4. С другой стороны, процесс чистки приводит к потреблению процессорного времени, так что на медленных или сильно загруженных DNS-серверах не очень эффективно производить чистку каждый час.

Интервал чистки по умолчанию устанавливается в 60 минут. Изменить его можно с помощью предписания *cleaning-interval* оператора *options*. Следующая конструкция:

```
options {
    cleaning-interval 120;
};
```

устанавливает интервал чистки в 120 минут. Чтобы полностью отключить чистку кэша, следует воспользоваться нулевым значением для интервала.

Интервал сканирования интерфейсов

Мы уже говорили, что по умолчанию BIND производит прослушивание всех сетевых интерфейсов узла. DNS-серверы BIND 8 и 9 достаточно интеллектуальны, чтобы заметить, когда один из сетевых интерфейсов узла перестает работать или вновь включается. С этой целью они периодически сканируют сетевые интерфейсы узла. Сканирование производится один раз за интервал сканирования, который по умолчанию длится 60 минут. Если известно, что на узле отсутствуют динамические сетевые интерфейсы, сканирование новых интерфейсов можно отключить, чтобы избежать ненужных издержек – установив нулевое значение интервала сканирования интерфейсов:

```
options {
    interface-interval 0;
};
```

С другой стороны, если узел производит настройку или отключение сетевых интерфейсов чаще, чем раз в час, может понадобиться сократить этот интервал.

Интервал создания статистических отчетов

Говорим сразу – изменение интервала создания статистических отчетов, то есть частоты записи статистики в файл для DNS-сервера BIND 8, практически не повлияет на производительность. Но данный раздел, посвященный периодически выполняемым служебным операциям, гораздо больше подходит для описания этого параметра, чем какой-либо другой раздел книги.

Синтаксис предписания *statistics-interval* идентичен синтаксису предписаний для прочих служебных интервалов:

```
options {
    statistics-interval 60;
};
```

По умолчанию интервал длится те же 60 минут, а установка нулевого значения отключает периодическое создание статистических отчетов. Поскольку BIND 9 не производит записи статистики в log-файл *syslog*, в нем отсутствует и изменяемый интервал создания статистических отчетов.

Значения TTL

Какое-то время BIND производил усечение значений TTL для кэшированных записей – до разумных значений. В BIND 8 и 9 это разумное значение поддается настройке.

В BIND 8.2 и DNS-серверах более поздних версий TTL для кэшированных отрицательных ответов можно ограничить с помощью предписания *max-ncache-ttl* оператора *options*. Эта возможность является своего рода страховкой для тех, кто произвел обновление до версии 8.2, в которой применяется новая схема отрицательного кэширования (документ RFC 2308 и все прочее, подробности приводятся в главе 4). Начиная с этой версии, отрицательная информация кэшируется DNS-сервером в соответствии со значением последнего поля SOA-записи зоны, а многие зональные администраторы до сих пор используют это поле в качестве значения TTL по умолчанию для зоны – оно, скорее всего, великовато для отрицательной информации. Поэтому предусмотрительные администраторы DNS-серверов могут воспользоваться вот такой конструкцией:

```
options {
    max-ncache-ttl 3600;  // 3600 секунд - это один час
};
```

чтобы сократить все более высокие значения отрицательного TTL до одного часа. По умолчанию принимается значение в 10800 секунд (три часа). Такая мера предосторожности гарантирует, что при поиске свежей информации клиент не получит отрицательный ответ (что могло бы произойти в случае отставания синхронизации вторичных DNS-серверов), а DNS-сервер не будет держать этот отрицательный ответ в памяти непозволительно долгое время, служа причиной ошибок разрешения.

DNS-серверы BIND 9 позволяют также изменять верхний предел допустимого значения TTL для кэшируемых записей – с помощью предписания *max-cache-ttl*. Время по умолчанию – одна неделя. DNS-серверы BIND 8 также ограничивают максимальное время жизни одной неделей, но не позволяют изменять этот предел.

Наконец, существует нечто, называемое *некорректное TTL*, которое, вообще говоря, вовсе не TTL. Это длительность интервала времени, в течение которого DNS-сервер помнит, что удаленный DNS-сервер не является авторитативным для зоны, которая ему делегирована. Это позволяет сэкономить драгоценное время, избегая запросов к DNS-серверу, который все равно ничего не знает об интересующих нас доменных именах. DNS-серверы BIND 8, начиная с версии 8.2, а также BIND 9 более поздние, чем 9.1.0, позволяют изменять *некорректное TTL* с помощью предписания *lame-ttl* оператора *options*. По умолчанию это значение составляет 600 секунд (10 минут), а предельное значение – 30 минут. Кэширование информации о DNS-серверах с некор-

ректным делегированием можно отключить, установив нулевое значения, хотя нам это кажется Исключительно Дурным Тоном.

Совместимость

Подводя итоги, рассмотрим некоторые предписания настройки, связанные с совместимостью DNS-сервера с клиентами и другими DNS-серверами.

Предписание *rfc2308-type1* позволяет управлять форматом отрицательных ответов, посылаемых DNS-сервером. По умолчанию DNS-серверы BIND 8 и 9 включают в отрицательный ответ только SOA-запись зоны. Второй допустимый вариант формата включает также и NS-записи зоны, но некоторые из старых версий DNS-серверов неправильно интерпретируют такие ответы – считая их перенаправлениями. Если, по какой-то странной причине (странной, потому что нам не удается придумать хотя бы одну) появилась необходимость включать в отрицательные ответы NS-записи, воспользуйтесь такой конструкцией:

```
options {
    rfc2308-type1 yes;
};
```

Впервые поддержка предписания *rfc2308-type1* появилась в BIND 8.2; в BIND 9 она отсутствует.

При посылке кэшированных отрицательных ответов более старым DNS-серверам могут возникать определенные проблемы. Во времена, когда отрицательное кэширование не существовало, все отрицательные ответы, разумеется, были авторитативными. Но некоторые разработчики DNS-серверов добавляли в код проверку: приниматься должны только авторитативные отрицательные ответы. Затем появилось отрицательное кэширование и неавторитативные отрицательные ответы. Ой!

Предписание *auth-nxdomain* оператора *options* позволяет устанавливать бит авторитативности для ответов, извлекаемых DNS-сервером из кэша, чтобы любой из древних DNS-серверов остался доволен. По умолчанию в серверах BIND 8 *auth-nxdomain* устанавливается в значение *on* (режим включен); в BIND 9 режим по умолчанию выключен.

И наконец, когда безрассудно смелые люди портировали BIND 8.2.2 на систему Windows NT, они обнаружили, что DNS-сервер должен реагировать на символы возврата каретки и начала новой строки, встречающиеся в концах строк (последовательность, определяющая конец строки в системах Windows) так же, как на просто символ новой строки (конец строки в системах Unix). Чтобы использовать такой режим, воспользуйтесь конструкцией:

```
options {
    treat-cr-as-space yes;
};
```

В BIND 9 предписание игнорируется, поскольку эта версия DNS-сервера считает оба варианта признаком конца строки.

Основы адресации в IPv6

Прежде чем рассмотреть следующие две темы, а именно отображение доменных имен в IPv6-адреса и обратное, мы опишем представление и структуру адресов IPv6. Как, вероятно, известно читателям, адреса IPv6 имеют длину 128 битов. Предпочтительное представление IPv6-адреса – восемь групп четырехзначных шестнадцатеричных цифр, разделяемых двоеточиями. Пример:

```
0123:4567:89ab:cdef:0123:4567:89ab:cdef
```

Первая группа шестнадцатеричных цифр (в нашем примере – 0123) представляет четыре наиболее значимых (старших) бита адреса.

Группы цифр, начинающиеся с одного или нескольких нулей, можно записывать сокращенно, поэтому предыдущий пример может выглядеть так:

```
123:4567:89ab:cdef:123:4567:89ab:cdef
```

Но каждая группа должна содержать по меньшей мере одну цифру; в противном случае следует использовать запись ::. Запись через :: позволяет производить сжатие последовательных групп нулей. Это удобно, если необходимо указать только IPv6-префикс. Например:

```
dead:beef::
```

определяет первые 32 бита IPv6-адреса в виде *dead:beef*, а остальные 96 – в виде групп нулей.

Пара символов :: может использоваться в начале IPv6-адреса для указания суффикса. К примеру, адрес loopback-интерфейса в IPv6 обычно записывается как:

```
::1
```

или как 127 нулей, за которыми следует единица. Символы :: могут использоваться и внутри адреса, в качестве сокращения для непрерывных групп нулей:

```
dead:beef::1
```

Сокращение :: может использоваться в пределах адреса лишь единожды, во избежание неоднозначностей.

Префиксы IPv6 записываются в формате, сходном с CIDR-записью в IPv4. Значимые биты префикса записываются в стандартном формате IPv6 и отделяются от десятеричного показателя числа значимых битов символом слэша. Таким образом, три приводимых ниже специфика-

ции префиксов абсолютно эквивалентны (хотя, очевидно, и не одинаково лаконичны):

```
dead:beef:0000:00f1:0000:0000:0000:0000/64
dead:beef::00f1:0:0:0:0/64
dead:beef:0:f1::/64
```

Адреса IP версии 4 имеют иерархическую структуру, которая отражает природу IPv4-сетей: отдельные сети подключены через провайдеров услуг Интернета, а те, в свою очередь, подключены к другим провайдерам, либо к несущим магистралям сети Интернет. Каждый провайдер отвечает за несколько битов результирующего 32-битного IP-адреса: провайдеры, расположенные ближе к несущим магистралям, отвечают за более старшие биты, а администраторы сетей – за оставшиеся.

Спецификация IPv6 разрабатывалась с прицелом на гораздо более крупную сеть Интернет, поэтому уровней иерархии в IPv6-адресах еще больше. Каждый уровень соответствует одному из сетевых уровней интернет-сети на основе IPv6-адресации.

В сердце интернет-сети IPv6 – агрегаторы высшего уровня (Top-Level Aggregators, TLAs). Это сети, которые подключены напрямую к несущим информационным магистралям сети и обеспечивают подключение для агрегаторов следующего уровня (Next-Level Aggregators, NLAs). NLA-сети обеспечивают подключение сетей отдельных площадок к интернет-сети IPv6 (рис. 10.5).

Как и в случае с IPv4-сетями, каждая из организаций заведует определенными битами IPv6-адреса. Чтобы помочь читателям представить

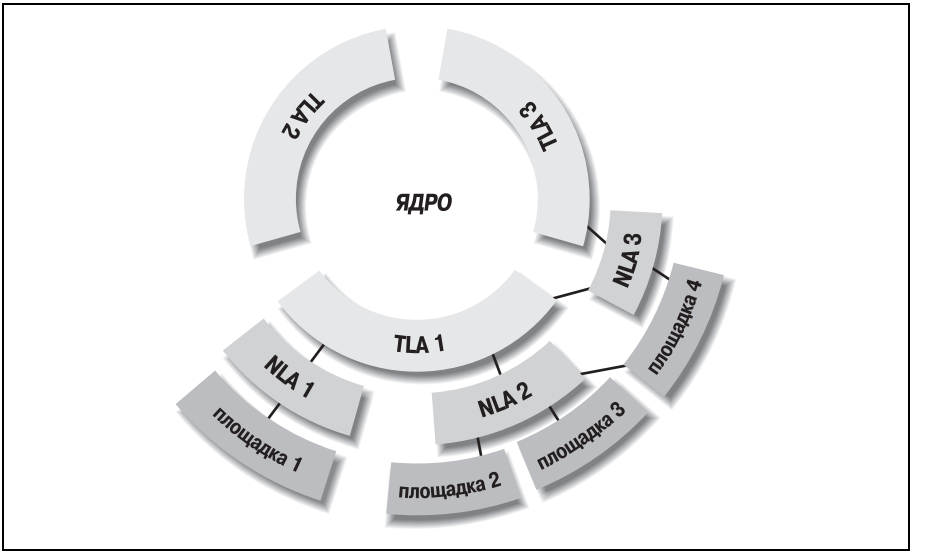

Рис. 10.5. *Структура интернет-сети IPv6*

иерархию адресации IPv6, мы приводим диаграмму для наиболее распространенной структуры IPv6-адреса, описанной в документе RFC 2374:

```
+--+-----+---+--------+--------+-------------------------------+
| 3|  13 | 8 |   24   |   16   |          64 bits              |
+--+-----+---+--------+--------+-------------------------------+
|FP| TLA |RES|  NLA   |  SLA   |         Interface ID          |
|  | ID  |   |  ID    |  ID    |                               |
+--+-----+---+--------+--------+-------------------------------+
```

FP – это префикс формата (*Format Prefix*), который занимает первые три бита адреса и определяет формат оставшейся части. Форматный префикс для данного конкретного формата, который называется (вдохните поглубже) *IPv6 Aggregatable Global Unicast Address Format* (IPv6-формат для индивидуальных (юникастовых) адресов), имеет значение 001. Следующие 13 битов определяют агрегатор высшего уровня. Восемь зарезервированных битов имеют нулевые значения, и еще 24 бита определяют агрегатор следующего уровня. Оставшиеся биты используются конкретной площадкой: *Site-Level Aggregator ID* (идентификатор агрегатора уровня площадки) или *SLA ID* – по сути дела эквивалентен битам подсети в IPv4-адресе, а *Interface ID* (идентификатор интерфейса) уникальным образом определяет отдельный интерфейс сети площадки.

В описанном формате адреса каждый идентификатор присваивается сущностью, которая расположена уровнем выше в иерархии. Так, единственный регистратор адресов высшего уровня присваивает TLA-идентификаторы агрегаторам высшего уровня. TLA, в свою очередь присваивают NLA-идентификаторы своим клиентам, а те – SLA-идентификаторы своим. NLA-идентификаторы должны быть уникальными только в пределах родительского TLA-идентификатора, как и SLA-идентификаторы в пределах отдельных NLA-идентификаторов.

Адреса и порты

DNS-серверы BIND 9 умеют использовать в качестве транспорта протоколы IPv4 и IPv6; то есть могут посылать и получать запросы и ответы через IPv4 и IPv6. DNS-серверы BIND 8 поддерживают в качестве транспорта только IPv4. Однако обе версии сервера поддерживают похожие предписания, предназначенные для выбора сетевых интерфейсов и портов, для которых ведется прослушивание.

Настройка для транспорта IPv4

С помощью предписания *listen-on* можно определить прослушиваемые сетевые интерфейсы для DNS-сервера BIND 8 или BIND 9. В простейшей форме *listen-on* принимает в качестве аргумента список отбора адресов:

```
options {
    listen-on { 192.249.249/24; };
};
```

DNS-сервер производит прослушивание любых сетевых интерфейсов локального узла, адреса которых входят в список отбора адресов. Чтобы указать альтернативный порт (с номером, отличным от 53) прослушивания, можно воспользоваться модификатором *port*:

```
options {
    listen-on port 5353 { 192.249.249/24; };
};
```

В BIND 9 существует возможность определять порт для каждого отдельного сетевого интерфейса:

```
options {
    listen-on { 192.249.249.1 port 5353; 192.253.253.1 port 1053; };
};
```

Обратите внимание, что большинство клиентов не может быть настроено на отправку запросов DNS-серверу через альтернативный порт, поэтому такой DNS-сервер будет полезен гораздо меньше, чем может показаться. Но он может выполнять передачу зон, поскольку в предписании *masters* может быть указан альтернативный порт:

```
zone "movie.edu" {
    type slave;
    masters port 5353 { 192.249.249.1; };
    file "bak.movie.edu";
};
```

Если у DNS-сервера BIND 9 несколько основных DNS-серверов, каждый из которых производит прослушивание собственного порта, можно воспользоваться такой конструкцией:

```
zone "movie.edu" {
    type slave;
    masters { 192.249.249.1 port 5353; 192.253.253.1 port 1053; };
    file "bak.movie.edu";
};
```

BIND 9 даже позволяет посылать NOTIFY-сообщения на альтернативные порты. Чтобы предписать DNS-серверу уведомлять вторичные DNS-серверы через один и тот же нестандартный порт, можно воспользоваться предписанием:

```
also-notify port 5353 { 192.249.249.9; 192.253.253.9; };  // два адреса узла zardoz
```

Чтобы посылать уведомления через различные порты:

```
also-notify { 192.249.249.9 port 5353; 192.249.249.1 port 1053; };
```

Если DNS-сервер должен работать с определенным локальным сетевым интерфейсом для отправки запросов, – например, потому, что

один из основных DNS-серверов опознает лишь один из его многочисленных адресов, – воспользуйтесь предписанием *query-source*:

```
options {
    query-source address 192.249.249.1;
};
```

Обратите внимание, что в качестве аргумента фигурирует не список отбора адресов, а единственный IP-адрес. Можно также указать конкретный исходный порт для запросов:

```
options {
    query-source address 192.249.249.1 port 53;
};
```

Поведение BIND по умолчанию таково: используется произвольный сетевой интерфейс, через который может быть произведена доставка для конечного адреса, и случайным образом выбранный порт, не требующий суперпользовательских привелегий. То есть:

```
options {
    query-source address * port *;
};
```

Обратите внимание, что *query-source* относится только к UDP-запросам; для TCP-запросов выбор исходного адреса всегда происходит на основе таблицы маршрутизации; исходный порт, при этом, также выбирается случайным образом.

Существует аналогичное предписание *transfer-source*, позволяющее определять исходный адрес, используемый для передачи зон. В BIND 9 эта настройка относится также к SOA-запросам, поступающим от вторичных DNS-серверов, а также к ретранслированным динамическим обновлениям:

```
options {
    transfer-source 192.249.249.1;
};
```

Как и в случае *query-source*, в качестве аргумента фигурирует единственный IP-адрес, но отсутствует ключевое слово *address*. В BIND 8 модификатор *port* отсутствует. В BIND 9 можно указать и исходный порт:

```
options {
    transfer-source 192.249.249.1 port 1053;
};
```

Это определение исходного порта действует только для UDP-сообщений (то есть SOA-запросов и ретранслированных динамических обновлений).

transfer-source может также использоваться в качестве предписания оператора *zone*, и в этом случае относится только к случаям передачи

(BIND 9 – еще и к SOA-запросам и динамическим обновлениям) этой зоны:

```
zone "movie.edu" {
    type slave;
    masters { 192.249.249.3; };
    file "bak.movie.edu";
    transfer-source 192.249.249.1;  // всегда использовать IP-адрес из той же
                                    // сети для передачи зоны movie.edu
};
```

И наконец, в BIND 9.1.0 существует предписание, позволяющее определять, с какого адреса посылаются NOTIFY-сообщения, а именно – *notify-source*. Такая возможность находит применение на узлах, состоящих в нескольких сетях одновременно, поскольку вторичные DNS-серверы принимают NOTIFY-сообщения для зоны только с IP-адресов, перечисленных в предписании *masters* для этой зоны. Синтаксис *notify-source* схож с синтаксисом прочих *source*-предписаний. Пример:

```
options {
    notify-source 192.249.249.1;
};
```

Как и в случае *transfer-source*, в *notify-source* может быть определен исходный порт, а само предписание может использоваться в операторе *zone* с целью частной настройки параметров для конкретной зоны:

```
zone {
    type slave;
    masters { 192.249.249.3; };
    file "bak.movie.edu";
    notify-source 192.249.249.1 port 5353;
};
```

Настройка для транспорта IPv6

По умолчанию DNS-сервер BIND 9 не производит прием IPv6-запросов. Чтобы настроить DNS-сервер на прослушивание локальных сетевых IPv6-интерфейсов, следует воспользоваться предписанием *listen-on-v6*:

```
options {
    listen-on-v6 { any; };
};
```

В отличие от своего IPv4-собрата, предписание *listen-on-v6* в качестве аргумента принимает только ключевые слова *any* и *none*. Тем не менее, можно настроить DNS-сервер BIND 9 на прослушивание альтернативного порта – либо нескольких альтернативных портов – с помощью модификатора *port*:

```
options {
    listen-on-v6 port 1053 { any; };
};
```

По умолчанию номер порта, разумеется, 53.

Также существует возможность определять используемый в исходящих запросах IPv6-адрес, а также порт с помощью предписания *transfer-source-v6*:

```
options {
    transfer-source-v6 222:10:2521:1:210:4bff:fe10:d24;
};
```

или:

```
options {
    transfer-source-v6 port 53 222:10:2521:1:210:4bff:fe10:d24;
};
```

По умолчанию используется исходный адрес, соответствующий сети, в которую посылается ответ, и случайный порт с минимальными привилегиями. Как и в случае *transfer-source*, предписание *transfer-source-v6* может использоваться в операторе *zone*. Определение исходного порта влияет только на SOA-запросы и ретранслируемые динамические обновления.

И наконец, BIND 9.1.0 и более поздних версий позволяет определить, какой IPv6-адрес будет использоваться для отправки NOTIFY-сообщений а-ля предписание *notify-source*. Соответствующее предписание IPv6 называется, само собой, *notify-source-v6*:

```
options {
    notify-source-v6 222:10:2521:1:210:4bff:fe10:d24;
};
```

Как и в случае *transfer-source-v6*, может быть указан исходный порт, а само предписание может использоваться в операторе *zone*.

IPv6: прямое и обратное отображение

Понятно, что существующие A-записи не справятся со 128-битными IPv6-адресами; BIND ожидает, что в части данных A-записи присутствует 32-битный адрес в восьмибитной нотации.

Организация IETF разработала простое решение этой проблемы, описанное в документе RFC 1886. 128-битные IPv6-адреса хранятся в новой адресной записи – AAAA, а для целей обратного отображения создан новый домен *ip6.int*. Это простое решение было реализовано в BIND 4.9. Однако простое решение понравилось далеко не всем, поэтому было придумано гораздо более сложное. Это решение, о котором мы скоро расскажем, связано с новыми типами записей – A6 и DNAME, а также требует полной ревизии кода DNS-сервера для реализации.

Использование старой записи AAAA и домена *ip6.int* в настоящее время не поощряется, но существует достаточное число IPv6-программ,

которые все еще используют именно это, а не более новое решение, так что важно понимать суть обоих вариантов.

AAAA и ip6.int

Итак, простой способ решить задачу описан в документе RFC 1886 и связан с использованием адресной записи, в четыре раза более длинной, чем A-запись. Речь идет о записи AAAA (четыре A). Данные в AAAA-записи представляются в формате IPv6-адреса, описанном ранее. Поэтому можно встретить AAAA-записи примерно следующего вида:

```
ipv6-host    IN    AAAA    4321:0:1:2:3:4:567:89ab
```

RFC 1886 также определяет *ip6.int*, новое пространство имен для обратного отображения IPv6-адресов. Каждый уровень поддоменов в пределах *ip6.int* представляет четыре бита 128-битного адреса в шестнадцатеричной системе счисления – в том же формате, что и компоненты адреса в AAAA-записи. Наименее важные (младшие) биты начинают доменное имя. В отличие от формата адресов, принятого для AAAA-записей, формат доменного имени не позволяет опускать нули из квартетов, поэтому в доменном имени *ip6.int*, соответствующем полному IPv6-адресу, всегда присутствует 32 шестнадцатеричных цифры и 32 уровня поддоменов. Вот доменное имя, которое соответствует адресу из последнего примера:

```
b.a.9.8.7.6.5.0.4.0.0.0.3.0.0.0.2.0.0.0.1.0.0.0.0.0.0.0.1.2.3.4.ip6.int.
```

Для этих доменных имен существуют PTR-записи, как и в случае доменных имен зоны *in-addr.arpa*:

```
b.a.9.8.7.6.5.0.4.0.0.0.3.0.0.0.2.0.0.0.1.0.0.0.0.0.0.0.1.2.3.4.ip6.int. IN
PTR  mash.ip6.movie.edu.
```

A6, DNAME-записи, бит-строковые метки и ip6.arpa

Мы описали простой способ. Более сложный – и принятый в качестве официального – способ реализации прямого и обратного отображения IPv6 связан с использованием записей типов A6 и DNAME. Записи A6 и DNAME описаны в документах RFC 2874 и RFC 2672, соответственно. Впервые поддержка записей этих типов появилась в BIND версии 9.0.0.

AAAA-записи и обратное отображение в *ip6.int* были заменены новыми механизмами, прежде всего, по той причине, что затрудняли перенумерацию сетей. К примеру, если организация меняет агрегаторы следующего уровня, ей придется изменить все AAAA-записи в файлах данных зон, поскольку 24 бита каждого IPv6-адреса идентифицируют NLA-агрегатор.[1] Или представьте смену агрегатора высшего уровня

[1] И разумеется, новый NLA-агрегатор может быть подключен через другой TLA-агрегатор, а это означает изменение еще 16 битов...

для NLA-агрегатора. Зональные данные клиентов в этот момент потеряли бы всякий смысл.

Записи A6 и прямое отображение

В целях упрощения перенумерации в записях A6 разрешается указывать лишь часть IPv6-адреса, к примеру, последние 64 бита (идентификатор интерфейса), связанные с сетевым интерфейсом узла, а оставшуюся часть адреса определять строковым доменным именем. Таким образом, администраторы могут указывать только часть адреса, которую непосредственно контролируют. Чтобы создать полный адрес, клиент или DNS-сервер должен разобрать цепочку записей A6 начиная с доменного имени узла и заканчивая идентификатором агрегатора высшего уровня. Цепь может иметь ветвления, если сеть площадки подключена к нескольким NLA-агрегаторам или NLA-агрегатор подключен к нескольким TLA-агрегаторам.

К примеру, следующая запись A6:

```
$ORIGIN movie.edu.
drunkenmaster  IN   A6   64  ::0210:4bff:fe10:0d24  subnet1.v6.movie.edu.
```

определяет последние 64 бита IPv6-адреса узла *drunkenmaster.movie.edu* (число 64 в данном случае определяет число битов префикса, *не* приводимых в данной A6-записи) и говорит, что оставшиеся 64 бита могут быть найдены с помощью поиска A6-записи в *subnet1.v6.movie.edu*.

subnet1.v6.movie.edu, в свою очередь, определяет 16 последних битов 64-битного префикса (SLA-идентификатора), который был опущен в адресе A6 для *drunkenmaster.movie.edu*, в также доменное имя для поиска следующей записи A6:

```
$ORIGIN v6.movie.edu.
subnet1  IN  A6  48  0:0:0:1::  movie-u.nla-a.net.
subnet1  IN  A6  48  0:0:0:1::  movie.nlab.net.
```

Первые 48 битов префикса данных для *subnet1.v6.movie.edu* установлены в нули, поскольку они не значимы в данном контексте.

По сути дела, эти записи предписывают нам произвести поиск *двух* записей A6 на следующем шаге, одну для *movie-u.nla-a.net*, а одну для *movie.nlab.net*. Это происходит потому, что Университет кинематографии подключен к двум NLA-агрегаторам, NLA A и NLA B. В зоне NLA A мы находим следующее:

```
$ORIGIN nla-a.net.
movie-u  IN  A6   40  0:0:21::  nla-a.tla-1.net.
```

восемь битов идентификатора площадки в пределах идентификатора NLA, установленные NLA-агрегатором A для сети Университета кинематографии. Дело в том, что поле NLA-идентификатора тоже имеет

иерархическое деление и состоит из идентификатора NLA, присвоенного TLA-агрегатором, *и* собственного идентификатора нашей сети. Поскольку NLA-агрегатор присваивает только идентификатор нашей площадки, а оставшаяся часть лежит на совести соответствующего TLA-агрегатора, мы и ожидаем увидеть только идентификатор площадки в зональных данных NLA-агрегатора. Оставшаяся часть NLA-идентификатора содержится в записи A6 соответствующей TLA-зоны.

В зоне NLA B мы обнаруживаем вот эту запись, которая также содержит идентификатор площадки для нашей сети:

```
$ORIGIN nlab.net.
movie  IN  A6  40  0:0:42::  nlab.tla-2.net.
```

В зонах TLA присутствуют такие данные:

```
$ORIGIN tla-1.net.
nla-a  IN  A6  16  0:10:2500::  tla-1.top-level-v6.net.
```

и:

```
$ORIGIN tla-2.net.
nlab   IN  A6  16  0:19:6600::  tla-2.top-level-v6.net.
```

И наконец, в зоне регистрации IPv6-адресов высшего уровня мы находим следующие записи, которые отражают TLA-идентификаторы для TLA 1 и TLA 2:

```
$ORIGIN top-level-v6.net.
tla-1  IN  A6  0  222::
tla-2  IN  A6  0  242::
```

Следуя по цепи записей A6, DNS-сервер может собрать все 128 битов двух IPv6 адресов *drunkenmaster.movie.edu*. Результаты:

```
222:10:2521:1:210:4bff:fe10:d24
242:19:6642:1:210:4bff:fe10:d24
```

Для первого адреса используется маршрут через TLA 1 и NLA A в сеть Университета, а для второго – маршрут через TLA 2 и NLA B. (Мы подключены к двум NLA-агрегаторам для надежности.) Обратите внимание, что если TLA 1 изменит NLA-значение для NLA A, потребуется изменить только запись A6 для *nla-a.tla-1.net* в данных соответствующей зоны; это изменение будет «каскадировано» во все цепи A6, проходящие через NLA A. Таким образом, управление адресацией в IPv6-сетях становится очень удобным, как и смена NLA-агрегаторов.

Если DNS-сервер встречается в NS-записи и владеет одной или несколькими записями A6, эти A6-записи должны содержать полные 128-битные IPv6-адреса. Это позволяет избежать тупиковой ситуации, когда DNS-клиент или DNS-сервер должен связаться с удаленным DNS-сервером в целях разрешения части IPv6-адреса этого DNS-сервера.

DNAME-записи и обратное отображение

Изучив, как работает прямое отображение для записей A6, перейдем к обратному отображению для IPv6-адресов. Как и в случае с записями A6, процесс гораздо сложнее, чем при использовании зоны *ip6.int*.

Обратное отображение IPv6-адресов связано с использованием записей типа DNAME, который описан в документе RFC 2672, а также бит-строковых меток, описанных в документе RFC 2673. DNAME-записи немного похожи на CNAME-записи с масками. Они используются для замены одного доменного суффикса другим. К примеру, если мы раньше использовали доменное имя *movieu.edu*, а затем сменили его на *movie.edu*, то старую зону *movieu.edu* можно заменить такой:

```
$TTL 1d
@    IN  SOA    terminator.movie.edu.  root.movie.edu. (
     2000102300
     3h
     30m
     30d
     1h  )

     IN   NS     terminator.movie.edu.
     IN   NS     wormhole.movie.edu.

     IN   MX     10 postmanrings2x.movie.edu.

     IN   DNAME  movie.edu.
```

Данная DNAME-запись в зоне *movieu.edu* сопоставляется с любым доменным именем, которое кончается на *movieu.edu*, кроме собственно имени *movieu.edu*. DNAME-записи, в отличие от CNAME-записей, могут сосуществовать с другими записями для того же доменного имени, если это не CNAME- или DNAME-записи. При этом владелец DNAME-записи *не может* включать поддомены.

Когда DNS-сервер *movieu.edu* получает запрос для любого доменного имени, которое кончается на *movieu.edu*, допустим, *cuckoosnest.movieu.edu*, запись DNAME предписывает «синтезировать» псевдоним, связывающий *cuckoosnest.movieu.edu* и *cuckoosnest.movie.edu*, заменив *movieu.edu* на *movie.edu*:

```
cuckoosnest.movieu.edu.  IN  CNAME  cuckoosnest.movie.edu.
```

Действие DNAME-записи отчасти схоже с действием команды *s* (substitute, замена) редактора *sed*. DNS-сервер *movieu.edu* возвращает данную CNAME-запись. Если запрос поступил от более нового сервера, DNAME-запись также возвращается в ответном сообщении, так что впоследствии получатель может самостоятельно синтезировать CNAME-записи на основе кэшированной DNAME-записи.

Второй ингредиент волшебства обратного отображения для IPv6 – *бит-строковые метки*, которые являются способом более компактной записи длинных последовательностей двоичных (однобитовых) меток

доменного имени. Предположим, необходимо разрешить делегирование между двумя произвольными битами IP-адреса. Это может заставить нас представлять каждый бит адреса в виде доменного имени. Но для доменного имени, представляющего IPv6-адрес, потребуется 128 меток! Ух!

Бит-строковые метки позволяют производить слияние битов в соседних метках с целью получения более короткой шестнадцатеричной, восьмеричной, двоичной строки или строки в восьмибитной нотации. Строка помещается в лексемы “\[” и “]”, чтобы ее можно было отличать от традиционных меток, и начинается с буквы, которая определяет основание строки: *b* для двоичного основания, *o* для восьмеричного и *x* для шестнадцатеричного.

Вот бит-строковые метки, соответствующие двум IPv6-адресам *drunkenmaster.movie.edu*:

```
\[x022200102521000102104bfffe100d24]
\[x024200196642000102104bfffe100d24]
```

Обратите внимание, что строка начинается со старшего бита, как и в текстовом представлении адреса IPv6, но метки следуют в порядке, обратном принятому в домене *in-addr.arpa*. Несмотря на это, две приведенные бит-строковые метки представляют в несколько иной кодировке доменные имена, которые начинаются следующим образом:

```
0.0.1.0.0.1.0.0.1.0.1.1.0.0.0.0.0.0.0.0.1.0.0.0.0.1.1.1.1.1.1.1...
```

Также отметим, что присутствуют все 32 шестнадцатеричные цифры адреса – нули нельзя удалять, поскольку отсутствуют двоеточия, разделяющие группы цифр.

Бит-строковые метки могут также использоваться для представления фрагментов IPv6-адресов, и в этом случае следует указывать число значащих битов строки, отделяя его от строки символом слэша. Идентификатор для TLA 1 – *\[x0222/16]*.

DNAME-записи и бит-строковые идентификаторы используются для сопоставления отдельных частей доменных имен, кодирующих IPv6-адреса, и итеративного изменения доменного имени от имени, для которого производится поиск, до имени из зоны, которая управляется организацией узла с указанным IPv6-адресом.

Представим себе, что мы производим обратное отображение для *\[x024200196642000102104bfffe100d24].ip6.arpa*, доменного имени, которое соответствует сетевому интерфейсу *drunkenmaster.movie.edu* (при доступе через TLA 2 и NLA B). Корневые DNS-серверы, вероятно, отправят наш DNS-сервер к DNS-серверам зоны *ip6.arpa*, содержащей следующие записи:

```
$ORIGIN ip6.arpa.
\[x0222/16]    IN     DNAME      ip6.tla-1.net.
\[x0242/16]    IN     DNAME      ip6.tla-2.net.
```

Первая из них соответствует началу доменного имени, для которого производится поиск, поэтому DNS-серверы *ip6.arpa* возвращают нашему серверу псевдоним, который выглядит следующим образом:

```
\[x0242001966420001021 04bfffe100d24].ip6.arpa.  IN  CNAME
\[x00196642000102104bfffe100d24].ip6.tla-2.net.
```

Первые четыре шестнадцатеричные цифры (наиболее значимые 16 битов) адреса удалены, а суффиксом для адреса в правой части записи псевдонима теперь является *ip6.tla-2.net*, поскольку известно, что этот адрес принадлежит к TLA 2. В *ip6.tla-2.net* мы находим:

```
$ORIGIN ip6.tla-2.net.
\[x00196600/24]   IN   DNAME     ip6.nlab.net.
```

Доменное имя в следующем запросе:

```
\[x00196642000102104bfffe100d24].ip6.tla-2.net
```

превращается в:

```
\[x42000102104bfffe100d24].ip6.nlab.net
```

Затем наш DNS-сервер посылает запрос для нового доменного имени DNS-серверам *ip6.nlab.net*. В зоне *ip6.nlab.net* присутствует такая запись:

```
$ORIGIN ip6.nlab.net.
\[x0042/8]    IN   DNAME    ip6.movie.edu.
```

которая превращает искомое доменное имя в:

```
\[x000102104bfffe100d24].ip6.movie.edu
```

Наконец, зона *ip6.movie.edu* содержит PTR-запись, которая позволяет получить окончательное доменное имя узла:

```
$ORIGIN ip6.movie.edu.
\[x000102104bfffe100d24/80]  IN   PTR   drunkenmaster.ip6.movie.edu.
```

К счастью, администраторы зоны чаще всего будут сталкиваться только с сопровождением PTR-записей, подобных тем, что присутствуют в *ip6.movie.edu*. Даже если администратор работает в агрегаторе высшего уровня или NLA-агрегаторе, создание DNAME-записей, «извлекающих» соответствующий NLA-идентификатор или идентификатор площадки из адресов клиентов, не столь уж непосильная задача. В результате можно будет использовать единственный файл для хранения информации, связанной с обратным отображением, даже несмотря на то, что каждый из узлов может иметь несколько адресов. Помимо этого, появляется возможность менять NLA-агрегаторы, не изменяя файлы данных зон.

- *TSIG*
- *Обеспечение безопасности DNS-сервера*
- *DNS и брандмауэры сети Интернет*
- *Расширения системы безопасности DNS*

11

Безопасность

– Надеюсь, волосы у тебя сегодня хорошо приклеены? – спросил Рыцарь, когда они тронулись в путь.
– Не лучше, чем всегда, – с улыбкой отвечала Алиса.
– Этого мало, – встревожился Рыцарь.
– Ветер тут в лесу такой сильный, что прямо рвет волосы с корнем!
– А вы еще не придумали средства от вырывания волос? – спросила Алиса.
– Нет, но зато я придумал средство от выпадения, – отвечал Рыцарь.

Почему следует думать о безопасности DNS? Зачем тратить время на обеспечение безопасности службы, которая по большей части занимается преобразованием имен в адреса? Мы расскажем читателям поучительную историю.

В июле 1997 года, в течение двух периодов по несколько дней каждый, пользователи сети Интернет, набиравшие в своих броузерах адрес *www.internic.net* с целью попасть на веб-сайт организации InterNIC, попадали на сайт, принадлежащий организации AlterNIC. (AlterNIC управляет альтернативным набором корневых DNS-серверов, которые делегируют авторитативность дополнительным доменам высшего уровня, с именами вроде *med* и *porn*.) Как это произошло? Евгений Кашпурев (Eugene Kashpureff), работавший в то время в организации AlterNIC, выполнил программу, которая «отравила» кэши крупных DNS-серверов по всему миру, заставив их думать, что адресом для имени *www.internic.net* в действительности является адрес веб-сервера AlterNIC.

Кашпурев никоим образом не пытался скрыть то, что он сделал; веб-сайт, на который попадали пользователи, был, вне всяких сомнений сайтом AlterNIC, а не InterNIC. А теперь вообразите, что некто отравил кэш вашего DNS-сервера, и имя *www.amazon.com* или *www.wellsfargo.com* приводит теперь на чужой веб-сервер, который находится вне юрисдикции местных законов. Теперь представьте, что ваши поль-

зователи при посещении сайта вводят номера и даты истечения действия своих кредитных карт, и вам все станет ясно.

Чтобы защитить пользователей от нарушений подобного рода, необходимо обеспечивать безопасность DNS. Безопасность DNS существует в нескольких ипостасях. Можно обеспечивать безопасность транзакций – запросов, ответов и всех прочих сообщений, посылаемых и получаемых DNS-сервером. Можно задуматься о безопасности DNS-сервера, об избирательных отказах в выполнении запросов, динамических обновлений, а также в передаче зональных данных – для неавторизованных адресов. Можно даже обезопасить зональные данные с помощью цифровых подписей.

Поскольку безопасность – одна из самых сложных тем в DNS, мы начнем с самого простого материала, и постепенно будем наращивать сложность.

TSIG

В BIND версии 8.2 появился новый механизм обеспечения безопасности для сообщений DNS, который носит название *транзакционных подписей*, сокращенно TSIG (transaction signatures). В TSIG используется механизм общих секретов и вычислительно необратимой хеш-функции для проверки подлинности сообщений DNS, в особенности ответов и обновлений.

Механизм TSIG, описанный в документе RFC 2845, относительно прост в настройке, не создает дополнительных сложностей для DNS-клиентов и DNS-серверов, а также достаточно гибок, чтобы обеспечить безопасность в контексте сообщений DNS (включая процесс передачи зоны) и динамических обновлений. (Прямая противоположность расширениям системы безопасности DNS, которые мы обсудим в конце этой главы.)

При настроенном и работающем механизме TSIG DNS-сервер или автор обновления добавляет TSIG-запись в раздел дополнительных данных сообщения DNS. TSIG-запись является «подписью» сообщения DNS и подтверждает, что отправитель сообщения обладает общим с получателем криптографическим ключом и что сообщение не изменялось после того, как было отправлено.[1]

[1] Апологеты от криптографии могут, конечно, заявить, что TSIG-«подписи» не являются подписями в криптографическом смысле, поскольку не позволяют производить подтверждение авторства. Поскольку любой владелец разделяемого ключа может создать подписанное сообщение, получатель подписанного сообщения не может утверждать, что оно отправлено действительным автором (поскольку сам вполне способен его подделать).

Необратимые хеш-функции

TSIG обеспечивает идентификацию и целостность данных посредством использования специального вида математических формул, который носит название *вычислительно необратимой хеш-функции*. Эта хеш-функция, известная также под именем криптографической контрольной суммы или дайджеста сообщения, вычисляет хеш-значение фиксированной длины на основе исходных данных произвольного объема. Волшебство вычислительно необратимой хеш-функции заключается в том, что каждый бит хеш-значения зависит от каждого из битов исходных данных. Если изменить единственный бит исходных данных, хеш-значение тоже изменится – очень сильно и непредсказуемо – настолько непредсказуемо, что задача обращения функции и нахождения ввода, приведшего к получению конкретного хешхеш-значения, является «вычислительно неосуществимой».

В механизме TSIG применяется вычислительно необратимая хеш-функция, которая называется MD5. В частности, разновидность MD5, которая называется HMAC-MD5. HMAC-MD5 работает в режиме учета ключей, то есть вычисляемое 128-битное хеш-значение зависит не только от исходных данных, но еще и от ключа.

TSIG-запись

Мы не станем подробно рассматривать синтаксис TSIG-записей, поскольку читателям нет необходимости его знать: TSIG – это «мета-запись», которая никогда не отражается в данных зоны и никогда не кэшируется DNS-клиентами и DNS-серверами. Отправитель сообщения DNS подписывает его с помощью TSIG-записи, а получатель удаляет и проверяет запись, прежде чем выполнять какие-либо действия, например, кэширование данных, содержащихся в сообщении.

Но следует знать, что TSIG-запись содержит хеш-значение, вычисленное для полного сообщения DNS и некоторых дополнительных полей. (Когда мы говорим «вычисленное для», то имеем в виду, что сообщение DNS в двоичном формате и дополнительные поля обрабатываются алгоритмом HMAC-MD5 с целью получения хеш-значения.) хеш-значение модифицируется по ключу, который является общим секретом отправителя и получателя сообщения. Положительный результат проверки хеш-значения доказывает, что сообщение было подписано владельцем общего секрета и что оно не изменялось после того, как было подписано.

Дополнительные поля TSIG-записи включают время подписи сообщения DNS. Это позволяет отражать атаки [повторного] воспроизведения (replay attacks), суть которых заключается в том, что взломщик перехватывает авторизованную транзакцию с подписью (например, динамическое обновление или удаление важной RR-записи) и воспроизводит ее позже. Получатель подписанного сообщения DNS проверяет

время подписи, чтобы убедиться, что это время находится в пределах допустимого (допустимое время определяется отдельным полем TSIG-записи).

Настройка TSIG

Прежде чем начать использовать TSIG для идентификации, необходимо создать один или несколько TSIG-ключей для сторон, обменивающихся данными. Так, если необходимо с помощью TSIG обезопасить передачу зоны с DNS-мастер-сервера *movie.edu* на вторичный, следует настроить оба сервера на использование общего ключа:

```
key terminator-wormhole.movie.edu. {
    algorithm hmac-md5;
    secret "skrKc4Twy/cIgIykQu7JZA==";
};
```

terminator-wormhole.movie.edu., аргумент оператора *key* в приведенном примере, является в действительности именем ключа, хотя выглядит как доменное имя. (Имя ключа кодируется в сообщениях DNS так же, как и обычные доменные имена.) В документе RFC по TSIG предлагается давать ключу имя, отражающее пару узлов, которые пользуются ключом. Помимо этого рекомендуется применять различные ключи для различных пар узлов.

Очень важно, чтобы имя ключа – а не только двоичные данные ключа – было одинаковым для обеих сторон, участвующих в транзакциях. В противном случае получатель при попытке проверить TSIG-запись обнаружит, что ничего не знает о ключе, который упоминается в этой TSIG-записи, и который был использован для вычисления хеш-значения. Такая ситуация приводит к получению примерно следующих ошибок:

```
Nov 21 19:43:00 wormhole named-xfer[30326]: SOA TSIG verification from server
[192.249.249.1], zone movie.edu: message had BADKEY set (17)
```

В настоящее время значение алгоритма всегда *hmac-md5*. Секрет представляет собой ключ в кодировке Base 64, созданный с помощью программы *dnssec-keygen*, входящей в состав пакета BIND 9, либо с помощью программы *dnskeygen*, входящей в состав пакета BIND 8. Ниже приводится пример создания ключа с помощью *dnssec-keygen*, которая проще в применении:

```
# dnssec-keygen -a HMAC-MD5 -b 128 -n HOST terminator-wormhole.movie.edu.
Kterminator-wormhole.movie.edu.+157+28446
```

Настройка *-a* позволяет задать имя алгоритма, с помощью которого должен быть создан ключ. (Это необходимо, поскольку *dnssec-keygen* умеет создавать ключи другого типа, как мы увидим в разделе, посвященном DNSSEC.) Параметр *-b* принимает в качестве аргумента длину

ключа; соответствующий документ RFC рекомендует использовать ключи длиной в 128 битов. Параметр *-n* в данном примере принимает в качестве аргумента HOST, тип генерируемого ключа. (В DNSSEC используются ключи типа ZONE.) Последний аргумент программы – имя ключа.

dnssec-keygen и *dnskeygen* создают в своих рабочих каталогах файлы, содержащие созданные ключи. *dnssec-keygen* печатает основу имен файлов на стандартный вывод. В приведенном примере программой *dnssec-keygen* были созданы файлы *Kterminator-wormhole.movie.edu.+157+28446.key* и *Kterminator-wormhole.movie.edu.+157+28446.private*. Ключ можно извлечь из любого файла. Если вы задаетесь вопросом, что это за забавные цифры – 157 и 28446, отвечаем: это номер алгоритма DNSSEC для ключа (157 соответствует алгоритму HMAC-MD5) и карта (fingerprint, «отпечатки пальцев») ключа (28446), хеш-значение, вычисленное для ключа с целью его последующей идентификации. Карта ключа не особенно полезна в контексте TSIG, но в DNSSEC реализована поддержка нескольких ключей для зоны, поэтому важно иметь возможность идентифицировать ключ по его карте.

Файл *Kterminator-wormhole.movie.edu.+157+28446.key* содержит строку:

```
terminator-wormhole.movie.edu. IN KEY 512 3 157 skrKc4Twy/cIgIykQu7JZA==
```

А вот содержимое *Kterminator-wormhole.movie.edu.+157+28446.private*:

```
Private-key-format: v1.2
Algorithm: 157 (HMAC_MD5)
Key: skrKc4Twy/cIgIykQu7JZA==
```

Можно также выбрать свой собственный ключ и перекодировать его в Base 64 с помощью программы *mmencode*:

```
% mmencode
foobarbaz
Zm9vYmFyYmF6
```

Поскольку реальный двоичный ключ, как подразумевается предписанием, является секретом, следует проявлять осторожность при переносе его на DNS-серверы (например, пользоваться *ssh*) и проследить за тем, чтобы ключ не мог прочесть любой желающий. Последнее достигается установлением для файла *named.conf* прав доступа, исключающих чтение файла пользователями, не входящими во владеющую группу, либо применением оператора *include* для чтения оператора *key* из другого файла, для которого установлены описанные права доступа:

```
include "/etc/dns.keys.conf";
```

Существует еще одна проблема, которая часто встречается при использовании TSIG: синхронизация времени. Отметка времени в TSIG-записи полезна для предотвращения атак, основанных на воспроизведе-

нии, но работоспособность этого свойства изначально уменьшается тем фактом, что часы DNS-серверов не синхронизированы. Это приводит к получению сообщений об ошибках примерно такого вида:

```
Nov 21 19:56:36 wormhole named-xfer[30420]: SOA TSIG verification from server
[192.249.249.1], zone movie.edu: BADTIME (-18)
```

Мы быстро избавились от этой проблемы, используя NTP (Network Time Protocol) – сетевой протокол времени.[1]

TSIG в работе

Теперь, уже совершив подвиг создания ключей TSIG для наших серверов, мы, вероятно, должны настроить серверы на практическое применение этих ключей. В BIND 8.2 и более поздних версий существует возможность обеспечить безопасность запросов, ответов, процесса передачи зоны и динамических обновлений с помощью TSIG.

Основой настройки является предписание *keys* инструкции *server*, которое сообщает DNS-серверу, что необходимо подписывать обычные запросы и запросы на передачу зоны, направляемые определенному удаленному DNS-серверу. К примеру, следующее предписание говорит локальному DNS-серверу, *wormhole.movie.edu*, что следует подписывать все запросы подобного рода, посылаемые по адресу 192.249.249.1 (узлу *terminator.movie.edu*) ключом *terminator-wormhole.movie.edu*:

```
server 192.249.249.1 {
    keys { terminator-wormhole.movie.edu.; };
};
```

На узле *terminator.movie.edu* мы можем ограничить передачу зоны до пакетов, подписанных ключом *terminator-wormhole.movie.edu*:

```
zone "movie.edu" {
    type master;
    file "db.movie.edu";
    allow-transfer { key terminator-wormhole.movie.edu.;
    };
};
```

terminator.movie.edu также подписывает этим ключом пакеты при передаче зоны, что позволяет серверу *wormhole.movie.edu* проверять их аутентичность.

Существует также возможность ограничить динамические обновления с помощью TSIG, используя предписания *allow-update* и *update-policy*, как мы показывали в предыдущей главе.

---

[1] Более подробная информация по NTP доступна на веб-сайте Time Synchronization Server по адресу *http://www.eecis.udel.edu/~ntp*.

Программы *nsupdate*, которые входят в состав пакета BIND версии 8.2 и более поздних, поддерживают посылку динамических обновлений с TSIG-подписями. Если ключевые файлы, созданные программой *dnssec-keygen* все еще существуют, имя любого из них можно указать в качестве аргумента ключа *-k* программы *nsupdate*. В следующих командах используется *nsupdate* из пакета BIND версии 9:

```
% nsupdate -k Kterminator-wormhole.movie.edu.+157+28446.key
```

или:

```
% nsupdate -k Kterminator-wormhole.movie.edu.+157+28446.private
```

В BIND версии 8.2 и более поздних синтаксис применения *nsupdate* несколько отличается. *-k* в качестве аргументов принимает имя каталога и ключа, разделенные двоеточием:

```
% nsupdate -k /var/named:terminator-wormhole.movie.edu.
```

Если файлов под рукой нет (возможно, что *nsupdate* выполняется на другом узле), можно указать имя ключа и секрет в командной строке *nsupdate* из пакета BIND 9:

```
% nsupdate -y terminator-wormhole.movie.edu.:skrKc4Twy/cIgIykQu7JZA==
```

Первым аргументом *-y* является имя ключа, за ним следует двоеточие, а затем секрет в кодировке Base 64. Нет необходимости маскировать символы секрета, поскольку кодировка Base 64 не содержит метасимволы командного интерпретатора, но при желании это можно делать.

Net::DNS, модуль Perl, разработанный Майклом Фером также позволяет посылать динамические обновления и запросы передачи зоны с TSIG-подписью. Более подробно модуль Net::DNS описан в главе 15 «Программирование с использованием библиотечных функций».

Итак, у нас есть удобный механизм обеспечения безопасности транзакций DNS, и мы переходим к вопросам безопасности собственно DNS-сервера.

Обеспечение безопасности DNS-сервера

В BIND версии 4.9 появились важные возможности, связанные с обеспечением защиты DNS-сервера. В BIND 8 и 9 эта традиция продолжилась добавлением новых возможностей, связанных с безопасностью. Они особенно важны, если DNS-сервер работает в сети Интернет, но также полезны для чисто внутренних DNS-серверов.

Мы начнем с обсуждения мер, которые следует предпринимать для всех DNS-серверов, которые необходимо обезопасить. Затем мы опишем модель, в которой DNS-серверы разделяются на два класса, один из которых обслуживает только запросы клиентов, а второй отвечает на запросы других DNS-серверов.

Версия BIND

Один из наиболее логичных способов обезопасить свой DNS-сервер – воспользоваться относительно свежей версией пакета BIND. Все версии BIND до 8.2.3 уязвимы по меньшей мере для нескольких из широко распространенных вариантов атак. Самая последняя информация по уязвимостям различных версий BIND доступна по адресу *http://www.isc.org/products/BIND/bind-security.html*.

Не ограничивайтесь новой версией пакета: новые атаки изобретаются постоянно, поэтому придется как следует постараться, чтобы совместить минимальную уязвимость и использование самой последней из безопасных версий BIND. Существенным подспорьем в процессе достижения этой цели является регулярное чтение конференции *comp.protocols.dns.bind*.

Существует еще один аспект связи версии пакета BIND с безопасностью: если взломщик способен легко выяснить, какую версию BIND вы применяете, он, вполне возможно, скорректирует свои атаки, исходя из уязвимых мест конкретно этой версии. Если читатели не в курсе, поясним: начиная примерно с BIND версии 4.9, DNS-серверы отвечают на определенный запрос информацией о своей версии. Если запросить TXT-записи класса CHAOSNET для доменного имени *version.bind*, BIND с удовольствием вернет примерно следующий ответ:

```
% dig txt chaos version.bind.

; <<>> DiG 9.1.0 <<>> txt chaos version.bind.
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34772
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;version.bind.                  CH      TXT

;; ANSWER SECTION:
version.bind.           0       CH      TXT     "9.1.0"
```

Чтобы решить эту проблемы, в BIND версии 8.2 и более поздних реализована возможность настраивать ответ DNS-сервера на запрос *version.bind*:

```
options {
    version "NE TVOE DELO";
};
```

Естественно, получение ответа «NE TVOE DELO» покажет внимательному взломщику, что используется версия 8.2 или более поздняя, но не даст ему никакой конкретики.

Ограничение запросов

До появления BIND версии 4.9 у администраторов не было возможности проконтролировать, кто посылает запросы DNS-серверам. И это имело определенный смысл: основная идея разработки DNS заключалась в том, чтобы сделать информацию легко доступной по всей сети Интернет.

Но сеть перестала быть исключительно открытой. В частности, люди, управляющие брандмауэрами Интернета, могут иметь достаточные основания скрывать определенные сегменты своего пространства имен от большей части сетевого мира, делая их доступными лишь ограниченному кругу лиц.

Предписание *allow-query*, существующее в BIND 8 и 9, позволяет контролировать доступ на уровне IP-адресов, которым разрешается выполнение запросов. Список управления доступом (access control list, ACL) может применяться к запросам данных из определенной зоны, либо к любым запросам, получаемым DNS-сервером. В частности, список управления доступом позволяет указать, каким IP-адресам разрешается посылка запросов DNS-серверу.

Ограничение для всех запросов

Глобальное предписание *allow-query* выглядит следующим образом:

```
options {
      allow-query { список_адресов; };
};
```

Так, чтобы разрешить нашему серверу отвечать только на запросы из трех основных сетей Университета кинематографии, мы воспользовались следующей инструкцией:

```
options {
      allow-query { 192.249.249/24; 192.253.253/24; 192.253.254/24; };
};
```

Ограничение запросов для определенной зоны

BIND 8 и 9 позволяют применить список управления доступом к определенной зоне. В этом случае следует просто использовать *allow-query* в качестве части оператора *zone* для зоны, которую необходимо защитить:

```
acl "HP-NET" { 15/8; };

zone "hp.com" {
      type slave;
      file "bak.hp.com";
      masters { 15.255.152.2; };
      allow-query { "HP-NET"; };
};
```

Авторитативный сервер любого типа, первичный или вторичный, может использовать список доступа для ограничения доступа к зоне. Списки управления доступом, применяемые в зоне, имеют более высокий приоритет, чем глобальные ACL, в пределах запросов для этой зоны. Более того, зональные ACL-списки могут быть менее строгими, чем глобальные. Если зональный ACL-список не определен, будут использованы глобальные ACL-списки.

В BIND версии 4.9 описанная функциональность обеспечивается записью *secure_zone*. Эта запись ограничивает не только запросы для отдельных RR-записей, но также и передачу зоны. (В BIND 8 и 9 ограничения, налагаемые на передачу зоны, определяются отдельно.) При этом в DNS-серверах BIND 4.9 не существует механизма определения, кому разрешено посылать вашим DNS-серверам запросы данных из зон, для которых эти серверы *не являются* авторитативными; механизм *secure_zone* распространяется только на зоны, в пределах авторитативности DNS-серверов.

Чтобы воспользоваться механизмом *secure_zone*, необходимо включить одну или несколько специальных TXT-записей в данные зоны на первичном DNS-мастер-сервере. Передача этих записей на дополнительные DNS-серверы зоны произойдет автоматически. Разумеется, только вторичные серверы BIND 4.9 поймут смысл этих записей.

TXT-записи являются особенными, потому что они связываются с псевдоименем домена *secure_zone*, а формат данных этой RR-записи также является специальным:

```
адрес:маска
```

или:

```
адрес:H
```

В первом варианте *адрес* – это IP-адрес в записи через точку для сети, которой следует разрешить доступ к данным зоны. *Маска* – это сетевая маска для сети, определяемой адресом. Чтобы разрешить всей сети 15/8 доступ к данным зоны, следует использовать запись 15.0.0.0:255.0.0.0. Для задания диапазона IP-адресов от 15.254.0.0 до 15.255.255.255 можно указать 15.254.0.0:255.254.0.0.

Второй вариант позволяет указать адрес отдельного узла, которому следует разрешить доступ к данным. Буква *H* в данном случае является эквивалентом маски 255.255.255.255; иначе говоря, проверяется каждый бит 32-битного адреса. Таким образом, запись 15.255.152.4:H дает узлу с IP-адресом 15.255.152.4 право получать данные из зоны.

Допустим, мы хотим ограничить получение информации из *movie.edu* узлами сетей Университета кинематографии, но при этом используем DNS-сервер BIND 4.9 вместо BIND 8 или 9. Мы могли бы добавить следующие строки к файлу *db.movie.edu* первичного мастер-сервера зоны *movie.edu*:

```
secure_zone    IN    TXT    "192.249.249.0:255.255.255.0"
secure_zone    IN    TXT    "192.253.253.0:255.255.255.0"
secure_zone    IN    TXT    "192.253.254.0:255.255.255.0"
secure_zone    IN    TXT    "127.0.0.1:H"
```

Обратите внимание, мы включили в список доступа адрес loopback-интерфейса (127.0.0.1), чтобы клиент, работающий на том же узле, что и DNS-сервер, мог посылать локальному DNS-серверу запросы.

Если бы забыли указать *:H*, то увидели бы следующее сообщение в log-файле демона *syslog*:

```
Aug 17 20:58:22 terminator named[2509]: build_secure_netlist
       (movie.edu): addr (127.0.0.1) is not in mask (0xff000000)
```

Помимо этого, обратите внимание, что записи *secure_zone* действуют только для зоны, в пределах которой определены, то есть для зоны *movie.edu* в нашем примере. Чтобы предотвратить несанкционированный доступ к данным других зон на сервере, следует добавить записи *secure_zone* в файлы данных этих зон на их первичных DNS-мастер-серверах.

Предотвращение несанкционированной передачи зоны

Важно не только контролировать, кто имеет право посылать запросы вашему DNS-серверу, но и гарантировать, что только подлинные вторичные DNS-серверы могут запросить передачу зоны. Пользователи на удаленных узлах, которые посылают запросы DNS-серверу, могут получать записи (к примеру, адресные) только для уже известных им доменных имен, по одному имени за запрос. Пользователи, которые могут инициировать передачу зоны с вашего сервера, смогут прочитать все записи зон. Разница примерно такая же, как между тем, чтобы разрешить произвольным людям использовать коммутатор для поиска телефонного номера Джона Кьюбикла, и тем, чтобы послать им копию телефонного справочника вашей корпорации.

С помощью предписания *allow-transfer* в BIND 8 и 9 или с помощью инструкции *xfrnets* BIND 4.9 администратор может указать на необходимость использования списка управления доступом при передаче зоны. *allow-transfer* в пределах оператора *zone* ограничивает передачу для определенной зоны, а при указании в операторе *options* – для всех случаев передачи зоны. В качестве аргумента используется список адресов.

IP-адреса дополнительных DNS-серверов зоны *movie.edu*: 192.249.249.1 и 192.253.253.1 (*wormhole.movie.edu*), 192.249.249.9 и 192.253.253.9 (*zardoz.movie.edu*). Следующий оператор *zone*:

```
zone "movie.edu" {
      type master;
```

```
        file "db.movie.edu";
       allow-transfer { 192.249.249.1; 192.253.253.1; 192.249.249.9; 192.253.253.9; };
};
```

разрешает получение зоны *movie.edu* с первичного DNS-мастер-сервера только перечисленным вторичным серверам. Поскольку по умолчанию DNS-серверы BIND 8 и 9 разрешают получение зоны по запросам с любых IP-адресов, и поскольку взломщики могут с такой же легкостью произвести получение зоны с одного из дополнительных DNS-серверов, следует добавить в файлы настройки вторичных серверов примерно такой оператор *zone*:

```
zone "movie.edu" {
      type slave;
      masters { 192.249.249.3; };
      file "bak.movie.edu";
      allow-transfer { none; };
};
```

В BIND 8 и 9 существует возможность применять глобальный ACL-список к передаче зоны. Это происходит по умолчанию для всех зон, не определяющих явным образом собственные списки управления доступом в операторе *zone*. К примеру, мы могли бы ограничить передачу зоны внутренними IP-адресами:

```
options {
      allow-transfer { 192.249.249/24; 192.253.253/24; 192.253.254/24; };
};
```

Инструкция BIND 4.9 – *xfrnets* также применяет список управления доступом ко всем случаям передачи зоны. В качестве аргументов *xfrnets* выступают сети или IP-адреса, которым разрешается получать зону с DNS-сервера. Сети определяются записью IP-адресов в десятичной нотации. Например:

```
xfrnets 15.0.0.0 128.32.0.0
```

разрешает получение зоны узлам сети 15/8 или 128.32/16. В отличие от TXT-записи *secure_zone*, *xfrnets* накладывает ограничения на все зоны, для которых сервер является авторитативным.

Если необходимо указать лишь часть сети (вплоть до единственного IP-адреса), можно добавить маску сети. Синтаксис для включения маски сети следующий: *сеть&маска сети*. Обратите внимание, что пробелы в этой записи недопустимы.

Следующая инструкция *xfrnets* позволяет сократить число адресов из предыдущего примера до единственного IP-адреса15.255.152.4 и подсети 128.32.1/24:

```
xfrnets 15.255.152.4&255.255.255.255 128.32.1.0&255.255.255.0
```

И наконец, как мы упоминали ранее, новомодные DNS-серверы BIND версии 8.2 и более поздних позволяют ограничить случаи передачи зоны дополнительными DNS-серверами, которые включают в запрос на передачу правильную транзакционную подпись. На основном DNS-сервере необходимо определить ключ в операторе *key*, а затем указать этот ключ в списке допустимых адресов:

```
key terminator-wormhole. {
    algorithm hmac-md5;
    secret "UNd5xYLjzOFPkoqWRymtgI+paxW927LU/gTrDyulJRI=";
};

zone "movie.edu" {
    type master;
    file "db.movie.edu";
    allow-transfer { key terminator-wormhole.; };
};
```

На стороне дополнительного DNS-сервера следует предписать использование подписи для запросов на передачу зоны. Ключ тот же:

```
key terminator-wormhole. {
    algorithm hmac-md5;
    secret "UNd5xYLjzOFPkoqWRymtgI+paxW927LU/gTrDyulJRI=";
};

server 192.249.249.3 {
    keys { terminator-wormhole.; };  // подписывать все запросы
                                     // к 192.249.249.3 этим ключом
};

zone "movie.edu" {
    type slave;
    masters { 192.249.249.3; };
    file "bak.movie.edu";
};
```

Если первичный DNS-мастер-сервер доступен из сети Интернет, вероятно, имеет смысл ограничить передачу зоны вторичными DNS-серверами. Если DNS-сервер расположен за брандмауэром, то беспокоиться о нем не стоит, если только вы не подозреваете собственных служащих в чтении зональных данных.

Выполнение BIND с наименьшими полномочиями

Если сетевой сервер вроде BIND работает от пользователя с высокими полномочиями, это представляет опасность; а DNS-сервер BIND обычно работает от пользователя root. Если взломщик находит уязвимое место DNS-сервера, позволяющее ему читать и изменять файлы, то фактически получает полный доступ к файловой системе. Если он сможет воспользоваться недостатком, который позволяет ему выполнять команды на системе, то будет выполнять их в качестве пользователя root.

В BIND версии 8.1.2 и более поздних содержится код, который позволяет изменять группу и пользователя, от которого работает DNS-сервер. Это позволяет запускать DNS-сервер с *наименьшими полномочиями*: то есть с минимальным набором прав, который необходим серверу для выполнения работы. В такой ситуации, если взломщик проникнет в систему через DNS-сервер, он не будет иметь полномочий администратора системы.

В BIND версии 8.1.2 и более поздних также существует возможность применять *chroot( )* при запуске DNS-сервера: то есть изменять вид файловой системой таким образом, чтобы корневой каталог для сервера на самом деле являлся не более чем отдельным каталогом в файловой системе узла. По сути дела, такая настройка ограничивает существование DNS-сервера этим каталогом, и, само собой, взломщиков, которые успешно проникли через защиту DNS-сервера.

Следующие ключи командной строки позволяют использовать описанные механизмы:

-u

Указать имя пользователя или идентификатор пользователя, который DNS-серверу следует применять при работе. Пример: *named -u bin*.

-g

Указать имя группы или идентификатор группы, который DNS-серверу следует применять при работе. Пример *named -g other*. Если указать только имя или идентификатор пользователя, DNS-сервер использует основную группу этого пользователя. DNS-серверы BIND 9 всегда используют основную группу пользователя, поэтому в них не поддерживается ключ *-g*.

-t

Указать каталог, которым с помощью *chroot( )* ограничивается DNS-сервер.

Приняв решение о применении ключей *-u* и *-g* следует понять, какого пользователя и какую группу указать. Лучше всего – создать специального нового пользователя и новую группу для работы DNS-сервера, например *named*. Поскольку DNS-сервер производит чтение файла *named.conf* прежде чем перестать работать от пользователя *root*, нет необходимости изменять права доступа для этого файла. Однако, вполне возможно, потребуется изменить права доступа и владельца для файлов данных зоны, чтобы DNS-сервер, работая от пользователя с пониженными полномочиями, мог их прочитать. Если применяется динамическое обновление, придется дать DNS-серверу права на запись в файлы динамически обновляемых зон.

Если DNS-сервер настроен таким образом, что сообщения записываются в файлы (а не в log-файл демона *syslog*), убедитесь, что эти файлы

существуют и доступны для записи DNS-серверу, прежде чем запускать сервер.

Применение ключа *-t* связано с некоторыми тонкостями настройки. В частности, следует убедиться, что все файлы, используемые *named*, присутствуют в каталоге, которым ограничивается DNS-сервер. Приводимая ниже процедура позволяет правильным образом подготовить новую среду для сервера. Предположим, что речь идет о каталоге */var/named*:[1]

1. Если каталог */var/named* не существует, создайте его. Создайте подкаталоги *dev*, *etc*, *lib*, *usr* и *var*. В подкаталоге *usr* создайте подкаталог *sbin*. В подкаталоге *var* – подкаталоги *named* и *run*:

```
# mkdir /var/named
# cd /var/named
# mkdir -p dev etc lib usr/sbin var/named var/run
```

2. Скопируйте файл *named.conf* в */var/named/etc/named.conf*:

```
# cp /etc/named.conf etc
```

3. Если вы работаете с BIND 8, скопируйте исполняемый файл *named-xfer* в подкаталог *usr/sbin/* либо *etc* (в зависимости от того, где этот файл существовал раньше, в */usr/sbin* или */etc*).

```
# cp /usr/sbin/named-xfer usr/sbin
```

   Как вариант можно поместить этот файл где угодно в пределах каталога */var/named* и использовать предписание *named-xfer*, чтобы объяснить серверу *named*, где искать этот файл. Помните, что необходимо удалить */var/named* из пути к файлу, поскольку при чтении файл *named.conf /var/named* будет являться корнем файловой системы. (Если вы используете BIND 9, пропустите этот шаг, поскольку в BIND 9 не применяется *named-xfer*.)

4. Создайте файл *dev/null* в новой «файловой системе»:

```
# mknod dev/null c 1 3
```

5. Если вы работаете с BIND 8 скопируйте стандартную разделяемую библиотеку C и загрузчик в подкаталог *lib*:

```
# cp /lib/libc.so.6 /lib/ld-2.1.3.so lib
# ln -s lib/ld-2.1.3.so lib/ld-linux.so.2
```

   Пути могут изменяться в зависимости от используемой операционной системы. Серверы BIND 9 самодостаточны, им не нужны библиотеки.

6. Отредактируйте загрузочные файлы таким образом, чтобы демон *syslogd* запускался с дополнительным ключом и аргументом ключа:

[1] Процедура разработана для системы Red Hat Linux 6.2, так что при использовании на других системах может несколько отличаться.

-a /var/named/dev/log. Во многих современных вариантах Unix запуск демона *syslogd* производится из файла */etc/rc.d/init.d/syslog*. При следующем перезапуске демона *syslogd* будет создан файл */var/named/dev/log*, в который *named* будет записывать сообщения.

Если демон *syslogd* не поддерживает ключ *-a*, воспользуйтесь оператором *logging* – который описан в главе 7 «Работа с BIND» – для записи сообщений в *chroot*-каталоге.

7. Если вы работаете с BIND 8 и используете ключ *-u* или *-g*, создайте в подкаталоге *etc* файлы *passwd* и *group*, чтобы обеспечить отображение аргументов ключей *-u* и *-g* в соответствующие числовые значения (как вариант можно просто использовать числовые значения в качестве аргументов ключей):

```
# echo "named:x:42:42:named:/:" > etc/passwd
# echo "named::42" > etc/group
```

Затем добавьте эти записи в файлы */etc/passwd* и */etc/group* системы. Если речь идет о DNS-сервере BIND 9, можно обойтись добавлением записей в системные файлы */etc/passwd* и */etc/group*, поскольку DNS-серверы BIND 9 производят чтение интересующей их информации перед вызовом *chroot( )*.

8. И наконец, отредактируйте загрузочные файлы, чтобы запускать *named* с ключом *-t /var/named* при загрузке системы. Как и в случае с демоном *syslogd*, во многих современных вариантах Unix запуск *named* производится из файла */etc/rc.d/init.d/named*.

Если вы привыкли использовать *ndc* для управления DNS-сервером BIND 8, можно продолжать пользоваться этой программой, указывая имя Unix-сокета в качестве аргумента ключа *-c*:

```
# ndc -c /var/named/var/run/ndc reload
```

rndc будет работать с DNS-сервером BIND 9 как и раньше, поскольку общается с сервером через порт 953.

Разделение функций DNS-серверов

По сути дела у DNS-сервера две основных задачи: отвечать на итеративные запросы удаленных DNS-серверов и отвечать на рекурсивные запросы локальных DNS-клиентов. Если мы разделим эти роли – выделив один набор DNS-серверов для работы с итеративными запросами, а другой набор отведя под ответы на рекурсивные запросы – то сможем более эффективно обеспечивать безопасность этих DNS-серверов.

«Делегированный» DNS-сервер

Некоторые из DNS-серверов отвечают на нерекурсивные запросы других DNS-серверов сети Интернет, поскольку эти DNS-серверы встречаются в NS-записях, делегирующих им ваши зоны. Такие DNS-серверы мы называем «делегированными».

Существуют особые меры, которые можно принять в целях обеспечения безопасности делегированных DNS-серверов. Но прежде необходимо убедиться, что эти DNS-серверы не получают рекурсивных запросов (то есть ни один из клиентов не настроен на использование этих серверов и никакой DNS-сервер не использует их в качестве ретрансляторов). Некоторые из принимаемых мер – скажем, предписание серверу отвечать нерекурсивно даже на рекурсивные запросы – препятствуют использованию этого сервера клиентами. Если какие-либо клиенты все-таки обращаются к делегированному DNS-серверу, имеет смысл задуматься о создании отдельного класса DNS-серверов, которые будут заниматься исключительно обслуживанием DNS-клиентов, либо об использовании варианта «Два сервера в одном», который описан далее в этой главе.

Убедившись, что DNS-сервер отвечает только на запросы других серверов, можно выключить рекурсию. Это исключает крупное направление атак: большинство атак, связанных с подделкой IP-пакетов, основано на побуждении атакуемого DNS-сервера сделать запрос DNS-серверам, которые управляются взломщиком, путем отправки атакуемому серверу рекурсивного запроса для доменного имени, входящего в зону, которая обслуживается серверами взломщика. Чтобы выключить рекурсию, воспользуйтесь следующим оператором для сервера BIND 8 или 9:

```
options {
     recursion no;
};
```

либо, если используется BIND 4.9:

```
options no-recursion
```

Следует также ограничить число получателей зоны известными вторичными серверами (этот момент описан выше, в разделе «Предотвращение несанкционированной передачи зоны»). И наконец, можно отключить поиск связующих записей. Некоторые DNS-серверы автоматически пытаются произвести разрешение доменных имен любых DNS-серверов, встречающихся в NS-записях; чтобы предотвратить эти действия и запретить DNS-серверу посылать собственные запросы, используйте следующую настройку DNS-сервера BIND 8 (в DNS-серверах BIND 9 поиск связующих записей по умолчанию отключен):

```
options {
      fetch-glue no;
};
```

либо, для DNS-сервера BIND:

```
options no-fetch-glue
```

«Разрешающий» DNS-сервер

Будем называть DNS-сервер, который обслуживает одного или нескольких клиентов или настроен в качестве ретранслятора для другого DNS-сервера, – «разрешающим» DNS-сервером. В отличие от делегированного DNS-сервера, разрешающий не может отказаться от выполнения рекурсивных запросов. Как следствие, его относительно безопасная настройка выглядит иначе. Поскольку известно, что наш DNS-сервер должен принимать запросы только от наших собственных DNS-клиентов, мы можем настроить его таким образом, чтобы он отвергал запросы с любого адреса, который не принадлежит списку IP-адресов наших клиентов.

Только в BIND 8 и 9 существует возможность ограничивать набор IP-адресов, с которых принимаются произвольные запросы. (В DNS-серверах BIND 4.9 существует лишь возможность ограничить набор IP-адресов, с которых принимаются запросы для зон авторитативности сервера – посредством использования TXT-записей *secure_zone*; но нас больше беспокоят рекурсивные запросы для всех прочих зон.) Следующее предписание *allow-query* ограничивает отправителей запросов нашей внутренней сетью:

```
options {
    allow-query { 192.249.249/24; 192.253.253/24; 192.253.254/24; };
};
```

При такой настройке посылать нашему DNS-серверу рекурсивные запросы, приводящие к отправке запросов другим DNS-серверами, смогут только клиенты внутренней сети, которые в достаточной степени доброжелательны.

Существует еще одна настройка, позволяющая повысить безопасность DNS-сервера – *use-id-pool*:

```
options {
    use-id-pool yes;
};
```

Предписание *use-id-pool* появилось в BIND 8.2. Оно сообщает серверу, что следует проявлять особую изобретательность при генерации случайных чисел-идентификаторов запросов. Обычно идентификаторы сообщений не являются в достаточной степени случайными, чтобы предотвратить прямолинейные атаки, которые заключаются в угадывании идентификаторов отправленных сервером запросов для создания поддельных ответов.

Код, реализующий более совершенную генерацию идентификаторов, стал стандартной частью BIND 9, поэтому для DNS-серверов BIND 9 это предписание можно не использовать.

Два сервера в одном

Что делать в случае, когда есть только один сервер для обслуживания ваших зон и DNS-клиентов, а покупка второго компьютера для еще одного DNS-сервера сопряжена с расходами, которые вы не можете себе позволить? Варианты по-прежнему существуют. Есть два решения, связанных с использованием одного сервера и возможностей настройки BIND 8 или 9. Один из вариантов – разрешить кому-угодно запрашивать информацию для зон, находящихся в пределах авторитативности DNS-сервера, но получение любой другой информации разрешить только внутренним клиентам. Удаленные клиенты смогут посылать DNS-серверу рекурсивные запросы, но эти запросы должны касаться информации из зон, для которых сервер является авторитативными, то есть они не будут приводить к созданию дополнительных запросов.

Вот файл *named.conf* для этого варианта настройки:

```
acl "internal" {
    192.249.249/24; 192.253.253/24; 192.253.254/24;
};

acl "slaves" {
    192.249.249.1; 192.253.253.1; 192.249.249.9; 192.253.253.9;
};

options {
    directory "/var/named";
    allow-query { "internal"; };
    use-id-pool yes;
};

zone "movie.edu" {
    type master;
    file "db.movie.edu";
    allow-query { any; };
    allow-transfer { "slaves"; };
};

zone "249.249.192.in-addr.arpa" {
    type master;
    file "db.192.249.249";
    allow-query { any; };
    allow-transfer { "slaves"; };
};
```

В данном случае менее строгий список управления доступом применяются для запросов информации из зон, находящихся в пределах авторитативности DNS-сервера, а более строгий – для ограничения всех остальных запросов.

В случае использования BIND 8.2.1 или более поздней версии можно упростить настройку, используя предписание *allow-recursion*:

```
acl "internal" {
```

```
    192.249.249/24; 192.253.253/24; 192.253.254/24;
};

acl "slaves" {
    192.249.249.1; 192.253.253.1; 192.249.249.9; 192.253.253.9;
};

options {
    directory "/var/named";
    allow-recursion { "internal"; };
    use-id-pool yes;
};

zone "movie.edu" {
    type master;
    file "db.movie.edu";
    allow-transfer { "slaves"; };
};

zone "249.249.192.in-addr.arpa" {
    type master;
    file "db.192.249.249";
    allow-transfer { "slaves"; };
};
```

Предписания *allow-query* уже не нужны: хотя DNS-сервер и может получать запросы, созданные за пределами внутренней сети, он будет считать их нерекурсивными во всех случаях. Как следствие, внешние запросы не заставят DNS-сервер создавать новые запросы. Этот вариант настройки также избавлен от одного недостатка предыдущего: если DNS-сервер является авторитативным для родительской зоны, то может получать запросы от удаленных DNS-серверов, которые пытаются произвести разрешение доменного имени из поддомена этой зоны. Решение с использованием *allow-query* привело бы к отказам выполнять такие, вполне допустимые запросы, в отличие от варианта с *allow-recursion*.

Еще один вариант связан с выполнением двух процессов *named* на одном узле. Один процесс для делегированного DNS-сервера, второй – для разрешающего. Поскольку нет способа объяснить удаленным серверам или клиентам, что один из серверов принимает запросы через нестандартный порт, эти серверы должны выполняться на разных IP-адресах.

Разумеется, если на узле присутствует более одного сетевого интерфейса, это не проблема. Даже если сетевой интерфейс всего один, операционная система может поддерживать псевдонимы IP-адресов. Псевдонимы позволяют указать несколько IP-адресов для одного сетевого интерфейса. За каждым IP-адресом можно закрепить один процесс *named*. И наконец, даже если в используемой операционной системе отсутствует реализация IP-псевдонимов, можно закрепить один сервер *named* за IP-адресом сетевого интерфейса, а второй – за адресом

loopback-интерфейса. Процесс, закрепленный за адресом обратной связи, сможет получать запросы только от локального узла, но это идеально, если необходимо обслуживать только локальный DNS-клиент.

Сначала приведем файл *named.conf* для делегированного DNS-сервера, закрепленного за IP-адресом сетевого интерфейса:

```
acl "slaves" {
    192.249.249.1; 192.253.253.1; 192.249.249.9; 192;253.253.9; };
};

options {
    directory "/var/named-delegated";
    recursion no;
    fetch-glue no;
    listen-on { 192.249.249.3; };
    pid-file "/var/run/named.delegated.pid";
};

zone "movie.edu" {
    type master;
    file "db.movie.edu";
    allow-transfer { "slaves"; };
};

zone "249.249.192.in-addr.arpa" {
    type master;
    file "db.192.249.249";
    allow-transfer { "slaves"; };
};

zone "." {
    type hint;
    file "db.cache";
};
```

А вот *named.conf* для разрешающего DNS-сервера, закрепленного за адресом обратной связи:

```
options {
    directory "/var/named-resolving";
    listen-on { 127.0.0.1; };
    pid-file "/var/run/named.resolving.pid";
    use-id-pool yes;
};

zone "." {
    type hint;
    file "db.cache";
};
```

Обратите внимание, что список управления доступом для разрешающего DNS-сервера не нужен, поскольку этот сервер получает запросы только с loopback-адреса, но не с других узлов. (Если бы разрешающий

DNS-сервер получал запросы через IP-псевдоним или второй сетевой интерфейс, можно было бы воспользоваться предписанием *allow-query*, чтобы ограничить использование этого DNS-сервера.) На делегированном сервере мы отключили рекурсию, но на разрешающем она должна быть включена. Помимо этого, мы определили для каждого сервера отдельный PID-файл и отдельный рабочий каталог, чтобы не возникало конфликтов из-за использования файлов с одинаковыми именами при создании PID-файлов, файлов отладочной диагностики и файлов статистики.

Чтобы можно было использовать разрешающий DNS-сервер, принимающий запросы через адрес обратной связи, файл *resolv.conf* на локальном узле должен содержать строку:

```
nameserver 127.0.0.1
```

в качестве самой первой инструкции *nameserver*.

Если используется BIND 9, можно объединить настройки двух DNS-серверов в один файл с помощью видов:

```
options {
    directory "/var/named";
};

acl "internal" {
    192.249.249/24; 192.253.253/24; 192.253.254/24;
};

view "internal" {
    match-clients { "internal"; };
    recursion yes;

    zone "movie.edu" {
        type master;
        file "db.movie.edu";
    };

    zone "249.249.192.in-addr.arpa" {
        type master;
        file "db.192.249.249";
    };

    zone "." {
        type hint;
        file "db.cache";
    };
};

view "external" {
    match-clients { any; };
    recursion no;

    zone "movie.edu" {
        type master;
```

```
            file "db.movie.edu";
        };

        zone "249.249.192.in-addr.arpa" {
            type master;
            file "db.192.249.249";
        };

        zone "." {
            type hint;
            file "db.cache";
        };
    };
```

Достаточно простой вариант настройки: два вида, внутренний и внешний. Для внутреннего вида, который относится только ко внутренней сети, рекурсия включена. Внешний вид относится ко всем остальным, рекурсия выключена. Зоны *movie.edu* и *249.249.192.in-addr.arpa* идентичны в обоих видах. С помощью видов можно сделать гораздо больше – скажем, определить различные версии зон для внутреннего и внешнего видов, но мы отложим такие варианты до следующего раздела.

DNS и брандмауэры сети Интернет

При разработке DNS брандмауэры сети Интернет в расчет не принимались. Доказательством гибкости DNS и ее реализации в пакете BIND является тот факт, что DNS можно настроить для работы с брандмауэрами и даже через них.

Однако настройка BIND для работы в экранированной среде – задача не то чтобы сложная, но требующая качественного, полного понимания DNS и некоторых малоизвестных особенностей BIND. Описание настройки занимает приличную часть этой главы, поэтому начнем с краткого путеводителя по нему.

Сначала мы рассмотрим два крупных семейства брандмауэров Интернета – пакетные фильтры и шлюзы прикладного уровня. Особенности каждого семейства влияют на настройку BIND для совместной работы с брандмауэром. Затем мы опишем две наиболее часто используемые совместно с брандмауэрами структуры DNS – ретрансляторы и внутренние корневые DNS-серверы, изучим их достоинства и недостатки. Будет представлено решение, объединяющее преимущества внутренних корневых серверов и ретрансляторов, – зоны ретрансляции. И наконец, мы рассмотрим расщепление пространства имен и настройку узла-бастиона, который является сердцем брандмауэра.

Типы программного обеспечения брандмауэров

Прежде чем начать настройку BIND для работы с брандмауэром, необходимо понимать, на что способен сетевой экран. Потенциал брандма-

уэра влияет на выбор архитектуры DNS и определение способа ее реализации. Если вы не знаете ответов на вопросы, задаваемые в этом разделе, найдите в вашей организации человека, который знает, и спросите его. Более правильным вариантом является сотрудничество с администратором брандмауэра в процессе разработки архитектуры DNS, поскольку оно позволяет гарантировать, что созданная структура будет эффективно сосуществовать с брандмауэром.

Заметим, что приводимые сведения о брандмауэрах сети Интернет не являются полными. В нескольких параграфах мы описываем два наиболее распространенных типа брандмауэров, используя минимум подробностей, который необходим, чтобы показать различия в потенциале и влияние на DNS-серверы. Полное руководство по этой теме содержится в книге E. Zwicky, S. Cooper и B. Chapman «Building Internet Firewalls» (O'Reilly).[1]

Пакетные фильтры

Работа первого типа брандмауэров основана на фильтрации пакетов. Брандмауэры, реализующие пакетные фильтры, работают преимущественно на транспортном и сетевом уровнях стека TCP/IP (уровни третий и четвертый эталонной модели OSI, если вам это о чем-то говорит). Решения о маршрутизации пакетов принимаются на основе критериев пакетного уровня, скажем, в зависимости от применяемого транспортного протокола (TCP или UDP), IP-адреса отправителя и получателя, исходного и целевого порта (рис. 11.1).

Наиболее важной особенностью пакетных фильтров является тот факт, что их обычно можно настроить на избирательное пропускание

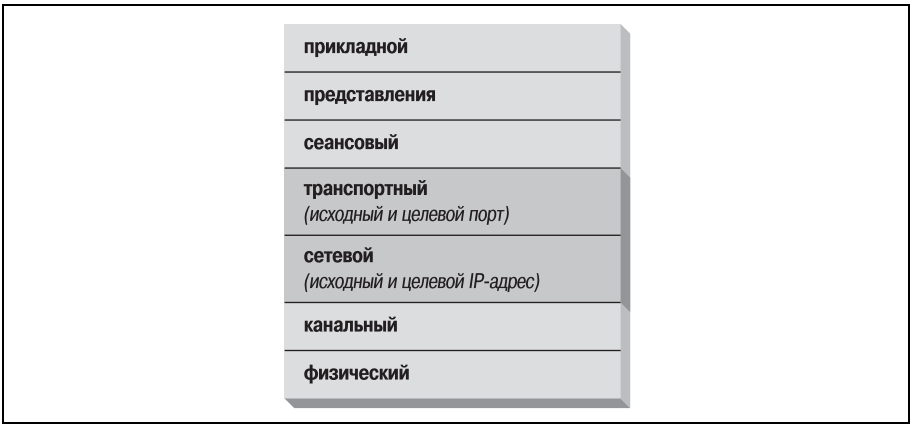

Рис. 11.1. *Пакетные фильтры работают на сетевом и транспортном уровнях стека*

[1] Элизабет Цвики, Саймон Купер и Брент Чапмен «Создание защиты в Интернете», издательство «Символ-Плюс», I кв. 2002 г.

трафика DNS между узлами Интернета и узлами внутренней сети. Иначе говоря, доступ к DNS-серверам Интернета можно ограничить произвольным набором узлов внутренней сети. Некоторые из таких брандмауэров даже предоставляют возможность разрешить DNS-серверам внутренней сети делать запросы ко внешним DNS-серверам (но не наоборот). Все брандмауэры Интернета, созданные на основе маршрутизаторов, используют фильтрацию пакетов. Широко применяются коммерческие брандмауэры с фильтрацией пакетов – FireWall-1 от Checkpoint, PIX от Cisco и SunScreen от Sun.

Хитрости совместного использования BIND 8/9 и брандмауэров с фильтрацией пакетов

Серверы BIND 4 всегда посылают запросы через исходный порт 53, стандартный порт для серверов DNS, и на целевой порт 53. С другой стороны, DNS-клиенты обычно посылают запросы через исходный порт с большим номером (больше 1023) и целевой порт 53. Хотя DNS-серверы должны посылать свои запросы через целевой порт DNS удаленного узла, нет особых причин посылать запросы через исходный порт DNS. И, кто бы мог подумать, DNS-серверы BIND 8 и 9 по умолчанию не посылают запросы через исходный порт с номером 53. Напротив, они посылают запросы через порты с большими номерами, подобно DNS-клиентам.

Это может служить источником проблем при использовании брандмауэров с фильтрацией пакетов, которые настроены пропускать сообщения, посылаемые одним DNS-сервером другому, но не сообщения, посылаемые клиентом DNS-серверу, поскольку в таких случаях брандмауэр ожидает, что сообщение DNS-сервера отправлено через исходный порт с номером 53 и на целевой порт 53.

Существует два решения этой проблемы:

- Перенастроить сетевой экран, разрешив DNS-серверу посылать и принимать запросы через порты, отличные от порта 53 (при условии, что зеленый свет для внешних пакетов, поступающих через старшие порты DNS-серверу, не подвергает опасности собственно брандмауэр).
- Вернуть прежнее поведение BIND, используя предписание *query-source*.

query-source в качестве аргументов принимает адресную спецификацию и необязательный номер порта. К примеру, следующий оператор:

```
options { query-source address * port 53; };
```

предписывает серверу BIND использовать порт 53 в качестве исходного порта для запросов, посылаемых через все локальные сетевые интерфейсы. Для ограничения числа адресов, с которых BIND будет посылать запросы, можно использовать адресную спецификацию без маски. Следующий оператор на узле *wormhole.movie.edu*:

```
options { query-source address 192.249.249.1 port *; };
```

предписывает серверу BIND посылать все запросы с адреса 192.249.249.1 (но не с 192.253.253.1) и использовать динамически изменяемые порты с большими номерами.

query-source с маской в адресной спецификации не работает в BIND 9 до версии 9.1.0, хотя более ранним реализациям BIND 9 можно объяснить, что все запросы следует посылать с определенного адреса через порт 53.

Шлюзы приложений

Шлюзы приложений работают на уровне приложений, несколькими уровнями выше в эталонной модели OSI, чем пакетные фильтры (рис. 11.2). В каком-то смысле они «понимают» прикладной протокол таким же образом, как и сервер для конкретного приложения. К примеру, шлюз приложения FTP может разрешить или запретить определенную операцию FTP, например *RETR* (команда *get*) или *STOR* (команда *put*).

Плохая (и важная для нас) новость заключается в том, что большинство брандмауэров, основанных на шлюзах приложений, работают только с прикладными протоколами на базе TCP. DNS, разумеется, работает преимущественно по UDP, а шлюзов приложений для DNS не су-

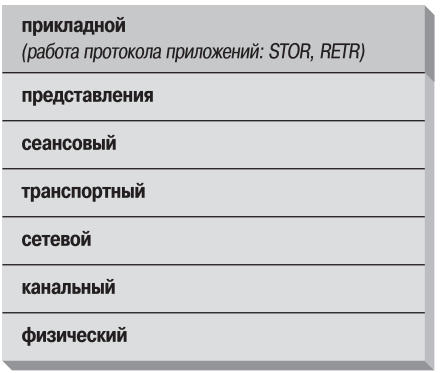

Рис. 11.2. *Шлюзы приложений работают на уровне приложений стека OSI*

ществует. Из этого следует, что в случае использования экранирования шлюзами приложений внутренние узлы скорее всего не смогут напрямую работать с DNS-серверами сети Интернет.

Распространенный Firewall Toolkit (набор инструментов брандмауэра) от Trusted Information Systems (TIS в настоящее время является частью Network Associates) – это набор шлюзов для широко применяемых протоколов Интернета, таких как Telnet, FTP и HTTP. Gauntlet от Network Associates и Eagle Firewall от Axent также основаны на шлюзах приложений.

Следует сказать, что описанные категории брандмауэров – это просто обобщение. Стандарты качества брандмауэров постоянно растут, так что ко времени, когда эта книга попадет в руки читателей, вполне возможно, уже будут существовать шлюзы приложений для DNS. Важно знать, в какое семейство входит применяемый брандмауэр, но лишь потому, что необходимо знать о возможностях экрана; гораздо более важно понять, позволяет ли применяемый брандмауэр обмен DNS-трафиком между произвольными узлами внутренней сети и сетью Интернет.

Плохой пример

Самый простой вариант настройки – разрешить свободное прохождение через брандмауэр трафика DNS (при условии, что брандмауэр может быть настроен таким образом). В этом случае любой внутренний DNS-сервер может посылать запросы любым DNS-серверам сети Интернет, а серверы сети Интернет могут посылать запросы любому из внутренних DNS-серверов. Дополнительная настройка не требуется.

К сожалению, это – по ряду причин – очень плохая идея:

Отслеживание версий

Разработчики пакета BIND постоянно находят и исправляют ошибки, связанные с безопасностью серверов. Как следствие, важно пользоваться более новой версией BIND, в особенности для DNS-серверов, которые выставлены на обозрение всей сети Интернет. Если только один или несколько DNS-серверов общаются с внешними DNS-серверами напрямую, обновить их будет достаточно легко. Если любой из DNS-серверов в сети может непосредственно общаться с внешними серверами, встанет задача обновления версий для *всех* серверов, а это уже гораздо сложнее.

Защищенность

Даже если на определенном узле отсутствует DNS-сервер, взломщик может воспользоваться тем преимуществом, что трафик DNS спокойно проходит через брандмауэр, и произвести атаку на этот узел. К примеру, соучастник заговора может установить на узле демона Telnet, который будет принимать соединения через порт DNS, и взломщик получит возможность проникнуть через *telnet*.

В оставшейся части главы мы постараемся подавать только хороший пример.

Ретрансляторы Интернет

Принимая во внимания опасности, связанные с разрешением неограниченного двунаправленного трафика DNS, большинство организаций ограничивают число внутренних узлов, которые общаются с сетью Интернет на языке DNS. В случае брандмауэра со шлюзами приложений, либо произвольного экрана, который не способен пропускать DNS-трафик, единственным узлом, который может общаться с DNS-серверами Интернета, остается узел-бастион (рис. 11.3).

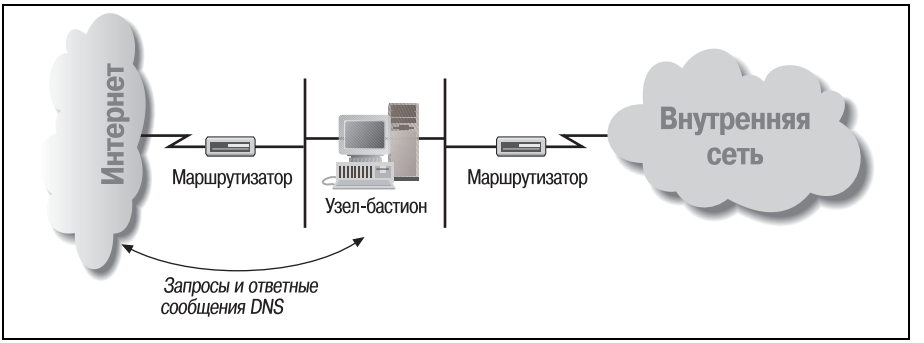

Рис. 11.3. *Небольшая сеть, выставляющая узел-бастион*

Брандмауэр, основанный на фильтрации пакетов, может быть настроен администратором таким образом, что произвольный набор внутренних DNS-серверов сможет взаимодействовать с DNS-серверами Интернета. Обычно это небольшое число узлов, на которых работают DNS-серверы, находящиеся во власти администратора сети (рис. 11.4).

Дополнительная настройка внутренних DNS-серверов, которые могут напрямую посылать запросы внешним DNS-серверам, не требуется. Файлы корневых указателей этих серверов содержат координаты корневых серверов сети Интернет, что позволяет производить разрешение доменных имен сети Интернет. Внутренние серверы имен, которые *не могут* посылать запросы DNS-серверам в Интернет, должны понимать, что неразрешенные запросы следует передавать другим DNS-серверам, которые смогут это сделать. Специально для этих целей существует инструкция или предписание *forwarders*, которое рассматривается в главе 10 «Дополнительные возможности».

На рис. 11.5 представлен распространенный вариант настройки ретрансляции: внутренние DNS-серверы передают запросы DNS-серверу, который работает на узле-бастионе.

Несколько лет назад мы установили в Университете кинематографии брандмауэр, чтобы защитить себя от Большого Злого Интернета. Наш

брандмауэр реализует фильтрацию пакетов, и мы договорились с администратором сетевого экрана, что он разрешит двум нашим DNS-

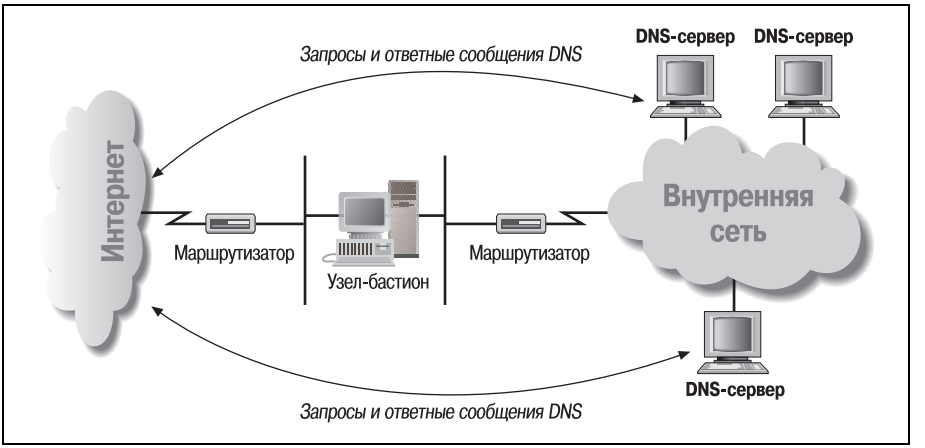

Рис. 11.4. *Небольшая сеть, выставляющая отдельные внутренние DNS-серверы*

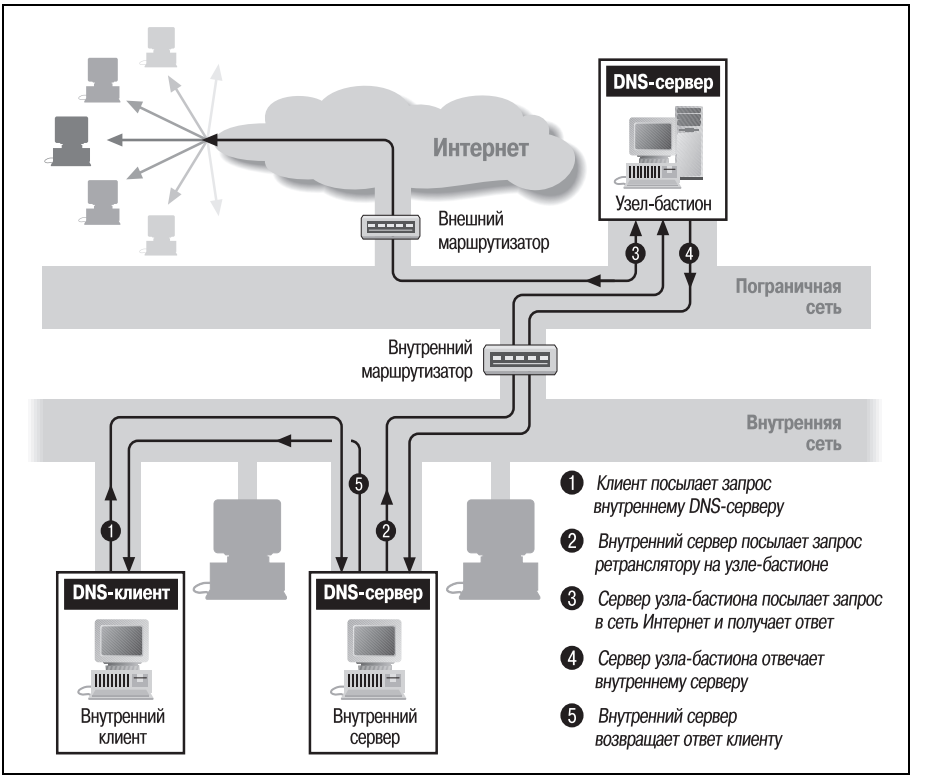

Рис. 11.5. *Применение ретрансляторов*

серверам, *terminator.movie.edu* и *wormhole.movie.edu*, обмениваться DNS-трафиком с DNS-серверами Интернета. Все прочие DNS-серверы университета мы настроили следующим образом. Для серверов BIND 8 и 9 были использованы такие настройки:

```
options {
    forwarders { 192.249.249.1; 192.249.249.3; };
    forward only;
};
```

а для серверов BIND 4 – такие:

```
forwarders 192.249.249.3 192.249.249.1
options forward-only
```

Мы изменяем порядок перечисления ретрансляторов, поскольку это способствует распределению нагрузки между ними. В случае DNS-серверов BIND 8.2.3 в этом нет смысла, поскольку они выбирают ретранслятор, исходя из времени передачи сигнала.

Когда внутренний DNS-сервер получает запрос для имени, разрешение для которого не может произвести локально, к примеру, для доменного имени Интернет, то передает запрос одному из ретрансляторов, которые производят разрешение, посылая запросы DNS-серверам в сети Интернет. Все просто!

Проблемы с ретрансляцией

К сожалению, все не настолько просто. Ретрансляция начинает мешать при делегировании поддоменов или создании крупной сети. Чтобы понять, о чем речь, взгляните на фрагмент файла настройки для *zardoz.movie.edu*:

```
options {
    directory "/var/named";
    forwarders { 192.249.249.1; 192.253.253.3; };
};

zone "movie.edu" {
    type slave;
    masters { 192.249.249.3; };
    file "bak.movie.edu";
};
```

zardoz.movie.edu является вторичным сервером зоны *movie.edu* и использует два ретранслятора. Что случится, если *zardoz.movie.edu* получит запрос для имени в пределах *fx.movie.edu*? Будучи авторитативным сервером зоны *movie.edu*, *zardoz.movie.edu* хранит NS-записи, делегирующие *fx.movie.edu* авторитативным серверам этой зоны. Но *zardoz.movie.edu* также настроен таким образом, что передает все запросы, которые не могут быть разрешены локально, узлам *terminator.movie.edu* и *wormhole.movie.edu*. Какое решение примет DNS-сервер?

Оказывается, *zardoz.movie.edu* игнорирует информацию о делегировании и передает запрос узлу *terminator.movie.edu*. Все работает, поскольку *terminator.movie.edu* получает рекурсивный запрос и передает его по поручению DNS-сервера *zardoz.movie.edu* серверу для *fx.movie.edu*. Но это не очень эффективно, поскольку узел *zardoz.movie.edu* мог бы без проблем послать прямой запрос самостоятельно.

Теперь представим себе сеть гораздо большего масштаба: корпоративные тенета, охватывающие целые континенты, сеть, с десятками тысяч узлов и сотнями или тысячами DNS-серверов. Вне внутренние DNS-серверы, не имеющего прямого подключения к сети Интернет, – то есть, подавляющее большинство – используют небольшую группу ретрансляторов. Что здесь не так?

Низкая надежность

Если ретрансляторы «сломаны», все DNS-серверы теряют способность производить разрешение как доменных имен сети Интернет, так и внутренних доменных имен, которые не кэшированы или не хранятся в виде данных авторитативности.

Концентрация нагрузки

На ретрансляторы ложится невероятно высокая нагрузка. Во-первых, они используются огромным количеством DNS-серверов, а во-вторых, запросы являются рекурсивными, и их выполнение требует определенных затрат времени и сил.

Низкая эффективность разрешения

Представим себе два внутренних DNS-сервера, которые являются авторитативными для зон *west.acmebw.com* и *east.acmebw.com* соответственно; оба сервера расположены в одном сегменте сети, в Боулдер-Сити, штат Колорадо. Оба настроены использовать корпоративный ретранслятор, расположенный в городе Бетесда, штат Мэриленд. DNS-сервер *west.acmebw.com* в целях разрешения доменного имени из *east.acmebw.com* посылает запрос ретранслятору в Бетесде. Ретранслятор в Бетесде возвращает запрос в Боулдер-Сити, DNS-серверу *east.acmebw.com*, который является соседом узла, сделавшего первоначальный запрос. DNS-сервер *east.acmebw.com* отправляет ответ в Бетесду, а оттуда ретранслятор посылает его снова в Боулдер-Сити.

В традиционном варианте настройки, когда используются корневые DNS-серверы, DNS-сервер *west.acmebw.com* быстро узнал бы, что DNS-сервер *east.acmebw.com* стоит в соседней комнате, и отдал бы ему предпочтение (из-за меньшего времени передачи сигнала). Использование ретрансляторов приводит к «короткому замыканию» в обычно эффективном процессе разрешения.

Вывод следующий: использование ретрансляторов вполне оправданно для небольших сетей и простых пространств имен, но скорее всего неадекватно для крупных сетей и развесистых пространств имен. По ме-

ре роста сети Университета кинематографии мы узнали об этом на собственном опыте и были вынуждены искать альтернативу.

Зоны ретрансляции

Проблемы можно решить с помощью зон ретрансляции, появившихся в BIND 8.2. Изменим настройки *zardoz.movie.edu* следующим образом:

```
options {
    directory "/var/named";
    forwarders { 192.249.249.1; 192.253.253.3; };
};

zone "movie.edu" {
    type slave;
    masters { 192.249.249.3; };
    file "bak.movie.edu";
    forwarders {};
};
```

Теперь, если *zardoz.movie.edu* получает запрос для доменного имени, заканчивающегося на *movie.edu*, но расположенного вне зоны *movie.edu* (например, в зоне *fx.movie.edu*), то игнорирует ретрансляторы и посылает итеративные запросы.

Но при этом *zardoz.movie.edu* по-прежнему посылает ретрансляторам запросы для доменных имен из зон обратного отображения. Чтобы уменьшить нагрузку, можно добавить несколько операторов *zone* в файл *named.conf*:

```
zone "249.249.192.in-addr.arpa" {
    type stub;
    masters { 192.249.249.3; };
    file "stub.192.249.249";
    forwarders {};
};

zone "253.253.192.in-addr.arpa" {
    type stub;
    masters { 192.249.249.3; };
    file "stub.192.253.253";
    forwarders {};
};

zone "254.253.192.in-addr.arpa" {
    type stub;
    masters { 192.253.254.2; };
    file "stub.192.253.254";
    forwarders {};
};

zone "20.254.192.in-addr.arpa" {
    type stub;
    masters { 192.253.254.2; };
```

```
        file "stub.192.254.20";
        forwarders {};
    };
```

Следует прокомментировать новые операторы *zone*: прежде всего, зоны обратного отображения Университета кинематографии теперь являются зонами-заглушками. Это значит, что DNS-сервер отслеживает NS-записи, периодически опрашивая основные DNS-серверы этих зон. Инструкция *forwarders* выключает ретрансляцию для доменных имен в доменах обратного отображения. Теперь, вместо того чтобы передавать ретрансляторам запрос PTR-записи для *2.254.253.192.in-addr.arpa*, *zardoz.movie.edu* пошлет прямой запрос одному из серверов *254.253.192.in-addr.arpa*.

Подобные операторы *zone* понадобятся для всех наших внутренних DNS-серверов, а это значит, что для всех DNS-серверов придется использовать версию BIND более позднюю, чем 8.2.[1]

Мы получаем достаточно надежную структуру для разрешения имен, которая минимизирует взаимодействие с сетью Интернет: посредством использования эффективного и надежного итеративного разрешения имен для внутренних доменных имен, а ретрансляции – только в случае необходимости произвести разрешение доменного имени сети Интернет. Если ретрансляторы не работают, либо разорвано подключение к сети Интернет, мы утрачиваем способность производить разрешение для доменных имен сети Интернет.

Внутренние корневые серверы

Чтобы избежать проблем с масштабированием при использовании ретрансляции, можно создать собственные корневые DNS-серверы. Внутренние корневые серверы будут обслуживать только DNS-серверы нашей организации. Они будут знать только о сегментах пространства имен, имеющих к ней непосредственное отношение.

Какой от них толк? Используя архитектуру, основанную на корневых DNS-серверах, мы получаем масштабируемость пространства имен сети Интернет (для большинства случаев ее должно хватать) плюс избыточность, распределение нагрузки, эффективное разрешение имен. Внутренних корневых DNS-серверов может быть столько же, сколько в сети Интернет – 13 или около того – в то время как подобное количество ретрансляторов будет представлять ненужную угрозу безопасности и приведет к излишним сложностям в настройке. Самое главное, внутренние корневые DNS-серверы не используются легкомысленно. Необходимость в консультации с внутренним корневым сервером у обычного DNS-сервера возникает только в том случае, когда

[1] BIND 9, как мы говорили в предыдущей главе, не поддерживает зоны ретрансляции до версии BIND 9.1.0.

истекает интервал хранения NS-записей для внутренних зон высшего уровня. Если используются ретрансляторы, они могут получать до одного запроса на *каждый* запрос разрешения имени, посылаемый обычным DNS-серверам.

Мораль этой сказки такова: если существует или планируется разветвистое пространство имен и большое число внутренних DNS-серверов, следует помнить, что внутренние корневые серверы масштабируются лучше, чем любое другое решение.

Где располагать внутренние корневые DNS-серверы

Поскольку DNS-серверы имеют привычку «фиксироваться» на наиболее близко расположенных корневых серверах, предпочитая серверы с минимальным временем передачи сигнала, добавление внутренних корневых DNS-серверов окупается. Если сеть организации охватывает США, Европу и побережье Тихого океана, следует иметь по меньшей мере один внутренний корневой DNS-сервер на каждом континенте. Если в Европе существует три крупных комплекса, следует создать в пределах каждого внутренний корневой DNS-сервер.

Делегирование для прямого отображения

Сейчас мы опишем настройку внутреннего корневого сервера. Внутренний корневой сервер производит прямое делегирование всем администрируемым зонам. К примеру, в сети *movie.edu* файл данных корневого сервера будет содержать следующие записи:

```
movie.edu.  86400  IN  NS  terminator.movie.edu.
            86400  IN  NS  wormhole.movie.edu.
            86400  IN  NS  zardoz.movie.edu.
terminator.movie.edu.  86400  IN  A  192.249.249.3
wormhole.movie.edu.    86400  IN  A  192.249.249.1
                       86400  IN  A  192.253.253.1
zardoz.movie.edu.      86400  IN  A  192.249.249.9
                       86400  IN  A  192.253.253.9
```

В сети Интернет эта информация отображалась бы в файлах данных серверов зоны *edu*. Понятно, что в сети *movie.edu* нет DNS-серверов *edu*, поэтому происходит прямое делегирование *movie.edu* от корня.

Обратите внимание, что записи не содержат делегирования для *fx.movie.edu* и всех остальных поддоменов *movie.edu*. DNS-серверы *movie.edu* в курсе, какие DNS-серверы являются авторитативными для каждого из поддоменов *movie.edu*, и все запросы, касающиеся информации из этих поддоменов, проходят через DNS-серверы *movie.edu*, поэтому (конкретно здесь) нет необходимости в делегировании.

Делегирование in-addr.arpa

Необходимо также произвести делегирование внутренними корневыми серверами для зон *in-addr.arpa*, которые соответствуют университетским сетям:

```
249.249.192.in-addr.arpa.  86400  IN  NS  terminator.movie.edu.
                           86400  IN  NS  wormhole.movie.edu.
                           86400  IN  NS  zardoz.movie.edu.
253.253.192.in-addr.arpa.  86400  IN  NS  terminator.movie.edu.
                           86400  IN  NS  wormhole.movie.edu.
                           86400  IN  NS  zardoz.movie.edu.
254.253.192.in-addr.arpa.  86400  IN  NS  bladerunner.fx.movie.edu.
                           86400  IN  NS  outland.fx.movie.edu.
                           86400  IN  NS  alien.fx.movie.edu.
20.254.192.in-addr.arpa.   86400  IN  NS  bladerunner.fx.movie.edu.
                           86400  IN  NS  outland.fx.movie.edu.
                           86400  IN  NS  alien.fx.movie.edu.
```

Обратите внимание, что мы *включили* делегирование для зон *254.253.192.in-addr.arpa* и *20.254.192.in-addr.arpa*, несмотря на то, что они соответствуют зоне *fx.movie.edu*. Необходимости делегировать *fx.movie.edu* нет, потому что делегирование уже произведено для родительской зоны, *movie.edu*. Серверы *movie.edu* делегируют полномочия *fx.movie.edu*, так что, исходя из принципа транзитивности, корневые серверы также делегируют полномочия *fx.movie.edu*. При этом ни одна из зон *in-addr.arpa* не является родительской для *254.253.192.in-addr.arpa* или *20.254.192.in-addr.arpa*, а значит, необходимо производить делегирование обеих зон от корня. Как говорилось ранее, нет необходимости добавлять адресные записи для трех DNS-серверов факультета Special Effects, *bladerunner.fx.movie.edu*, *outland.fx.movie.edu* и *alien.fx.movie.edu*, поскольку удаленный DNS-сервер может и без этого найти их адреса, следуя за делегированием от *movie.edu*.

Файл db.root

Осталось лишь добавить SOA-запись для корневой зоны и NS-записи для этого и всех остальных внутренних корневых DNS-серверов:

```
$TTL 1d
.  IN  SOA  rainman.movie.edu.  hostmaster.movie.edu.  (
            1    ; порядковый номер
            3h   ; обновление
            1h   ; повторение
            1w   ; устаревание
            1h ) ; отрицательное TTL

   IN  NS  rainman.movie.edu.
   IN  NS  awakenings.movie.edu.

rainman.movie.edu.    IN  A  192.249.249.254
awakenings.movie.edu. IN  A  192.253.253.254
```

Внутренние корневые DNS-серверы работают на узлах *rainman.movie.edu* и *awakenings.movie.edu*. Не следует устанавливать корневой сервер на узел-бастион, поскольку если внешний DNS-сервер случайно запросит у этого сервера данные, для которых он не является авторитативным, то получит в ответ перечень корневых DNS-серверов – причем внутренних!

Поэтому полный файл *db.root* (по договоренности, мы называем файл данных корневой зоны именем *db.root*) выглядит так:

```
$TTL 1d
.  IN  SOA  rainman.movie.edu.  hostmaster.movie.edu.  (
            1    ; порядковый номер
            3h   ; обновление
            1h   ; повторение
            1w   ; устаревание
            1h ) ; отрицательное TTL

   IN  NS  rainman.movie.edu.
   IN  NS  awakenings.movie.edu.

rainman.movie.edu.    IN  A  192.249.249.254
awakenings.movie.edu. IN  A  192.253.253.254

movie.edu.  IN  NS  terminator.movie.edu.
            IN  NS  wormhole.movie.edu.
            IN  NS  zardoz.movie.edu.

terminator.movie.edu.  IN  A  192.249.249.3
wormhole.movie.edu.    IN  A  192.249.249.1
                       IN  A  192.253.253.1
zardoz.movie.edu.      IN  A  192.249.249.9
                       IN  A  192.253.253.9

249.249.192.in-addr.arpa.  IN  NS  terminator.movie.edu.
                           IN  NS  wormhole.movie.edu.
                           IN  NS  zardoz.movie.edu.
253.253.192.in-addr.arpa.  IN  NS  terminator.movie.edu.
                           IN  NS  wormhole.movie.edu.
                           IN  NS  zardoz.movie.edu.
254.253.192.in-addr.arpa.  IN  NS  bladerunner.fx.movie.edu.
                           IN  NS  outland.fx.movie.edu.
                           IN  NS  alien.fx.movie.edu.
20.254.192.in-addr.arpa.   IN  NS  bladerunner.fx.movie.edu.
                           IN  NS  outland.fx.movie.edu.
                           IN  NS  alien.fx.movie.edu.
```

Файл *named.conf* на узлах *rainman.movie.edu* и *awakenings.movie.edu* содержит следующие строки:

```
zone "." {
    type master;
    file "db.root";
};
```

Либо для сервера BIND 4 и файла *named.boot*:

```
primary    .    db.root
```

Они заменяют оператор *zone* типа *hint* или инструкцию *cache* – корневому DNS-серверу не нужен файл корневых указателей, чтобы узнать координаты других корневых серверов, поскольку они и без того содержатся в файле *db.root*. Означает ли это, что каждый корневой DNS-сервер является первичным мастером для корневой зоны? Нет, только в случае использования древней версии BIND. Все версии BIND, более поздние, чем 4.9, позволяют объявить DNS-сервер вторичным для корневой зоны, в то время как BIND 4.8.3 и более ранних версий настаивает на том, что каждый корневой DNS-сервер должен являться первичным для корневой зоны.

Если в сети не так уж много свободных узлов, из которых можно сделать внутренние корневые DNS-серверы, не все потеряно! Каждый внутренний DNS-сервер (DNS-сервер внутренней сети, работающий не на узле-бастионе) может быть *одновременно* и внутренним корневым DNS-сервером, и авторитативным DNS-сервером для всех прочих загружаемых зон. Помните, что единственный DNS-сервер может быть авторитативным для большого числа зон, включая и корневую.

Настройка прочих внутренних DNS-серверов

Создав внутренние корневые DNS-серверы, следует настроить DNS-серверы, работающие на узлах внутренней сети, на использование новых корневых серверов. Любой сервер, работающий на узле без прямого подключения к сети Интернет (то есть за сетевым экраном), должен иметь перечень внутренних корневых серверов в файле корневых указателей:

```
; Файл корневых указателей внутренних серверов для узлов
; Университета кинематографии, не имеющих прямого подключения к сети Интернет
;
; Не используйте этот файл на узле с прямым подключением к сети Интернет!
;

.  99999999  IN  NS  rainman.movie.edu.
   99999999  IN  NS  awakenings.movie.edu.

rainman.movie.edu.     99999999  IN  A  192.249.249.254
awakenings.movie.edu.  99999999  IN  A  192.253.253.254
```

DNS-серверы, работающие на узлах с подобным файлом корневых указателей, смогут производить разрешение доменных имен зоны *movie.edu* и доменах *in-addr.arpa* Университета кинематографии, но не имен за пределами этих доменов.

Использование внутренних корневых серверов внутренними DNS-серверами

Чтобы увязать схему воедино, рассмотрим пример разрешения на внутреннем DNS-сервере, который специализируется на кэшировании и обладает информацией о внутренних корневых DNS-серверах. Внутренний DNS-сервер получает запрос для доменного имени из зоны *movie.edu*, скажем запрос адреса для имени *gump.fx.movie.edu*. Если внутренний DNS-сервер не может ответить кэшированной информацией, то посылает запрос одному из внутренних корневых DNS-серверов. Если этот сервер уже контактировал с внутренними корневыми DNS-серверами, то запомнил время передачи сигнала для каждого из них и теперь имеет возможность выбрать корневой сервер с наименьшим временем реагирования. Этому корневому серверу посылается нерекурсивный запрос адреса для имени *gump.fx.movie.edu*. Внутренний корневой сервер отвечает ссылками на DNS-серверы зоны *movie.edu* – *terminator.movie.edu*, *wormhole.movie.edu* и *zardoz.movie.edu*. Кэширующий DNS-сервер продолжает поиск, посылая следующий нерекурсивный запрос адреса *gump.fx.movie.edu* одному из DNS-серверов *movie.edu*. DNS-сервер *movie.edu* возвращает ссылки на DNS-серверы зоны *fx.movie.edu*. Кэширующий DNS-сервер посылает тот же нерекурсивный запрос адреса *gump.fx.movie.edu* одному из DNS-серверов *fx.movie.edu* и, наконец, получает ответ.

Сравним такую настройку с работой варианта ретрансляции. Допустим, наш DNS-сервер, специализирующийся на кэшировании, настроен не на использование внутренних корневых DNS-серверов, а на передачу запросов сначала узлу *terminator.movie.edu*, а затем *wormhole.movie.edu*. В этом случае наш сервер производит поиск адреса *gump.fx.movie.edu* в кэше и, не найдя его, передает запрос серверу *terminator.movie.edu*. Затем *terminator.movie.edu* посылает запрос к одному из DNS-серверов *fx.movie.edu* от имени кэширующего DNS-сервера и возвращает полученный ответ. Когда кэширующий DNS-сервер в следующий раз будет производить поиск для имени из зоны *fx.movie.edu*, то обратится к ретранслятору точно таким же образом, несмотря на то, что ответ на предыдущий запрос (адреса для *gump.fx.movie.edu*) скорее всего содержал имена и адреса DNS-серверов зоны *fx.movie.edu*.

Отправка почтовых сообщений в сеть Интернет

Минутку! Это еще не все, что на что способны внутренние корневые DNS-серверы. Мы говорили об отправке почтовых сообщений в сеть Интернет, которая не связана с необходимостью изменять настройки программы *sendmail* по всей сети.

Заставить работать почту можно с помощью масок в записях, а именно масок в MX-записях. Допустим, необходимо сделать так, чтобы почта, отправляемая в сеть Интернет, передавалась через узел *postmanrings2x.movie.edu*, узел-бастион Университета кинематографии, кото-

рый напрямую подключен к сети Интернет. Добиться нужного результата можно, добавив следующие записи в файл *db.root*:

```
*        IN    MX    5 postmanrings2x.movie.edu.
*.edu.   IN    MX    10 postmanrings2x.movie.edu.
```

MX-запись с маской **.edu* нужна в дополнение к записи с маской * из-за существующих для масок правил вывода, о которых можно подробнее прочитать в разделе «Маски» главы 16 «Обо всем понемногу». По существу, поскольку для зоны *movie.edu* существуют явно определенные данные, первая маска не приведет к сопоставлению *movie.edu* или любого другого поддомена зоны *edu*. Нужна еще одна, явно определенная запись с маской для *edu*, с которой будут сопоставляться поддомены *edu* помимо *movie.edu*.

Теперь почтовые программы, работающие на внутренних узлах зоны *movie.edu*, будут отправлять почту, адресованную доменным именам сети Интернет, узлу *postmanrings2x.movie.edu* для ретрансляции. К примеру, почтовые сообщения, адресованные домену *nic.ddn.mil*, будут сопоставлены с первой MX-записью, включающей маску:

```
% nslookup -type=mx nic.ddn.mil.   — Сопоставляется с MX-записью для
                                     имени *
Server:  rainman.movie.edu
Address:  192.249.249.19

nic.ddn.mil
     preference = 5, mail exchanger = postmanrings2x.movie.edu
postmanrings2x.movie.edu    internet address = 192.249.249.20
```

Почтовые сообщения, адресованные домену *vangogh.cs.berkeley.edu*, будут сопоставлены со второй MX-записью:

```
% nslookup -type=mx vangogh.cs.berkeley.edu. — Сопоставляется с
                                               MX-записью для имени *.edu
Server:  rainman.movie.edu
Address:  192.249.249.19

vangogh.cs.berkeley.edu
     preference = 10, mail exchanger = postmanrings2x.movie.edu
postmanrings2x.movie.edu    internet address = 192.249.249.20
```

Когда почтовые сообщения появятся на *postmanrings2x.movie.edu*, узле-бастионе, почтовая программа *postmanrings2x.movie.edu* произведет поиск MX-записей для конечных адресов. Поскольку узел *postmanrings2x.movie.edu* будет производить разрешение доменных имен, используя пространство имен сети Интернет, а не пространство имен внутренней сети, будут найдены реальные MX-записи, и почта будет доставлена адресатам. При этом нет необходимости изменять настройки *sendmail*.

Отправка почтовых сообщений для конкретных доменных имен сети Интернет

У схемы, построенной на основе внутренних корневых DNS-серверов, есть еще один плюс: она позволяет передавать почту, предназначенную различным доменам сети Интернет, через различные узлы-бастионы в случае, когда таких узлов несколько. К примеру, мы можем захотеть посылать почту, адресованную узлам в домене *uk*, нашему узлу-бастиону, расположенному в Лондоне, который затем будет передавать почтовые сообщения в Интернет. Это может быть очень удобно, если мы хотим, чтобы почта ходила в сети максимально быстро, или же желаем сократить расходы, связанные с использованием определенной сети Великобритании.

Университет кинематографии связан собственным сетевым каналом с университетом-побратимом, который расположен в Лондоне, неподалеку от киностудии Пайнвуд. Из соображений безопасности мы хотим, чтобы почта для адресатов в Соединенном Королевстве, посылалась через этот сетевой канал, и, затем, отправлялась дальше через узел студии Пайнвуд. Мы добавляем следующие записи с масками в файл *db.root*:

```
; holygrail.movie.ac.uk - по другую сторону Интернет-канала
                          в Соединенное Королество
*.uk.    IN    MX    10 holygrail.movie.ac.uk.
holygrail.movie.ac.uk.    IN   A    192.168.76.4
```

Теперь почта, адресованная пользователям в поддоменах домена *uk* будет отправляться узлу *holygrail.movie.ac.uk*, принадлежащему сети университета-побратима, в которой – как мы полагаем – существует возможность передать почтовые сообщения во все точки страны.

Проблемы внутренних корневых DNS-серверов

К сожалению, не только у ретрансляции есть проблемы: архитектура на основе внутренних DNS-серверов также имеет ограничения. Самое большое из них – внутренние узлы не «видят» пространство имен сети Интернет. В некоторых сетях это не проблема, поскольку большинство внутренних узлов не имеют прямого подключения к сети Интернет. А на тех немногих узлах, что связаны с сетью Интернет напрямую, DNS-клиенты настроены на использование DNS-сервера узла-бастиона. На некоторых из этих узлов, вероятно, работают серверы-посредники (proxy), позволяющие прочим внутренним узлам получать доступ к службам сети Интернет.

Однако в других сетях используемый сетевой экран или другое программное обеспечение могут требовать от внутренних узлов способности разрешать имена из пространства имен сети Интернет. В таких сетях архитектура, основанная на внутренних корневых DNS-серверах, работать не будет.

Расщепление пространства имен

Многие организации желали бы иметь возможность объявлять сети Интернет совсем не те зональные данные, что доступны внутренним узлам. В большинстве случаев, большая часть внутренних зональных данных не относится к сети Интернет, поскольку в организации используется брандмауэр. Такой сетевой экран может запрещать прямой доступ к большинству внутренних узлов, а также производить отображение внутренних, незарегистрированных IP-адресов во множество IP-адресов, закрепленных за организацией. Таким образом, в организации может существовать необходимость удаления излишней информации из внешнего представления зоны, либо преобразования внутренних адресов в их видимые извне эквиваленты.

Но в BIND не реализована автоматическая фильтрация и преобразования зональных данных. Поэтому многие организации вручную создают структуры, которые получили название «расщепленных пространств имен». В расщепленном пространстве имен каноническое пространство имен доступно только в пределах внутренней сети, а сокращенная, преобразованная версия этого пространства, получившая название *затененного пространства имен*, доступна извне – из сети Интернет.

Затененное пространство имен содержит отображения имен в адреса и адресов в имена только для узлов, доступных из сети Интернет через брандмауэр. Видимые извне адреса могут являться преобразованными эквивалентами внутренних адресов. Затененное пространство имен может также содержать одну или несколько записей, позволяющих направлять почтовые сообщения, поступающие из сети Интернет, через брандмауэр – почтовому серверу.

Поскольку в Университете кинематографии используется брандмауэр Интернет, который в значительной степени ограничивает доступ ко внутренней сети извне, мы приняли решение создать затененное пространство имен. Если речь идет о зоне *movie.edu*, то мы должны предоставить только информацию о доменном имени *movie.edu* (SOA-запись и несколько NS-записей), об узле-бастионе (*postmanrings2x.movie.edu*) и о нашем новом внешнем DNS-сервере *ns.movie.edu*, который по совместительству является внешним веб-сервером *www.movie.edu*. Адрес внешнего интерфейса на узле-бастионе – 200.1.4.2, адрес веб-сервера/DNS-сервера – 200.1.4.3. Файл данных для затененной зоны *movie.edu* выглядит следующим образом:

```
$TTL 1d
@    IN    SOA    ns.movie.edu.    hostmaster.movie.edu. (
                                 1    ; Порядковый номер
                                 3h   ; Обновление
                                 1h   ; Повторение
                                 1w   ; Устаревание
                                 1h ) ; Отрицательное TTL
```

```
        IN    NS    ns.movie.edu.
        IN    NS    ns1.isp.net.       ; DNS-сервер нашего интернет-провайдера
                                       ; является дополнительным для зоны movie.edu

        IN    A     200.1.4.3
        IN    MX    10 postmanrings2x.movie.edu.
        IN    MX    100 mail.isp.net.

www             IN    CNAME movie.edu.

postmanrings2x  IN    A     200.1.4.2
                IN    MX    10 postmanrings2x.movie.edu.
                IN    MX    100 mail.isp.net.

;postmanrings2x.movie.edu обрабатывает почту, адресованную ns.movie.edu
ns              IN    A     200.1.4.3
                IN    MX    10 postmanrings2x.movie.edu.
                IN    MX    100 mail.isp.net.

*               IN    MX    10 postmanrings2x.movie.edu.
                IN    MX    100 mail.isp.net.
```

Обратите внимание, здесь не упомянут ни один из поддоменов зоны *movie.edu*, и нет никакой информации о делегировании DNS-серверам этих поддоменов. В этой информации просто нет необходимости, поскольку ресурсы этих поддоменов не могут быть использованы из сети Интернет, а входящая почта, адресованная узлам в этих поддоменах, сопоставляется с маской.

Файл *db.200.1.4*, который нужен для обратного отображения двух IP-адресов Университета кинематографии, видимых узлам из сети Интернет, выглядит так:

```
$TTL 1d
@    IN    SOA    ns.movie.edu.    hostmaster.movie.edu. (
                          1    ; Порядковый номер
                          3h   ; Обновление
                          1h   ; Повторение
                          1w   ; Устаревание
                          1h ) ; Отрицательное TTL

     IN    NS    ns.movie.edu.
     IN    NS    ns.isp.net.

2    IN    PTR    postmanrings2x.movie.edu.
3    IN    PTR    ns1.movie.edu.
```

Следует проявить осторожность и убедиться, что DNS-клиент узла-бастиона не настроен на использование DNS-сервера *ns.movie.edu*. Поскольку этот сервер не может видеть реально существующее для зоны *movie.edu* пространство имен, его использование приведет к тому, что узел *postmanrings2x.movie.edu* утратит способность производить отображение внутренних доменных имен в адреса и внутренних адресов в имена.

Настройка узла-бастиона

Узел-бастион представляет особый случай в схеме расщепленного пространства имен. Он работает одновременно в двух мирах: один сетевой интерфейс соединяет его с сетью Интернет, а второй – с внутренней сетью. Мы произвели расщепление пространства имен на два: как может узел-бастион видеть и пространство имен сети Интернет, и каноническое внутреннее пространство имен? Если при настройке этого узла поместить в файл корневых указателей координаты корневых серверов сети Интернет, узел-бастион будет использовать информацию о делегировании, полученную от DNS-серверов зоны *edu*, для доступа к внешнему DNS-серверу *movie.edu*, обладающему затененными зональными данными. Узел-бастион будет слеп во всем, что касается внутреннего пространства имен, которое ему как раз необходимо видеть – чтобы регистрировать соединения, доставлять входящую почту, и выполнять другие рутинные задачи. С другой стороны, если настроить этот узел с использованием внутренних корневых DNS-серверов, он не будет видеть пространства имен сети Интернет, которое необходимо видеть, чтобы могли выполняться функции узла-бастиона. Как быть?

Если бы у нас были внутренние DNS-серверы, умеющие производить разрешение как внутренних имен, так и доменных имен сети Интернет – как в приведенном выше примере с использованием зон ретрансляции, – то мы могли бы просто настроить DNS-клиент узла-бастиона на работу с этими DNS-серверами. Но если мы используем ретрансляцию внутри сети, то в зависимости от типа брандмауэра может понадобиться работа ретранслятора и на самом узле-бастионе. Если брандмауэр не пропускает трафик DNS, понадобится по меньшей мере специальный кэширующий DNS-сервер, настроенный на использование корневых DNS-серверов сети Интернет и работающий на узле-бастионе. В этом случае он будет использоваться внутренними DNS-серверами в качестве ретранслятора для запросов, которые они не могут разрешить самостоятельно.

Если внутренние DNS-серверы не поддерживают работу с зонами ретрансляции, DNS-сервер на узле-бастионе следует настроить в качестве вторичного для зоны *movie.edu* и всех зон *in-addr.arpa*, для которых он будет производить разрешение адресов. В этом случае, если сервер узла-бастиона получает запрос для доменного имени из *movie.edu*, он возьмет локальные авторитативные данные для разрешения имени. (Если внутренние DNS-серверы поддерживают зоны ретрансляции, DNS-сервер узла-бастиона никогда не будет получать запросов для имен из *movie.edu*.) Если доменное имя находится в делегированном поддомене *movie.edu*, DNS-сервер использует NS-записи зоны и отправляет запрос для имени внутреннему DNS-серверу. То есть нет необходимости настраивать этот сервер в качестве вторичного для какого-либо из поддоменов *movie.edu* (скажем, *fx.movie.edu*), а только для «самой верхней» зоны (рис. 11.6).

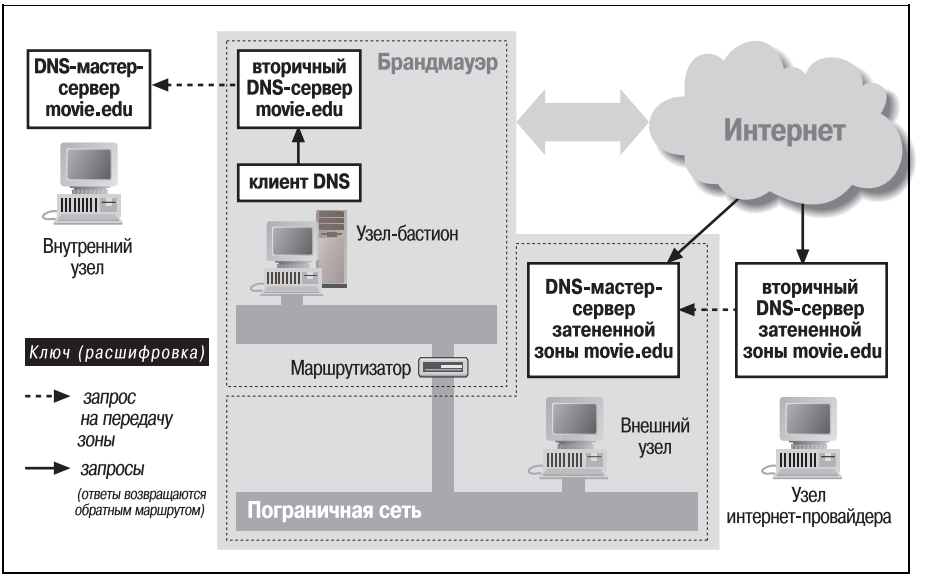

Рис. 11.6. *Расщепленная архитектура DNS*

Файл *named.conf* на нашем узле-бастионе может выглядеть следующим образом:

```
options {
    directory "/var/named";
};

zone "movie.edu" {
    type slave;
    masters { 192.249.249.3; };
    file "bak.movie.edu";
};

zone "249.249.192.in-addr.arpa" {
    type slave;
    masters { 192.249.249.3; };
    file "bak.192.249.249";
};

zone "253.253.192.in-addr.arpa" {
    type slave;
    masters { 192.249.249.3; };
    file "bak.192.253.253";
};

zone "254.253.192.in-addr.arpa" {
    type slave;
    masters { 192.253.254.2; };
    file "bak.192.253.254";
};
```

```
zone "20.254.192.in-addr.arpa" {
    type slave;
    masters { 192.253.254.2; };
    file "bak.192.254.20";
};

zone "." {
    type hint;
    file "db.cache";
};
```

Эквивалентный файл *named.boot* выглядит так:

```
directory    /var/named
secondary    movie.edu    192.249.249.3   bak.movie.edu
secondary    249.249.192.in-addr.arpa    192.249.249.3    bak.192.249.249
secondary    253.253.192.in-addr.arpa    192.249.249.3    bak.192.253.253
secondary    254.253.192.in-addr.arpa    192.253.254.2    bak.192.253.254
secondary    20.254.192.in-addr.arpa     192.253.254.2    bak.192.254.20
cache    .    db.cache    ; перечень корневых
                          ; DNS-серверов Интернет
```

Защита зональных данных узла-бастиона

К сожалению, загрузка этих зон на узле-бастионе делает их данные потенциально доступными узлам сети Интернет, а именно этого мы и пытались избежать, расщепляя пространство имен. Если мы используем DNS-сервер BIND версии 4.9 или более поздней, то всегда способны защитить зональные данные с помощью TXT-записи *secure_zone* или предписания *allow-query* (оба варианта рассмотрены ранее в тексте главы). С помощью *allow-query* мы можем связать глобальный список управления доступом с данными зоны. Вот новый оператор *options* из файла *named.conf*:

```
options {
    directory "/var/named";
    allow-query { 127/8; 192.249.249/24; 192.253.253/24;
        192.253.254/24; 192.254.20/24; };
};
```

Используя механизм BIND 4.9 *secure_zone*, мы можем запретить внешний доступ к данным зоны, включив следующие TXT-записи в каждый файл данных:

```
secure_zone     IN    TXT    "192.249.249.0:255.255.255.0"
                IN    TXT    "192.253.253.0:255.255.255.0"
                IN    TXT    "192.253.254.0:255.255.255.0"
                IN    TXT    "192.254.20.0:255.255.255.0"
                IN    TXT    "127.0.0.1:H"
```

Адрес loopback-интерфейса должен обязательно присутствовать в списке, иначе клиент узла-бастиона может столкнуться с трудностями при попытке получить ответ от локального DNS-сервера!

Окончательная настройка

В завершение необходимо принять прочие меры, которые мы обсуждали ранее, для обеспечения безопасности DNS-сервера узла-бастиона. В частности, необходимо:

- Ограничить передачу зон.
- Использовать предписание *use-id-pool* (BIND 8.2 и последующие версии, но не BIND 9).
- (Необязательно) Выполнять BIND в *chroot*-среде с наименьшими полномочиями.

В конце концов, наш файл *named.conf* принял следующий вид:

```
acl "internal" {
    127/8; 192.249.249/24; 192.253.253/24;
    192.253.254/24; 192.254.20/24;
};

options {
    directory "/var/named";
    allow-query { "internal"; };
    allow-transfer { none; };
    use-id-pool yes;
};

zone "movie.edu" {
    type slave;
    masters { 192.249.249.3; };
    file "bak.movie.edu";
};

zone "249.249.192.in-addr.arpa" {
    type slave;
    masters { 192.249.249.3; };
    file "bak.192.249.249";
};

zone "253.253.192.in-addr.arpa" {
    type slave;
    masters { 192.249.249.3; };
    file "bak.192.253.253";
};

zone "254.253.192.in-addr.arpa" {
    type slave;
    masters { 192.253.254.2; };
    file "bak.192.253.254";
};

zone "20.254.192.in-addr.arpa" {
    type slave;
    masters { 192.253.254.2; };
    file "bak.192.254.20";
};
```

```
zone "." {
    type hint;
    file "db.cache";
};
```

Узел-бастион и виды

Если бы на узле-бастионе использовался DNS-сервер BIND 9, мы могли бы применить виды – для безопасного представления затененной зоны *movie.edu* внешнему миру на том же сервере, который производит разрешение доменных имен Интернета. Это может устранить необходимость работы внешнего DNS-сервера на том же узле, на котором работает наш веб-сервер, *www.movie.edu*. В противном случае у нас будет два DNS-сервера для обслуживания видимой извне зоны *movie.edu*.

Эти настройки очень похожи на приводимые в разделе «Виды» главы 10:

```
options {
    directory "/var/named";
};

acl "internal" {
    127/8; 192.249.249/24; 192.253.253/24; 192.253.254/24; 192.254.20/24;
};

view "internal" {
    match-clients { "internal"; };
    recursion yes;

    zone "movie.edu" {
        type slave;
        masters { 192.249.249.3; };
        file "bak.movie.edu";
     };

    zone "249.249.192.in-addr.arpa" {
        type slave;
        masters { 192.249.249.3; };
        file "bak.192.249.249";
    };

    zone "253.253.192.in-addr.arpa" {
        type slave;
        masters { 192.249.249.3; };
        file "bak.192.253.253";
    };

    zone "254.253.192.in-addr.arpa" {
        type slave;
        masters { 192.253.254.2; };
        file "bak.192.253.254";
    };

    zone "20.254.192.in-addr.arpa" {
```

```
        type slave;
        masters { 192.253.254.2; };
        file "bak.192.254.20";
    };

    zone "." {
        type hint;
        file "db.cache";
    };
};

view "external" {
    match-clients { any; };
    recursion no;

    acl "ns1.isp.net" { 199.11.28.12; };

    zone "movie.edu" {
        type master;
        file "db.movie.edu.external";
        allow-transfer { "ns1.isp.net"; };
    };

    zone "4.1.200.in-addr.arpa" {
        type master;
        file "db.200.1.4";
        allow-transfer { "ns1.isp.net"; };
    };

    zone "." {
        type hint;
        file "db.cache";
    };
};
```

Обратите внимание, что внешний и внутренний виды представляют различные версии зоны *movie.edu*: один вид загружает данные зоны из файла *db.movie.edu*, второй – из файла *db.movie.edu.external*. Если бы внешний вид содержал больше зон, вероятно, пришлось бы использовать отдельные подкаталоги для хранения файлов данных зон, видимых извне, и файлов данных «внутренних» зон.

Расширения системы безопасности DNS

Транзакционные подписи (TSIG), описанные ранее в этой главе, хорошо подходят для обеспечения безопасности обмена между двумя серверами либо между DNS-сервером и клиентом, посылающим обновления. Однако транзакционные подписи не помогут, если защищенность одного DNS-серверов под вопросом: злоумышленник, проникший на узел, на котором работает один из DNS-серверов, может получить доступ к TSIG-ключам. Более того, поскольку в TSIG применяются разделяемые секреты, не очень практично использовать этот механизм для многих DNS-серверов. TSIG также невозможно применять для об-

мена между администрируемыми DNS-серверами и произвольными DNS-серверами сети Интернет, поскольку невозможно распределять и сопровождать такое количество ключей.

Наиболее распространенный способ решения проблем, связанных с сопровождением ключей, – применение *шифрования с открытым ключом*. Расширения системы безопасности DNS, описанные в документе RFC 2535, позволяют администраторам зон использовать шифрование с открытым ключом для создания цифровых подписей зональных данных, таким образом подтверждая подлинность данных.

Примечание: мы опишем расширения системы безопасности DNS (DNSSEC) в том виде, в котором они определяются документом RFC 2535. Однако рабочий комитет DNSEXT организации IETF все еще продолжает работу над DNSSEC, и отдельные аспекты системы могут измениться, прежде чем она станет стандартом.
Кроме того: несмотря на то, что BIND 8 реализует предварительную поддержку DNSSEC уже в версии BIND 8.2[1], эта реализация не была готова к реальному применению вплоть до BIND 9. Поэтому в примерах речь идет о пакете BIND 9. Если необходимо пользоваться DNSSEC, на сегодняшний день это самый подходящий вариант.

Шифрование с открытым ключом и цифровые подписи

Шифрование с открытым ключом решает проблем распределения ключей путем использования ассиметричного алгоритма шифрования. В таком алгоритме один ключ используется для расшифровки данных, зашифрованных другим ключом. Эти два ключа – *пара ключей* – создаются одновременно с помощью математической формулы. Это единственный простой способ найти два ключа, обладающих специальной ассиметрией (при которой один из ключей может расшифровать зашифрованное вторым): вычисление одного из ключей по второму – задача невероятно сложная. (В наиболее популярном ассиметричном алгоритме шифрования, RSA, такое вычисление связано с разложением на множители очень больших чисел.)

При применении шифрования с открытым ключом прежде всего создается пара ключей. Затем один ключ из пары делается свободно доступным (к примеру, публикуется в каталоге), а второй сохраняется в секрете. Тот, кто хочет установить с создателем ключей безопасный контакт, может перед отправкой зашифровать свое сообщение, поль-

[1] В частности, BIND 8 не умеет следовать по цепи доверия. Он может произвести проверку SIG-записей только в зонах, для которых существует оператор *trusted-keys*.

зуясь открытым ключом. (Более того, сообщение можно даже поместить в конференцию или на веб-сайт.) Если адресат хранит закрытый ключ в секрете, только он сможет расшифровать сообщение.

И наоборот, автор пары ключей может зашифровать свое сообщение закрытым ключом. Получатель может удостовериться, что сообщение действительно принадлежит автору ключей, попробовав расшифровать его с помощью открытого ключа. Если сообщение после расшифровки имеет какой-то смысл, а отправитель действительно сохранил закрытый ключ в секрете, то сообщение действительно написано им. Успешная расшифровка также подтверждает, что сообщение не было изменено в процессе доставки (скажем, при прохождении через почтовый сервер), поскольку в противном случае не была бы получена правильная расшифровка. Так сообщение проверяется получателем.

К сожалению, шифрование больших объемов данных с помощью ассиметричных алгоритмов происходит медленно – гораздо медленнее, чем при использовании симметричных алгоритмов. Но при использовании шифрования с открытым ключом, в целях проверки подлинности (а не для сохранения конфиденциальности), совершенно необязательно шифровать сообщение целиком. Вместо этого сообщение сначала обрабатывается вычислительно необратимой хеш-функцией. Затем достаточно зашифровать только хеш-значение, которое является представлением исходных данных. Зашифрованное хеш-значение, называемое теперь *цифровой подписью*, добавляется к сообщению, для которого будет производиться проверка подлинности. Получатель может проверить подлинность сообщения, расшифровав подпись и обработав сообщение собственной копией вычислительно необратимой хеш-функции. Если полученные хеш-значения совпадают, сообщение подлинное. Процесс подписи и проверки сообщения отражен на рис. 11.7.

Запись KEY

В DNSSEC, расширениях безопасности DNS, с каждой защищаемой зоной связывается пара ключей. Закрытый ключ зоны хранится в безопасном месте – часто в файле на узле DNS-сервера. Открытый ключ распространяется в составе RR-записи нового типа, которая связывается с доменным именем зоны. Речь идет о записи KEY.

Запись типа KEY в действительности является записью общего назначения, как станет ясно из наших объяснений. Записи этого типа можно использовать для хранения различных видов ключей шифрования, а не только открытых ключей для зон, защищаемых с помощью DNSSEC. Тем не менее, в этой книге мы будем изучать только то применение записи KEY, которое связано с хранением открытых ключей зон.

Запись KEY выглядит следующим образом:

```
movie.edu. IN KEY 256 3 1 AQPdWbrGbVv1eDhNgRhpJMPonJfA3reyEo82ekwRnjbX7+uBxB11Bq
L7 LAB7/C+ebOvCtI53FwMhkkNkTmA6bI8B
```

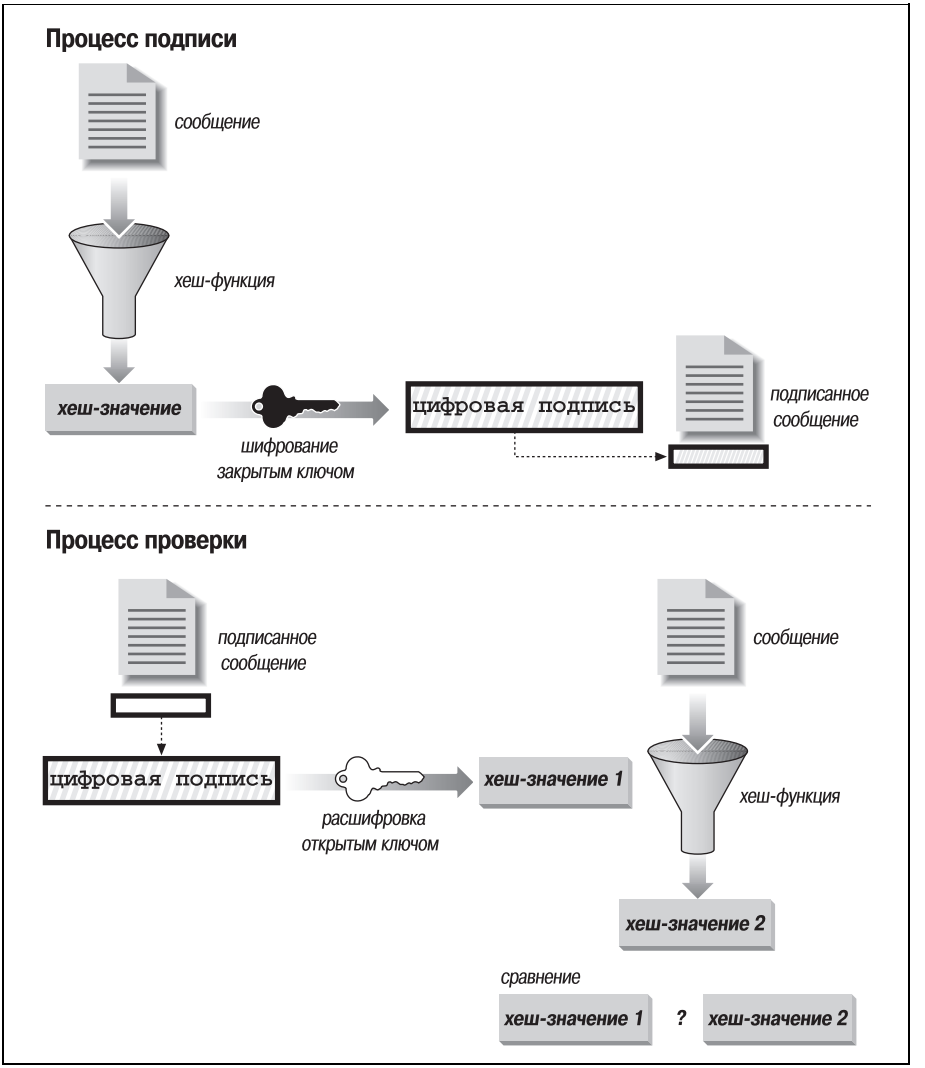

Рис. 11.7. *Подпись и проверка сообщения*

Запись принадлежит доменному имени зоны, связываемой с открытым ключом. За полем типа следует поле флагов, в котором записано значение 256. Поле флагов имеет длину два байта и содержит набор одно- и двухбитовых значений:

```
 0   1   2   3   4   5   6   7   8   9   0   1   2   3   4   5
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|  A/C  | Z | XT| Z | Z | NAMTYP| Z | Z | Z | Z |      SIG      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Если значение первого бита – нуль, ключ может использоваться для проверки подлинности. Само собой, ключ, который не может быть использован для проверки подлинности, не особо полезен в DNSSEC; так что этот бит всегда сброшен.

Если значение второго бита – нуль, ключ может использоваться для сохранения конфиденциальности. DNSSEC не делает данные зоны закрытыми, но и не запрещает использовать открытый ключ зоны для сохранения конфиденциальности, так что для открытого ключа зоны этот бит всегда сброшен.

Запись KEY, в которой оба этих бита установлены, называется *пустым ключом*. Позже мы расскажем, как пустые ключи используются в родительских зонах.

Значение третьего бита зарезервировано для будущих нужд. Четвертый бит – это бит «расширения флага». Его существование позволяет обеспечить удлинение поля флагов. Если этот бит установлен, запись KEY должна содержать еще одно двухбайтовое поле после поля алгоритма (третье поле после типа записи и непосредственно перед открытым ключом (который обычно записывается в четвертом поле). Значения битов дополнительного двухбайтого поля пока еще не определены, так что четвертый бит всегда сброшен. Пятый и шестой биты зарезервированы, как и третий, и должны быть сброшены.

Седьмой и восьмой биты кодируют тип ключа:

00 Ключ пользователя. Почтовый клиент пользователя может использовать ключ пользователя для шифрования почты, адресованной этому пользователю. Этот тип ключа не используется в DNSSEC.

01 Открытый ключ зоны. Все ключи в DNSSEC имеют этот тип.

10 Ключ узла. В реализации IPSEC ключ узла может использоваться для шифрования всез IP-пакетов, посылаемых этому узлу. В DNSSEC ключ узла не используется.

11 Значение зарезервировано для будущих нужд.

Биты с девятого по двенадцатый зарезервированы и должны быть сброшены. Последние четыре бита составляют поле подписи, которое в настоящее время вышло из употребления.

В приведенной записи KEY поле флагов (первое после типа записи) говорит о том, что эта KEY-запись является ключом зоны *movie.edu* и может использоваться для проверки подлинности и сохранения конфиденциальности.

Следующее поле записи, в котором в нашем примере записано значение 3, носит название *октета протокола*. Поскольку KEY-записи могут использоваться в различных целях, следует указывать предназначение каждого конкретного ключа. Определены следующие возможные значения:

0 Зарезервировано.

1 Этот ключ используется в системе TLS (Transport Layer Security, безопасность транспортного уровня), описанной в RFC 2246.

2 Этот ключ используется в контексте электронной почты, к примеру, может являться ключом S/MIME.

3 Ключ используется в контексте DNSSEC. Очевидно, для всех ключей DNSSEC октет протокола имеет значение 3.

4 Ключ используется в контексте IPSEC.

255

Ключ используется с любым протоколом, в котором реализована поддержка KEY-записей.

Все значения большие 4 и меньшие 255 доступны для использования в будущем.

Следующее (третье) поле записи KEY, которое в нашем примере содержит значение 1, определяет номер алгоритма. DNSSEC может использоваться совместно с несколькими алгоритмами шифрования на основе открытого ключа, поэтому следует указывать, какой алгоритм используется для зоны и с каким алгоритмом следует использовать данный ключ. Определены следующие значения:

0 Зарезервировано.

1 RSA/MD5. Документ RFC 2535 рекомендует, но не требует, использование RSA/MD5. Однако алгоритм RSA является очень популярным, а патент на алгоритм недавно истек, поэтому ходят слухи о том, что использование RSA может стать обязательным.

2 Алгоритм Диффи-Хеллмана (Diffie-Hellman). Документ RFC 2535 допускает использование этого алгоритма.

3 DSA. Документ RFC 2535 предписывает обязательную *поддержку* (не использование) DSA. Но, как мы уже сказали, такое положение дел может измениться.

4 Зарезервировано для алгоритма шифрования на основе эллиптических кривых.

В наших примерах будут использоваться ключи RSA, поскольку они, скорее всего, станут стандартом.

Последнее поле KEY-записи содержит собственно открытый ключ в кодировке Base 64. DNSSEC поддерживает ключи различных длин, как мы скоро увидим при создании открытого ключа для *movie.edu*. Чем длиннее открытый ключ, чем сложнее подобрать соответствующий закрытый ключ, тем больше времени отнимает процесс подписи данных зоны закрытым ключом и процесс проверки данных с помощью открытого ключа.

Пустые ключи не содержат открытых ключей, но для них присутствуют октеты протоколов и номера алгоритмов.

Запись SIG

Если открытый ключ зоны хранится в записи KEY, то должен существовать и тип записей для хранения подписи соответствующего закрытого ключа? Так и есть, тип записей называется SIG. В SIG-записи хранится цифровая подпись закрытого ключа для структуры *RRset*. RRset – это набор RR-записей с общим владельцем, классом и типом; так, все адресные записи *wormhole.movie.edu* составляют структуру RRset. То же верно и для всех MX-записей *movie.edu*.

Почему подписывается набор записей (RRset), а не отдельные записи? Чтобы сэкономить время. Невозможно произвести поиск единственной адресной записи *wormhole.movie.edu*; DNS-сервер возвращает всю группу. Так зачем подписывать каждую запись в отдельности, имея возможность подписать их все сразу?

Вот SIG-запись для адресных записей *wormhole.movie.edu*:

```
wormhole.movie.edu.          SIG     A 1 3 86400 20010102235426 (
                             20001203235426 27791 movie.edu.
                             1S/LuuxhSHs2LknPC7K/7v4+PNxESKZnjX6CtgGLZDWf
                            Rmovkw9VpW7htTNJYhzlFck/BO/k17tRj0fbQ6JWaA== )
```

Имя владельца в данном случае *wormhole.movie.edu*, и оно совпадает с именем, к которому привязаны подписываемые записи. Первое после типа содержит *тип покрытия* (в данном случае A). Оно определяет какие именно записи *wormhole.movie.edu* подписываются; в данном случае речь идет об адресных записях. Для каждого типа записей *wormhole.movie.edu* понадобится отдельная SIG-запись.

Второе поле, в котором записано значение 1, определяет номер алгоритма. Интерпретация поля такая же как для соответствующего поля записей KEY, так что значение 1 означает использование алгоритма RSA/MD5. При использовании ключа RSA для подписи зон, подписи, естественно, будут иметь тип RSA/MD5. Если для подписи зоны используются различные типы ключей, скажем, ключ RSA и ключ DSA, для каждого RRset-набора будет присутствовать отдельная SIG-запись, первая с номером алгоритма 1 (RSA/MD5), а вторая с номером алгоритма 3 (DSA).[1]

Третье поле носит название *поля метки*. Оно отражает число меток в имени владельца подписанных записей. В имени *wormhole.movie.edu*, очевидно, три метки, поэтому поле содержит значение 3. В каком случае поле метки может не соответствовать числу меток в имени владельца SIG-записи? Когда SIG-запись создается для записи с маской. К

[1] Зона может быть подписана ключами двух различных алгоритмов, чтобы люди, программное обеспечение которых реализует только поддержку DSA, также были в состоянии производить проверку данных, а те, кому больше нравится RSA, могли использовать RSA.

сожалению (или к счастью, чтобы мы не сошли с ума), BIND не поддерживает записи с масками в защищенных зонах.

Четвертое поле содержит исходное значение TTL для записей в RRset-наборе. (Предполагается, что записи набора имеют одинаковое время жизни.) Значение TTL должно храниться здесь, поскольку DNS-сервер, кэширующий записи из RRset-набора, к которому относится эта SIG-запись, будет постепенно уменьшать TTL кэшируемых записей. Не имея исходного значения TTL, невозможно обработать записи в их исходном виде вычислительно необратимой хеш-функцией и произвести проверку цифровой подписи.

Следующие два поля содержат временные отметки истечения срока действия и создания подписи. Обе даты хранятся в виде числа секунд, прошедших с момента начала эпохи Unix, то есть с первого января 1970 года, но в текстовом представлении записей SIG они записываются в формате YYYYMMDDHHMMSS – для удобства. (Время устаревания для приведенной ранее SIG-записи соответствует 11:54 2 января 2001 года.) Время создания подписи обычно совпадает со временем выполнения программы, производящей подпись зоны. Время устаревания подписи также выбирается во время запуска программы. После того как срок действия подписи закончился, SIG-запись более не может использоваться для проверки RRset-набора. Неприятная новость: это означает, что придется периодически повторно подписывать данные зоны, чтобы подписи могли использоваться. Приятная новость: повторная подпись отнимает намного меньше времени, чем в первый раз.

Следующее (седьмое) поле SIG-записи, которое в нашем примере содержит число 27791, – это поле *карты ключа*. Карта ключа – это значение, полученное на основе открытого ключа, соответствующего закрытому ключу, которым подписана зона. Если для зоны существует более одного открытого ключа (а так оно и будет при смене ключей), в процессе проверки программное обеспечение DNSSEC использует карту ключа, чтобы определить, какой ключ следует использовать для проверки подписи.

Восьмое поле, содержащее значение *movie.edu*, – это *поле автора*. Как можно ожидать, оно содержит доменное имя открытого ключа, который должен использовать проверяющий при оценке подлинности подписи. Это имя в паре с картой ключа определяет KEY-запись, которую следует использовать. В большинстве случаев имя автора содержит доменное имя зоны, которой принадлежат подписанные записи. Но в одном варианте – о котором мы скоро расскажем – может содержать доменное имя родительской зоны.

Последнее поле – *поле подписи*. Оно содержит цифровую подпись закрытого ключа зоны для записей набора и собственно SIG-записи, за исключением самого поля. Как и ключ в KEY-записи, подпись представляется в кодировке Base 64.

Запись NXT

В DNSSEC используется еще один тип записей – NXT. Сейчас мы объясним его предназначение.

Что происходит при поиске для доменного имени, которое не существует в защищенной зоне? Если бы зона не была защищена, DNS-сервер просто вернул бы код ошибки «no such domain name» (доменное имя не существует). Но как подписать код ошибки? Если подписывать все ответное сообщение, его будет проблематично кэшировать.

NXT-записи предназначены решить эту проблему. Они «заполняют» пробелы между двумя последовательно расположенными доменными именами зоны, позволяя определить, какое доменное имя следует за определенным доменными именем – отсюда и название записи («next» – следующий).

Но разве понятие «последовательно расположенных доменных имен» не подразумевает, что доменные имена в зоне должны располагаться в некоем каноническом порядке? Разумеется, так и есть.

Чтобы упорядочить доменные имена зоны, следует сначала отсортировать их по самой правой метке, затем перейти к следующей слева, и т. д. Сортировка по меткам производится без учета регистра символов, в алфавитном (словарном) порядке, причем цифры предшествуют буквам, а несуществующие метки – цифрам (другими словами, *movie.edu* предшествует *0.movie.edu*). Таким образом, доменные имена зоны *movie.edu* сортируются следующим образом:

```
movie.edu
bigt.movie.edu
carrie.movie.edu
cujo.movie.edu
dh.movie.edu
diehard.movie.edu
fx.movie.edu
bladerunner.fx.movie.edu
outland.fx.movie.edu
horror.movie.edu
localhost.movie.edu
misery.movie.edu
robocop.movie.edu
shining.movie.edu
terminator.movie.edu
wh.movie.edu
wh249.movie.edu
wh253.movie.edu
wormhole.movie.edu
```

Обратите внимание, как *movie.edu* предшествует *bigt.movie.edu*, так и *fx.movie.edu* предшествует имени *bladerunner.fx.movie.edu*.

Когда зона приведена в канонический порядок, записи NXT приобретают смысл. Вот одна NXT-запись (по сути дела – первая) зоны *movie.edu*:

```
movie.edu.                      NXT     bigt.movie.edu. ( NS SOA MX SIG NXT )
```

Эта запись говорит о том, что следующее после *movie.edu* доменное имя в зоне – *bigt.movie.edu*, как мы можем видеть по сортированному списку доменных имен. Помимо этого, здесь содержится информация о том, что для имени *movie.edu* существуют NS-записи, SOA-запись, MX-записи, SIG-запись и NXT-запись.

Последняя NXT-запись зоны является особенной. Поскольку следующее доменное имя не существует, последняя NXT-запись ссылается на первую запись зоны:

```
wormhole.movie.edu.                NXT     movie.edu. ( A SIG NXT )
```

Другими словами, чтобы показать, что *wormhole.movie.edu* является последним доменным именем в зоне, мы говорим, что следующим доменным именем является *movie.edu*, первое доменное имя зоны.

Каким же образом NXT-записи позволяют производить проверку подлинности для отрицательных ответов? Если бы мы произвели локальный поиск для *www.movie.edu*, то получили бы в ответ NXT-запись *wormhole.movie.edu*, сообщающую, что *www.movie.edu* не существует, поскольку в зоне не существует доменных имен после *wormhole.movie.edu*. Точно так же, если бы мы произвели поиск TXT-записей для *movie.edu*, то получили бы NXT-запись, которая приводится выше, сообщающую, что для *movie.edu* не существует TXT-записей, хотя существуют записи NS, SOA, MX, SIG и NXT.

SIG-запись для этой NXT-записи сопровождает ее в ответном сообщении, таким образом удостоверяя подлинность факта отсутствия искомого доменного имени или типа данных.

Очень важно, что NXT-записи абсолютно конкретно сообщают, что имени не существует в зоне. Простая, универсальная запись, сообщающая нам, что «это не существует», может быть перехвачена и повторно послана впоследствии злоумышленником, вводя нас в заблуждение относительно существования доменных имен или записей.

Читатели могут не беспокоиться о перспективе ручного добавления и сопровождения этих записей (О-хо-хо, я добавил узел, теперь придется исправлять NXT-записи...) – в BIND существует инструмент, позволяющий автоматически добавлять NXT- и SIG-записи.

Некоторых также заинтересует, какую информацию о зоне предоставляют NXT-записи злоумышленнику. Взломщик может, к примеру, произвести поиск NXT-записи, связанной с доменным именем зоны и получить следующее имя, затем повторить процесс для всех имен зоны. Таков, к сожалению, неизбежный эффект обеспечения безопасности зоны. Просто повторяйте мантру: «Моя зона защищена, но открыта для всех».

Цепь доверия

Еще один аспект теории DNSSEC, который нам следует обсудить, – цепь доверия. (Не подумайте, что речь идет о деликатных психологических упражнениях по созданию сплоченного коллектива!) До сих пор у каждого RRset-набора нашей защищенной зоны была соответствующая SIG-запись. Чтобы дать возможность другим проверять SIG-записи, наша зона раздает всему миру открытый ключ, инкапсулированный в KEY-записи. Но представим себе, что злоумышленник проник на первичный DNS-мастер-сервер. Что помешает ему создать собственную пару ключей? Он сможет изменить данные зоны, повторно подписать зону новым закрытым ключом и раздавать всему миру новый открытый ключ, инкапсулированный в KEY-записи.

Чтобы подобных проблем не возникало, открытые ключи подвергаются «сертификации» в более высоких инстанциях. Более высокая инстанция подтверждает, что ключ *movie.edu* в KEY-записи принадлежит организации, которая владеет и управляет этой зоной, а не какой-то первой встречной свинье. Прежде чем дать такое подтверждение, инстанция требует доказательства, что мы те, за кого себя выдаем, и что мы имеем надлежащие полномочия, чтобы администрировать *movie.edu*.

В нашем случае более высокой инстанцией является родительская зона *edu*. Создав пару ключей и подписав зону, мы посылаем открытый ключ администраторам *edu*, наряду с удостоверениями личности и доказательствами того, что мы являемся Двумя Подлинными Администраторами *movie.edu*.[1] Администраторы родительской зоны подписали нашу KEY-запись закрытым ключом зоны *edu* и вернули ее нам, чтобы мы могли добавить ее к своей зоне. Вот наша KEY-запись и сопровождающая ее SIG-запись:

```
movie.edu               IN SIG  KEY 1 2 3600 20010104010141 (
                                20001205010141 65398 edu.
                                aE4sCZKgFtp5RuD1sib0+19dc3MF/y9S2Fr8+h66g+Y2
                              1bc31M4y0493cSoyRpapJrd7qfG+Cr7GK+uY+eLCRA== )
                        KEY     256 3 1 (
                                AQPdWbrGbVv1eDhNgRhpJMPonJfA3reyEo82ekwRnjbX
                              7+uBxB11BqL7LAB7/C+eb0vCtI53FwMhkkNkTmA6bI8B )
```

Обратите внимание, что в поле имени автора SIG-записи содержится строка *edu*, а не *movie.edu*; это свидетельствует о том, что наша KEY-запись была подписана закрытым ключом родительской зоны, а не локальной.

Что если кто-то сможет проникнуть на первичный DNS-мастер-сервер зоны *edu*? KEY-записи зоны *edu* подписываются закрытым ключом

[1] В настоящее время ни одна зона высшего уровня не производит подпись KEY-записей для порожденных зон, хотя некоторые европейские регистраторы, вполне вероятно, вскоре этим займутся.

корневой зоны. А корневая зона? Дело в том, что открытый ключ корневой зоны широко известен и доступен на каждом DNS-сервере, поддерживающем DNSSEC.[1]

То есть открытый ключ корневой зоны будет доступен на каждом DNS-сервере, как только DNSSEC получит широкое распространение. В настоящее же время ни корневая зона, ни зона *edu* не подписываются, и ни у одной из них нет собственной пары ключей. В ожидании широкого распространения механизма DNSSEC его можно использовать частично.

Защищенные сегменты

Предположим, мы начали использовать DNSSEC в Университете кинематографии, чтобы повысить защищенность данных зоны. Мы подписали зону *movie.edu*, но не можем получить подпись зоны *edu* для нашей KEY-записи, поскольку эта зона еще не является защищенной и для нее не существует пара ключей. Как могут другие DNS-серверы в сети Интернет проверять подлинность наших данных? Да и вообще, как могут наши собственные DNS-серверы проверять подлинность наших же данных?

В DNS-серверах BIND 9 существует механизм, позволяющий указать в файле *named.conf* открытый ключ, соответствующей определенной зоне. Речь идет об операторе *trusted-keys*. Вот оператор *trusted-keys* для *movie.edu*:

```
trusted-keys {
     movie.edu. 256 3 1 "AQPdWbrGbVv1eDhNgRhpJMPonJfA3reyEo82ekwRnjbX7+uBxB11Bq
L7 LAB7/C+eb0vCtI53FwMhkkNkTmA6bI8B";
};
```

По сути дела – это запись KEY без полей класса и типа. Еще одно отличие – ключ заключен в кавычки. Доменное имя зоны может быть заключено в кавычки, но не обязательно. Если бы зона *movie.edu* имела несколько открытых ключей – скажем, еще и ключ DSA, мы могли бы включить все ключи в оператор:

```
trusted-keys {
    movie.edu. 256 3 1 "AQPdWbrGbVv1eDhNgRhpJMPonJfA3reyEo82ekwRnjbX7+uBxB11Bq
L7 LAB7/C+eb0vCtI53FwMhkkNkTmA6bI8B";
    movie.edu. 256 3 3 "AMnD8GXACuJ5GVnfCJWmRydg2A6JptSm6tjH7QoL81SfBY/kcz1Nbe
Hh z4l9AT1GG2kAZjGLjHO7BZHY+joz6iYMPRCDaPOIt9LO+SRfBNZg62P4 aSPT5zVQPahDIMZmTIvv
O7FV6IaTV+cQiKQl6noro8uTk4asCADrAHwO iVjzjaYpoFF5AsBOcJU18fzDiCNBUb0VqE1mKFuRA/K
```

[1] Нам вспоминается байка про человека, который как-то спросил жреца, что держит Землю. Жрец ответил, что Земля покоится на панцире огромной черепахи. Тогда человек спросил, что держит черепаху. И жрец ответил: «Черепаха покоится на спине огромного слона». Но что же, спросил человек, держит слона? Разозленный жрец выпалил «А дальше слоны – до самого конца!»

```
1KyxM2vJ3U7IS toOIgACiCfHkYK5r3qFbMvF1GrjyVwfwCC4NcMsqEXIT8IEI/YYIgFt4 Ennh";
};
```

Приведенный оператор *trusted-keys* позволяет DNS-серверу BIND 9 производить проверку подлинности любых записей зоны *movie.edu*. DNS-сервер сможет также осуществлять проверку записей из порожденных зон вроде *fx.movie.edu*, в случае если их KEY-записи подписаны закрытым ключом зоны *movie.edu*, и для *внуков movie.edu*, в случае наличия корректной цепи доверия, которую можно проследить зо открытого ключа зоны *movie.edu*. Другими словами, *movie.edu* становится *защищенным сегментом*, в пределах которого DNS-серверы этой зоны могут производить проверку подлинности данных из защищенных зон.

Пустые ключи

Защищенный сегмент позволяет DNS-серверу зоны производить проверку подлинности подписанных записей, начиная с определенной точки пространства имен *Пустые ключи* решают совершенно противоположную задачу: они сообщают DNS-серверу, что записи, начиная с определенной точки, не защищены. Допустим, администраторы *fx.movie.edu* еще не занимались вопросами безопасности своей зоны. Когда мы подписываем зону *movie.edu*, программный модуль BIND 9, ответственный за подписи, добавляет специальный пустой ключ к зоне *movie.edu* – для *fx.movie.edu*:

```
fx.movie.edu.           KEY     49408 3 3 (
                                 )
```

Обратите внимание, что открытый ключ в этой записи не закодирован в Base 64. Если присмотреться *внимательно* (или воспользоваться инженерным калькулятором), можно понять, что поле флагов говорит о том, что ключ нелья применять ни для проверки подлинности, ни для сохранения конфиденциальности; то есть ключ – пустой. DNS-серверы, реализующие поддержку DNSSEC, интерпретируют этот факт следующим образом: зона *fx.movie.edu* не является защищенной и не следует ожидать, что она содержит подписанные данные.

Если программный модуль BIND 9, ответственный за подписи, обнаружит при запуске файл, содержащий KEY-запись *fx.movie.edu*, пустой ключ будет удален, поскольку предполагается, что зона *fx.movie.edu* защищена.

Как используются записи

Рассмотрим, какие действия совершает DNS-сервер, реализующий механизм DNSSEC, для проверки записи в *movie.edu*. В частности, интересно посмотреть, что происходит при поиске адреса для *wormhole.movie.edu*. Во-первых, DNS-сервер посылает запрос этого адреса:

```
% dig +dnssec +norec wormhole.movie.edu.
```

```
; <<>> DiG 9.1.0 <<>> +dnssec +norec wormhole.movie.edu.
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15766
;; flags: qr aa ra; QUERY: 1, ANSWER: 3, AUTHORITY: 4, ADDITIONAL: 6

;; OPT PSEUDOSECTION:
; EDNS: version:    0, udp=   4096
;; QUESTION SECTION:
;wormhole.movie.edu.            IN      A

;; ANSWER SECTION:
wormhole.movie.edu.     86400   IN      A       192.249.249.1
wormhole.movie.edu.     86400   IN      A       192.253.253.1
wormhole.movie.edu.     86400   IN      SIG     A 1 3 86400 20010215174848
20010116174848 27791 movie.edu.
cYKQvgVksHjwGedNz72iyIpjXBhtSOeUEQA6VOb6l8asG3mpV6hzrzNf
YwTpLoh9FSjSfOkUzmXkW9aYJmd5Bw==

;; AUTHORITY SECTION:
movie.edu.              86400   IN      NS      outland.fx.movie.edu.
movie.edu.              86400   IN      NS      wormhole.movie.edu.
movie.edu.              86400   IN      NS      terminator.movie.edu.
movie.edu.              86400   IN      SIG     NS 1 2 86400 20010215174848
20010116174848 27791 movie.edu.
ZXRnlbJBWJa4XX3YTWgkYnoQjGLFDN+2JwoGpLpxTidwkJOFT+N3gMSw anSxa22b+X/
7v4b99t2WMcxCtUIXvw==

;; ADDITIONAL SECTION:
outland.fx.movie.edu.   86400   IN      A       192.253.254.3
terminator.movie.edu.   86400   IN      A       192.249.249.3
terminator.movie.edu.   86400   IN      SIG     A 1 3 86400 20010215174848
20010116174848 27791 movie.edu.
GSnxseyN4w5sA2Fb9uK9zVNSRJRbbcvrODaDRwLDO8X2m6ZBbkRssSHJ tZYwoO4ZIFERLKakB//
VTDMhYJmNvw==
movie.edu.              86400   IN      KEY     256 3 1
AQPdWbrGbVv1eDhNgRhpJMPonJfA3reyEo82ekwRnjbX7+uBxB11BqL7 LAB7/
C+ebOvCtI53FwMhkkNkTmA6bI8B
movie.edu.              86400   IN      SIG     KEY 1 2 86400 20010215174837
20010116174837 65398 edu. LW+nc2gmz618u/LjDtlKSorv9OkJOwC8wj/sa/CpzCJJqceB/
55JhsWI t1ADlfQwb4h9hs6oMeN2sU9jHiYQmw==

;; Query time: 3 msec
;; SERVER: 206.168.194.122#53(206.168.194.122)
;; WHEN: Tue Jan 16 10:49:48 2001
;; MSG SIZE  rcvd: 671
```

Обратите внимание, что следует указать ключ командной строки *+dnssec*. DNS-серверы BIND версии 9.1.0 и более поздних включают записи DNSSEC (SIG, NXT и KEY) в ответ только в том случае, когда клиент явным образом сообщает, что способен работать с DNSSEC. Как это делают авторы запросов? Установкой специального флага в «псевдораз-

деле» заголовка. Псевдораздел в действительности не является частью заголовка. В действительности это запись нового типа, OPT, которая включается в сообщение DNS. OPT-записи обычно сообщают о способностях автора запроса.

Заметим также, что ответное сообщение включает четыре SIG-записи: одну для записей из раздела ответа, одну для записей из раздела авторитативности, одну для адресной записи *terminator.movie.edu* из дополнительного раздела, и еще одну для KEY-записи *movie.edu* из дополнительного раздела. Дополнительный раздел мог бы содержать SIG-запись для *outland.fx.movie.edu*, адресные записи для *wormhole.movie.edu*, и SIG-запись для этих адресных записей – в случае, если бы хватило места, и сообщение уместилось в одну UDP-дейтаграмму.

Чтобы произвести проверку SIG-записей, DNS-сервер должен изучить KEY-запись *movie.edu*, которая включена в дополнительный раздел. Но прежде чем использовать ключ, DNS-сервер должен проверить SIG-запись для этого ключа. Таким образом требуется по меньшей мере один дополнительный запрос: к одному из DNS-серверов зоны *edu*, на предмет получения открытого ключа – разумеется, только в том случае, когда открытый ключ для *movie.edu* не указан в операторе *trusted-keys*.

DNSSEC и производительность

Из приведенного вывода *dig* должно быть ясно, что DNSSEC увеличивает средний размер сообщений DNS, требует значительно большей вычислительной мощности от DNS-серверов, производящих проверку зональных данных, и что подпись зоны значительно увеличивает ее размер (существующая оценка примерно такова – подпись увеличивает размер зоны в семь раз). Каждый из этих фактов имеет последствия, причем далеко не всегда очевидные:

- Более крупные сообщения DNS означают учащение случаев усечения сообщений, то есть больше переходов к использованию TCP. Использование TCP, разумеется, гораздо более требовательно к ресурсам, чем использование UDP.
- Проверка данных зоны требует времени и замедляет процесс разрешения.
- Более крупные зоны означают более тяжелые и хуже поддающиеся управлению процессы *named*.

По сути дела, сложность DNSSEC привела к тому, что архитектура BIND 8 не смогла справиться с этим механизмом. Так что появление механизма DNSSEC послужило толчком для разработки BIND 9, умеющего использовать преимущества многопроцессорных систем. Если планируется применять подписи для зон, следует убедиться, что авторитативные DNS-серверы обеспечены достаточными объемами памяти для загрузки крупных зон. Если DNS-серверы в основном занимаются разрешением в защищенных зонах, убедитесь также, что им доступны

соответствующие процессорные мощности, которые позволят осуществлять проверку цифровых подписей. И помните, BIND 9 способен использовать любой дополнительный процессор узла, на котором работает.

Подпись для зоны

Итак, читатели получили теоретическую подготовку, необходимую для создания подписи зоны. Мы рассмотрим процесс на примере зоны *movie.edu*. Помните, что мы использовали инструменты BIND 9, которые намного легче в применении, чем инструменты BIND 8, и, конечно, поддержка DNSSEC в BIND 9 намного более совершенна, чем в BIND 8.

Создание пары ключей

Итак, во-первых мы создали пару ключей для зоны *movie.edu*:

```
# cd /var/named
# dnssec-keygen -a RSA -b 512 -n ZONE movie.edu.
Kmovie.edu.+001+27791
```

Мы выполнили программу *dnssec-keygen* в рабочем каталоге DNS-сервера. Это было сделано для нашего же удобства: файлы данных зон расположены в этом же каталоге, и нет необходимости в аргументах использовать полные имена. При этом, если мы собираемся использовать динамические обновления с DNSSEC, то ключи должны находиться в рабочем каталоге DNS-сервера.

Вспомним опции программы *dnssec-keygen* из раздела TSIG этой главы (как давно это было):

-*a* Используемый алгоритм шифрования, в данном случае RSA. Мы могли бы также использовать алгоритм DSA, но RSA более эффективен.

-*b* Длина создаваемых ключей, в битах. Ключи RSA могут иметь длину от 512 до 2000 битов. Ключи DSA – от 512 до 1024 битов, причем длина должна нацело делиться на 64.

-*n* Тип ключа. Ключи DNSSEC всегда являются ключами зоны.

Единственный самостоятельный аргумент в данном случае – доменное имя зоны, *movie.edu*. Программа *dnssec-keygen* отображает основу имен файлов, в которых записаны ключи. Как мы уже рассказывали в разделе TSIG, числа в в этих строках (001 и 27791) соответствуют номеру алгоритма DNSSEC, использованного для создания KEY-записи (001 соответствует RSA/MD5), а также карте ключа, которая используется для различения ключей, связанных с одной и той же зоной.

Открытый ключ записывается в файл *основа.key* (*Kmovie.edu.+001+27791.key*). Закрытый ключ записывается в файл *основа.private* (*Kmovie.edu.+001+27791.private*). Помните, что закрытый ключ следу-

ет хранить в тайне, поскольку любой, кому известен этот ключ, способен официально подделать данные зоны. *dnssec-keygen* пытается нам помочь, делая файлы *.private* доступными для чтения и записи только пользователю, который выполняет программу.

Отправка ключей на подпись

Теперь следует отправить нашу KEY-запись администратору родительской зоны – на подпись. В BIND 9 существует приятная маленькая программа, которая упаковывает ключи для передачи, *dnssec-makekeyset*:

```
# dnssec-makekeyset -t 172800 Kmovie.edu.+001+27791.key
```

В результате работы *dnssec-makekeyset* был создан файл *keyset-movie.edu*[1], который содержит следующие строки:

```
$ORIGIN .
$TTL 172800      ; 2 days
movie.edu               IN SIG  KEY 1 2 86400 20010104034839 (
                                20001205034839 27791 movie.edu.
                                M7RDKMyc9w1djDcOmQAXQc1PJdmLRBg3nfaGEUZe9Fbi
                               mjiNVaQK33IWhzI95oD8ASOWqRDy5TusTXt4nx1/dQ== )
                        KEY     256 3 1 (
                                AQPdWbrGbVv1eDhNgRhpJMPonJfA3reyEo82ekwRnjbX
                               7+uBxB11BqL7LAB7/C+ebOvCtI53FwMhkkNkTmA6bI8B )
```

Ключ *-t* позволяет задать параметр TTL для передаваемых записей. Это предпочтительное значение TTL для наших записей (в секундах), о котором мы и сообщаем администратору родительской зоны. Администратор родительской зоны может, само собой, проигнорировать это значение. SIG-запись содержит подпись, удостоверяющую KEY-запись нашей зоны, которая была создана с помощью закрытого ключа зоны. Так мы доказываем, что действительно обладаем закрытым ключом, который соответствует открытому ключу из KEY-записи, а не посылаем какую-то KEY-запись, которую только что нашли на улице.

Поля временных отметок создания и устаревания подписи по умолчанию принимают значения «только что» и «через 30 дней». Это предпочтительные значения времени жизни, учитываемые создателем подписи. Можно использовать ключи *-s* (start, начало) и *-e* (end, конец) для задания собственных временных отметок создания и устаревания подписи. Оба ключа принимают аргументы в формате YYYYMMDDHHMMSS, либо смещения. Смещение для *-s* вычисляется отно-

[1] В версиях *dnssec-makekeyset*, поставляемых в комплекте BIND 9.0.1 и более ранних версий, имя файла выглядит так: *movie.edu.keyset*. Однако такое построение имени вызывало проблемы – имя файла набора ключей корневой зоны было *.keyset*, такой файл в файловой системе Unix является скрытым.

сительно текущего времени. Смещение для *-e* вычисляется относительно времени создания подписи (start).

Эти два поля можно также использовать для создания нескольких наборов ключей (keysets) и одновременной передачи их на подпись администратору родительской зоны. К примеру, можно единовременно передать администратору родительской зоны наборы ключей, действующие в январе, феврале и марте, а затем вводить их в действие по одному каждый месяц.

Файл был отправлен администратору родительской зоны на подпись. Поскольку сообщение включало подтверждение наших прав[1], администратор подписал файл с помощью программы *dnssec-signkey*:

```
# dnssec-signkey keyset-movie.edu Kedu.+001+65398.private
```

и вернул нам результат, файл *movie.edu.signedkey*:

```
$ORIGIN .
$TTL 172800       ; 1 час
movie.edu               IN SIG  KEY 1 2 3600 20010104010141 (
                                 20001205010141 65398 edu.
                                  aE4sCZKgFtp5RuD1sib0+19dc3MF/y9S2Fr8+h66g+Y2
                                1bc31M4y0493cSoyRpapJrd7qfG+Cr7GK+uY+eLCRA== )
                        KEY      256 3 1 (
                                  AQPdWbrGbVv1eDhNgRhpJMPonJfA3reyEo82ekwRnjbX
                                7+uBxB11BqL7LAB7/C+eb0vCtI53FwMhkkNkTmA6bI8B )
```

Если бы наличие подписей для KEY-записей не сильно нас беспокоило, мы могли бы пропустить этот шаг. В таком случае только DNS-серверы с записью *trusted-keys*, включающей *movie.edu*, смогли бы производить проверку наших данных.

Подписываем зону

Прежде чем подписать зону, мы должны добавить KEY-запись к файлу данных зоны:

```
# cat "$INCLUDE Kmovie.edu.+001+27791.key" >> db.movie.edu
```

Так программа, создающая подпись, будет в курсе, какой ключ следует использовать. Она автоматически обнаружит файл *movie.edu.signedkey* и воспользуется его содержимым.

Подпишем зону с помощью *dnssec-signzone*:

```
# dnssec-signzone -o movie.edu. db.movie.edu
```

[1] Поскольку зоны высшего уровня еще не начали подписывать ключи, вопрос идентификации администраторов порожденных зон пока остается открытым. Вполне возможно использование почтовых сообщений с криптографической подписью.

Мы использовали ключ *-o* для задания суффикса по умолчанию для файла данных зоны, поскольку *dnssec-signzone* не изучает файл *named.conf* и не в состоянии определить, какую зону описывает файл. Единственный самостоятельный аргумент в данном случае – имя файла данных зоны.

В результате мы получаем новый файл данных зоны, *db.movie.edu.signed*, который начинается следующим образом:

```
$ORIGIN .
$TTL 86400      ; 1 день
movie.edu               IN SOA  terminator.movie.edu. al.robocop.movie.edu. (
                           2000092603 ; порядковый номер
                           10800      ; обновление (3 часа)
                           3600       ; повторение(1 час)
                           604800     ; устаревание(1 неделя)
                           3600       ; минимум (1 час)
                                )
                        SIG     SOA 1 2 86400 20010104041530 (
                                20001205041530 27791 movie.edu.
                                aO1eZKhGSm99GgC9PfLXfHjl3tAWN/Vn33msppmyhN7a
                               RlfvJMTpSoJ9XwQCdjqhzOlcnCnQiL+jZkqU3uUecg== )
                        NS      terminator.movie.edu.
                        NS      wormhole.movie.edu.
                        NS      outland.fx.movie.edu.
                        SIG     NS 1 2 86400 20010104041530 (
                                20001205041530 27791 movie.edu.
                                pkmZJHqFlNmZdNjyupBMMzDDGweGsf9TS1EGci9cwKe5
                               cOo9h/yncInn2e8QSakjxpwKB8aw9D9uiStxJ/sLvQ== )
                        SIG     MX 1 2 86400 20010104041530 (
                                20001205041530 27791 movie.edu.
                                ZcKKeTOXaNlw83eSzRxt74DaLXvQtPYCdgKGOfSiJmYQ
                               WxI5zZUEWA6ku3w48mo9jbVF+/7nF3QcpFTIiwV1ug== )
$TTL 3600       ; 1 час
                        SIG     NXT 1 2 3600 20010104041530 (
                                20001205041530 27791 movie.edu.
                                upMjK21eD7OQkrHpxSWqkOPcRXbfL8WAgQVK1aGHcTPE
                               X3JtaLtCLuKLd3YFs7T8BuZoN7aJYRVREWSPVedYPw== )
                        NXT     bigt.movie.edu. ( NS SOA MX SIG KEY NXT )
$TTL 172800     ; 2 дня
                        SIG     KEY 1 2 172800 20001205040220 (
                                20001205040219 65398 edu.
                                HIReZ98rieIuRIO4XsoL+xLRLe8tCQbNKD8USlV35vb4
                               VsLUGCAEgBq7lLsHty7YCskbxhQu8ncysBKnR/muiA== )
                        KEY     256 3 1 (
                                AQPdWbrGbVv1eDhNgRhpJMPonJfA3reyEo82ekwRnjbX
                               7+uBxB11BqL7LAB7/C+ebOvCtI53FwMhkkNkTmA6bI8B )
$TTL 86400      ; 1 день
                        MX      10 postmanrings2x.movie.edu.
```

Верьте или нет, все это просто записи, связанные с доменным именем *movie.edu*. Файл данных зоны в целом в четыре раза длиннее и в пять раз больше. Ух!

И наконец, мы отредактировали оператор *zone* в файле *named.conf*, который теперь загружает новый файл данных зоны:

```
zone "movie.edu" {
    type master;
    file "db.movie.edu.signed";
};
```

Затем мы перезагрузили зону и проверили вывод *syslog*.

Вот ключи *dnssec-signzone*, которые мы еще не применяли:

-s, -e

Эти ключи позволяют задать временные отметки создания и устаревания подписей, которые следует использовать в SIG-записях; синтаксис аргументов в точности соответствует синтаксису для *dnssec-makekeyset*.

-i Позволяет задать длину цикла повторной подписи (эту тему мы рассмотрим через минуту). До BIND 9.1.0 вместо этого ключа использовался ключ *-c*.

-f Позволяет указать имя файла, в который происходит запись подписанной зоны. По умолчанию к имени исходного файла данных зоны просто добавляется суффикс *.signed*.

В качестве второго самостоятельного аргумента можно также указать закрытый ключ, который следует использовать для подписи зоны. По умолчанию *dnssec-signzone* подписывает зону каждым из закрытых ключей зоны, присутствующих в каталоге. Если указать имя одного или нескольких файлов, содержащих закрытые ключи зоны, в качестве аргумента, зона будет подписана с использованием только этих ключей.

Помните, что каждый раз, при изменении данных зоны – зону следует подписывать заново, хотя, конечно, нет необходимости каждый раз генерировать новую пару KEY-записей и подписывать новые ключи. Можно заново подписать зону, выполнив *dnssec-signzone* для подписанных зональных данных:

```
# dnssec-signzone -o movie.edu -f db.movie.edu.signed.new db.movie.edu.signed
# mv db.movie.edu.signed db.movie.edu.signed.bak
# mv db.movie.edu.signed.new db.movie.edu.signed
# rndc reload movie.edu
```

Программа достаточно сообразительна, она производит повторное вычисление NXT-записей, подписывает новые записи, а также заново подписывает записи, у которых скоро истекает срок действия подписи. По умолчанию *dnssec-signzone* повторно подписывает записи, срок

действия подписей для которых истекает в пределах 7,5 дней (четверть стандартного времени действия подписи). Если задать нестандартные значения для времени создания подписи и ее устаревания, *dnssec-signzone* соответствующим образом изменит связанные с временным циклом значения. Как вариант можно указать длину цикла с помощью ключа *-i* (ранее – *-c*).

DNSSEC и динамические обновления

Использование *dnssec-signzone* – не единственный способ подписывать зональные данные. DNS-сервер BIND 9 умеет подписывать динамически обновляемые записи на лету.[1] Королева в восхищении!

Если закрытый ключ для защищенной зоны доступен в рабочем каталоге DNS-сервера (и содержится в *.private*-файле с правильным именем), DNS-сервер BIND 9 подписывает любые записи, добавляемые посредством динамического обновления. При удалении и добавлении любых записей DNS-сервер исправляет (и подписывает заново) все соседние NXT-записи.

Продемонстрируем на примере. Для начала произведем поиск для доменного имени, которое еще не существует в зоне *movie.edu*:

```
% dig +dnssec perfectstorm.movie.edu.

; <<>> DiG 9.1.0 <<>> +dnssec perfectstorm.movie.edu.
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 4705
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 4, ADDITIONAL: 3

;; OPT PSEUDOSECTION:
; EDNS: version:    0, udp=   4096
;; QUESTION SECTION:
;perfectstorm.movie.edu.                IN      A

;; AUTHORITY SECTION:
movie.edu.              3600    IN      SOA     terminator.movie.edu.
al.robocop.movie.edu. 2001011600 10800 3600 604800 3600
movie.edu.              3600    IN      SIG     SOA 1 2 86400 20010215174848
20010116174848 27791 movie.edu.
Ea0+xyEsj0Hy4JP115r0D0UFVpWfxqf0NQA8hpKwlLCsxJ3rA+sJBg2Q ZiCTEwfAcwGRfbNsRYu/
CcuV/VJTDA==
misery.movie.edu.       86400   IN      NXT     robocop.movie.edu. A SIG NXT
misery.movie.edu.       86400   IN      SIG     NXT 1 3 86400 20010215174848
20010116174848 27791 movie.edu. ZVfV9KbPb8hKZdZirlpv+WnUxv72di8lUgZiot/
JaWDsZPfNoYqSnKPW ND4H92guwj7oR6CgrhsgLJ9dMDYSpg==
```

(Мы немного сократили вывод.) Обратите внимание на NXT-запись для *misery.movie.edu*, которая сообщает, что доменное имя не сущест-

[1] Еще одна возможность DNSSEC, которой нет в BIND 8.

вует. Используем *nsupdate* и добавим адресную запись для *perfectstorm.movie.edu*:

```
% nsupdate
> update add perfectstorm.movie.edu. 3600 IN A 192.249.249.91
>
```

Теперь снова произведем поиск для *perfectstorm.movie.edu*:

```
% dig +dnssec perfectstorm.movie.edu.

; <<>> DiG 9.1.0 <<>> +dnssec perfectstorm.movie.edu.
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 11973
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 4, ADDITIONAL: 9

;; OPT PSEUDOSECTION:
; EDNS: version:    0, udp=   4096
;; QUESTION SECTION:
;perfectstorm.movie.edu.                IN      A

;; ANSWER SECTION:
perfectstorm.movie.edu. 3600    IN      A       192.249.249.91
perfectstorm.movie.edu. 3600    IN      SIG     A 1 3 3600 20010215195456
20010116185456 27791 movie.edu. C/
JXdCLUdugxN91vODZuUDTusi2XNNttb4bdB2nBujLxjwwPAf/D5MJz //
cDtuZ3X+uYzhkN8MDROqOwUQuQSA==

;; AUTHORITY SECTION:
movie.edu.              86400   IN      NS      terminator.movie.edu.
movie.edu.              86400   IN      NS      outland.fx.movie.edu.
movie.edu.              86400   IN      NS      wormhole.movie.edu.
movie.edu.              86400   IN      SIG     NS 1 2 86400 20010215195301
20010116195301 27791 movie.edu.
1ZR592izM1AjMusJ26e4lvQOV9lFiFvQh6hCluBxSv7FwNqF7TcJFImc
W52XhXbHUEtiFOzDqYMHOzPV7j23nA==
```

(И снова мы подсократили результат.) На этот раз мы не только получили адресную запись, но и видим SIG-запись, созданную на основе закрытого ключа *movie.edu*. Подпись истекает через 30 дней после обновления – по умолчанию, но это значение можно изменить с помощью предписания *sig-validity-interval*, которое принимает число дней в качестве аргумента:[1]

```
options {
    sig-validity-interval 7;  // SIG-записи для обновленных записей
                              // существуют ровно неделю
};
```

[1] До BIND 9.1.0 аргумент *sig-validity-interval* интерпретировался как число секунд, а не дней.

Момент подписи всегда регистрируется с вычитанием часа из времени обновления, что позволит «собеседникам», часы которых отстают от наших, спокойно выполнять проверку.

Произведя поиск для имени *perfectstorm2.movie.edu* (хотя каким образом можно снять продолжение к *этому* фильму, я не знаю[1]), мы увидим следующее:

```
% dig +dnssec perfectstorm2.movie.edu.

; <<>> DiG 9.1.0 <<>> +dnssec perfectstorm2.movie.edu.
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 11232
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 4, ADDITIONAL: 3

;; OPT PSEUDOSECTION:
; EDNS: version:    0, udp=   4096
;; QUESTION SECTION:
;perfectstorm2.movie.edu.       IN      A

;; AUTHORITY SECTION:
movie.edu.              3600    IN      SOA     terminator.movie.edu.
al.robocop.movie.edu. 2001011601 10800 3600 604800 3600
movie.edu.              3600    IN      SIG     SOA 1 2 86400 20010215195456
20010116185456 27791 movie.edu. c1RwtgBX2SO8Q7Hz7vJD0aJfNfA6lsrqH4txHJI/
slRpx/UFYbnz3Gje N0JspZEihdLw0ZYMEiN6hnwRAzB4ag==
perfectstorm.movie.edu. 3600    IN      NXT     robocop.movie.edu. A SIG NXT
perfectstorm.movie.edu. 3600    IN      SIG     NXT 1 3 3600 20010215195456
20010116185456 27791 movie.edu.
qsB9l5AmSrB+qKmv+cKa+htCw84zwaakTmPC2yl+shzSEparrKwIMSR6 x5N69w8cze/
AW+gyFIwQZZkfZInJZA==
```

Обратите внимание на NXT-запись: она была автоматически добавлена при добавлении адресной записи *perfectstorm.movie.edu*, поскольку доменное имя *perfectstorm.movie.edu* было новым для зоны. Очень мило!

Однако несмотря на впечатляющий результат, следует проявлять осторожность, разрешая динамические обновления защищенной зоны. Следует использовать качественную идентификацию (скажем, TSIG) для проверки подлинности обновлений, потому что иначе у взломщика появится легкий способ изменить «защищенную» зону. Помимо этого, следует помнить, что сами по себе динамические обновления обрабатываются довольно быстро, но динамические обновления защищенной зоны требуют повторных вычислений для NXT-записей и, что более важно, ассиметричного шифрования (при вычислении новых SIG-записей). Таким образом, DNS-серверу потребуется больше системных ресурсов и больше времени для обработки таких запросов.

---

[1] Напоминаю, что все узлы гипотетического Университета Кинематографии авторы называют по названиям фильмов. В данном случае речь идет о фильме, известном в российском прокате как «Идеальный шторм». – *Примеч. ред.*

Смена ключей

Мы утверждали, что нет необходимости генерировать новый ключ для каждой подписи зоны, однако существуют случаи, в которых придется создать новый ключ, либо потому, что закрытый ключ исчерпал свою полезность, либо – что гораздо хуже – был «вычислен».

После определенного срока использования становится просто опасным продолжать подписывать записи тем же закрытым ключом. Чем больший объем информации, зашифрованной закрытым ключом, доступен злоумышленником, тем проще – с помощью криптоанализа – определить, какой именно используется ключ. Точного правила, позволяющего определить момент смены ключа, не существует, но мы приведем основные положения:

- Чем крупнее зона, тем большие объемы данных шифруются закрытым ключом при ее подписи. Если зона очень крупная, следует чаще менять ключи.
- Чем длиннее ключ, тем сложнее его подобрать. Длинные ключи можно менять реже.
- Чем чаще обновляются и подписываются данные зоны, тем больший объем информации, зашифрованной закрытым ключом, становится доступен. Если зона обновляется часто, в особенности если речь идет о динамических обновлениях защищенной зоны, то и ключи следует менять часто.
- Чем большую ценность для взломщика представляет подделка данных зоны, тем больше времени и денег он будет тратить на подбор закрытого ключа. Если целостность данных зоны особенно важна, следует менять ключи часто.

Поскольку мы обновляем зону *movie.edu* примерно раз в день, а сама зона не особенно крупная, то меняем пару ключей каждые шесть месяцев. В конце концов, мы же обычный университет. Если бы нас больше беспокоили данные зоны, мы использовали бы более длинные ключи и меняли бы их более часто.

К сожалению, переход к использованию нового ключа не столь прост – тут не обойтись простым созданием нового ключа и заменой старого. Если сделать это, DNS-серверы, кэшировавшие данные нашей зоны, не будут иметь возможности получить KEY-запись зоны и произвести проверку данных. Поэтому переход к новому ключу разбит на следующие этапы:

1. Создание новой пары ключей.
2. Создание набора ключей, содержащего новую KEY-запись и старую KEY-запись, и отправка этого набора администратору родительской зоны.
3. Создание набора ключей, содержащего только новую KEY-запись, и отправка этого набора администратору родительской зоны.

4. В случае использования *trusted-keys* следует добавить соответствующую запись к оператору для новой записи KEY. Следует также уведомить всех остальных, кто использует *trusted-keys*.
5. Следует включить подписанный набор ключей, полученный от администратора родительской зоны и содержащий обе записи KEY.
6. Необходимо прописать зональные данные новым закрытым ключом, но оставить старую запись KEY в зоне.
7. Когда все записи, подписанные предыдущим закрытым ключом, устареют, следует удалить предыдущую KEY-запись из зоны.
8. Следует включить подписанный набор ключей, полученный от администратора родительской зоны и содержащий только новую запись KEY.
9. Осталось только подписать зональные данные новым закрытым ключом.

Выполним эту процедуру. Сначала создадим новую пару ключей:

```
# dnssec-keygen -a RSA -b 512 -n ZONE movie.edu.
Kmovie.edu.+001+47703
```

Теперь создадим набор ключей, содержащий обе записи KEY, и отправим его администратору родительской зоны:

```
# dnssec-makekeyset -t 172800 Kmovie.edu.+001+27791.key
Kmovie.edu.+001+47703.key
# mail -s "Sign my keys, please" hostmaster@nsiregistry.net < keyset-
movie.edu
# mv keyset-movie.edu keyset-movie.edu.2key
```

и набор ключей, содержащий только новую запись KEY, который также отправим администратору родительской зоны:

```
% dnssec-makekeyset -t 172800 Kmovie.edu.+001+47703.key
% mail -s "Sign my keys, please" hostmaster@nsiregistry.net < keyset-
movie.edu
```

(По правде говоря, потребуется гораздо больше, чем эти два письма, чтобы кто-то наконец подписал наши ключи.)

Первый набор ключей содержит обе KEY-записи и SIG-записи для них:

```
$ORIGIN .
$TTL 172800     ; 2 days
movie.edu               IN SIG  KEY 1 2 172800 20010104060917 (
                                20001205060917 27791 movie.edu.
                                RyNYoZ/kOtHqnFhUiVs2yjJWPFNeP8BKZ/Jaw+7xO9Jl
                              ZwJN2ZYQjVNVGLk3OrJlxQRjCCdaaYQSq8u81up3xw== )
                        IN SIG  KEY 1 2 172800 20010104060917 (
                                20001205060917 47703 movie.edu.
                                 1JGNBQydq6U+qKfq1wxfu1nsu283Zf7mNDDmuBtuuB7o
                              lwaeBL96tzBKpMUAcDYXsM8zxiStF+wTY+I5wfgevA== )
```

```
                                KEY     256 3 1 (
                                         AQPdWbrGbVv1eDhNgRhpJMPonJfA3reyEo82ekwRnjbX
                                        7+uBxB11BqL7LAB7/C+eb0vCtI53FwMhkkNkTmA6bI8B )
                                KEY     256 3 1 (
                                         AQPjAfGtDkx6PSgYFs6G2vY1bJEldAMudngn6ZpGJnyg
                                        k+LeU5PYiYd48wFLimihjyzlTWMb+C+egtQGpDVQulez )
```

Обратите внимание, что одна из SIG-записей была создана ключом со старой картой 27791 (прежний закрытый ключ), а вторая ключом с картой 47703 (новый закрытый ключ). Это доказывает, что мы обладаем парой соответствующих закрытых ключей.

Получив ответ от администратора родительской зоны, мы сохраняем данные в каталоге */var/named* под именем *movie.edu.signedkey*, то есть в файле, который включаем с помощью директивы $INCLUDE в *db.movie.edu.signed*. Вот так выглядит файл *movie.edu.signedkey*:

```
$ORIGIN .
$TTL 172800     ; 2 days
movie.edu               IN SIG  KEY 1 2 172800 20010104060917 (
                                 20001205060917 65398 edu.
                                 qzvmuTVv9yGZf963ZuN2jxk8brEX/VP3sI5pOM/g2mU/
                                EPa57fyhHDNo7ny8Q2Su5vXnAIoxaaKAR8VmognQ7A== )
                        KEY     256 3 1 (
                                 AQPdWbrGbVv1eDhNgRhpJMPonJfA3reyEo82ekwRnjbX
                                7+uBxB11BqL7LAB7/C+eb0vCtI53FwMhkkNkTmA6bI8B )
                        KEY     256 3 1 (
                                 AQPjAfGtDkx6PSgYFs6G2vY1bJEldAMudngn6ZpGJnyg
                                k+LeU5PYiYd48wFLimihjyzlTWMb+C+egtQGpDVQulez )
```

SIG-запись *edu* относится к обеим KEY-записям, поэтому можно использовать любой из ключей для подписи зональных данных.

Теперь подпишем зональные данные, используя *только* новый закрытый ключ:

```
# dnssec-signzone -o movie.edu. db.movie.edu Kmovie.edu.+001+47703.private
```

Программе *dnssec-signzone* не нравится повторно подписывать зоны, уже подписанные другим ключом, поэтому мы начали опять с обычной версии зоны *movie.edu*. Вот фрагмент подписанного файла данных зоны, *db.movie.edu.signed*:

```
$ORIGIN .
$TTL 86400      ; 1 день
movie.edu               IN SOA  terminator.movie.edu. al.robocop.movie.edu. (
                           2000092603 ; порядковый номер
                           10800      ; обновление (3 часа)
                           3600       ; повторение (1 час)
                           604800     ; устаревание (неделя)
                           3600       ; минимум (1 час)
                           )
```

```
                        SIG     SOA 1 2 86400 20010104062430 (
                                20001205062430 47703 movie.edu.
                                LIsndGD5q2VPWb+HaOffFP54UE6RYPweqtTp1xhgw4B9
                              Pyb/7z54J8q8LCONmzQ6SthnfecBQhDBpc72HfNeJQ== )
                        NS      terminator.movie.edu.
                        NS      wormhole.movie.edu.
                        NS      outland.fx.movie.edu.
                        SIG     NS 1 2 86400 20010104062430 (
                                20001205062430 47703 movie.edu.
                                Ktq2mYMzTrBfGjdSb2F7ghyh2nXaLcOiTPV4k8I64jlO
                              nJt/hsBZPpeyM2u+Zymvp3mJMWg66E4tirjOAvlGXw== )
                        SIG     MX 1 2 86400 20010104062430 (
                                20001205062430 47703 movie.edu.
                                20001205062430 47703 movie.edu.
                                l/XnJ+JWhmAZLp6YF27YQQlOyT7iZOqGDXPw86OP6U1H
                              NmgDkUKoHfD6CdYwpKz15NyxRKilVmx2ne3oBOTUEQ== )
$TTL 3600       ; 1 час
                        SIG     NXT 1 2 3600 20010104062430 (
                                20001205062430 47703 movie.edu.
                                2sxN3rQXn/JklugmyGV+on1Io6tV1wEYP6m4oD1xHCP1
                              +NHPR+uT2IknW8SvGc3Kaj16kb2Ej+i3RvleWSI4Tg== )
                        NXT     bigt.movie.edu. ( NS SOA MX SIG KEY NXT )
$TTL 172800     ; 2 дня
                        SIG     KEY 1 2 172800 20010104060917 (
                                20001205060917 65398 edu.
                                qzvmuTVv9yGZf963ZuN2jxk8brEX/VP3sI5pOM/g2mU/
                              EPa57fyhHDNo7ny8Q2Su5vXnAIoxaaKAR8VmognQ7A== )
                        KEY     256 3 1 (
                                AQPdWbrGbVv1eDhNgRhpJMPonJfA3reyEo82ekwRnjbX
                              7+uBxB11BqL7LAB7/C+ebOvCtI53FwMhkkNkTmA6bI8B )
                        KEY     256 3 1 (
                                AQPjAfGtDkx6PSgYFs6G2vY1bJEldAMudngn6ZpGJnyg
                              k+LeU5PYiYd48wFLimihjyzlTWMb+C+egtQGpDVQulez )
$TTL 86400      ; 1 день
                        MX      10 postmanrings2x.movie.edu.
```

Хотя зона включает две KEY-записи, а также SIG-запись *edu*, которая относится к обеим, прочие записи зоны были подписаны только новым закрытым ключом – с картой 47703.

Второй подписанный набор ключей нужен при удалении старой записи KEY: SIG-запись, посланная нам из зоны *edu*, и относящаяся к обеим KEY-записям, становится бесполезной. Но если мы используем набор ключей, содержащий только новую запись KEY, все будет в порядке.

Подозреваем, что, прочитав все это, читатели примут решение использовать самые длинные ключи, какие только возможно, чтобы никогда не пришлось их менять.

К чему все это?

Мы понимаем, что DNSSEC выглядит немного, скажем так, устрашающе. (Нам стало плохо, когда мы впервые столкнулись с этим механизмом.) Но в основе его разработки лежит очень и очень важная идея: максимально затруднить подделку пакетов DNS. В наше время все большее распространение получает использование сети Интернет для ведения дел, а в этом случае очень важно быть уверенным, что попадешь именно туда, куда собирался.

Тем не менее, мы осознаем, что DNSSEC и прочие меры обеспечения безопасности, описанные в этой главе, нужны далеко не всем читателям. Следует соотносить потребность в защищенности и стоимость реализации защиты – в плане ноши, которую эта реализация возлагает на инфраструктуру и продуктивность администратора.

12

nslookup и dig

- *Насколько хорош nslookup?*
- *Пакетный или диалоговый?*
- *Настройка*
- *Как отключить список поиска*
- *Основные задачи*
- *Прочие задачи*
- *Разрешение проблем с nslookup*
- *Лучшие в сети*
- *Работа с dig*

– Что это ты там бормочешь? – спросил Шалтай, впервые прямо взглянув на нее. – Скажи-ка мне лучше, как тебя зовут и зачем ты сюда явилась.
– Меня зовут Алиса, а...
– Какое глупое имя, – нетерпеливо прервал ее Шалтай-Болтай. – Что оно значит?
– Разве имя должно что-то значить? – проговорила Алиса с сомнением.
– Конечно, должно, – ответил Шалтай-Болтай и фыркнул.

Чтобы стать экспертом в решении проблем, связанных с DNS-серверами, необходим отладочный инструмент, который позволяет выполнять DNS-запросы, причем инструмент достаточно гибкий. В этой главе мы изучим *nslookup*, поскольку эта программа распространяется в комплекте BIND и доступна на многих операционных системах. Разумеется, это не значит, что *nslookup* – самый лучший из существующих для отладки инструментов. У *nslookup* есть недостатки, и их так много, что использование этого инструмента в BIND 9 в настоящее время осуждается («deprecated», на человеческом языке – «официально не в почете»). Тем не менее, мы рассмотрим *nslookup*, поскольку это вездесущая программа. Затем мы перейдем к инструменту *dig*, который обеспечивает схожую функциональность и не страдает от недостатков, присущих *nslookup*.

Помните, что эта глава не всеобъемлюща, существуют аспекты *nslookup* и *dig* (по большей части слабо понятные и редко используемые), которые оставлены за кадром. Любознательные читатели всегда найдут эту информацию в соответствующих страницах руководства.

Насколько хорош nslookup?

По большей части *nslookup* будет использоваться для отправки запросов тем же способом, какой используется DNS-клиентом. Но иногда *nslookup* может применяться для отправки запросов DNS-серверам, как это делает не клиент, но собственно DNS-сервер. Способ применения этой программы зависит от проблемы, которую необходимо решить. Читатели спросят: «Насколько точно *nslookup* эмулирует DNS-клиент или DNS-сервер? Использует ли функции библиотеки клиента BIND?» Нет, *nslookup* использует собственные функции для посылки запросов DNS-серверам, но эти функции основаны на функциях клиента. Как следствие поведение *nslookup* очень схоже с поведением клиента, но немного отличается. Мы будем обращать внимание читателей на эти различия. Что касается эмуляции поведения DNS-сервера, *nslookup* позволяет посылать DNS-серверу запросы, которые абсолютно не отличаются от посылаемых DNS-серверами, но при этом используется другая схема передачи запросов. При этом, как и DNS-сервер, *nslookup* умеет производить передачу копии данных зоны. Итак, *nslookup* не эмулирует в точности ни клиент, ни DNS-сервер, но эмулирует их в достаточной степени, чтобы служить достойным инструментом для диагностики. Теперь рассмотрим различия, о которых мы только что упоминали.

Множественные серверы

nslookup умеет общаться только с одним DNS-серверов единовременно. В этом заключается его самое большое отличие от DNS-клиента. Клиент использует все инструкции *nameserver* из файла *resolv.conf*. Если в *resolv.conf* присутствуют две инструкции *nameserver*, клиент посылает запрос сначала первому серверу, затем второму, затем снова первому, и снова второму, и так, пока не будет получен ответ, либо не откажется от дальнейших попыток. Процесс повторяется для каждого запроса. С другой стороны, *nslookup* пытается получить ответ от первого из серверов, перечисленных в *resolv.conf*, пока не потеряет всякую надежду, а затем переходит ко второму серверу. Получив ответ, программа фиксируется на одном из серверов и не посылает запросы второму. При этом *желательно*, чтобы диагностирующий инструмент работал только с одним DNS-сервером, в целях сокращения числа переменных, участвующих в анализе проблемы. Если бы *nslookup* работал с несколькими DNS-серверами, мы в значительно меньшей степени могли бы управлять сеансами диагностики. Таким образом, для диагностирующего инструмента фиксация на единственном сервере совершенно естественна.

Интервалы ожидания

Интервалы ожидания *nslookup* соответствуют интервалам ожидания при работе клиента с единственным DNS-сервером. Интервалы ожида-

ния для DNS-сервера, при этом, основаны на том, насколько быстро DNS-сервер ответил на последний запрос, то есть являются динамически изменяющимися величинами. Интервалы ожидания *nslookup* никогда не совпадают с интервалами ожидания DNS-серверов, но это также не является проблемой. При посылке запросов удаленным DNS-серверам с помощью *nslookup* необходимо знать лишь результат, а не время, потраченное на поиск ответа.

Список поиска

nslookup точно так же, как и клиент, реализует список поиска. Версии *nslookup*, поставляемые в составе BIND версий до 4.9, в основном используют «полный» список поиска: локальное доменное имя и имена всех предков, состоящие не менее чем из двух меток. Версии *nslookup*, поставляемые в составе BIND версии 4.9 и более поздних, используют сокращенный список поиска, который содержит только локальное доменное имя. Мы покажем, как определить эту характеристику *nslookup* в случае, если существует такая необходимость.

Список поиска не реализуется DNS-серверами, поэтому для работы в качестве DNS-сервера *nslookup* следует проинформировать о необходимости отключить список поиска (подробности чуть позже).

Передача зоны

nslookup производит передачу зоны в точности, как DNS-сервер. Но в отличие от DNS-сервера *nslookup* не проверяет порядковые номера в SOA-записях перед получением данных зоны; в случае, если существует такая необходимость, проверять порядковые номера придется вручную.

Использование NIS и файла /etc/hosts

В этом последнем разделе мы не сравниваем *nslookup* с клиентом или DNS-сервером, но изучаем способы поиска по именам в целом. Как инструмент, поставляемый концорциумом ISC, *nslookup* использует только DNS; он не будет работать с NIS или файлом */etc/hosts*. Большинство приложений способны использовать DNS, NIS или */etc/hosts*, в зависимости от настройки системы. Не надейтесь, что *nslookup* поможет обнаружить проблемы, если узел не настроен на использование DNS-серверов.[1]

---

[1] Это возможно в случаях, когда поставщик усовершенствовал *nslookup* для работы с серверами NIS и файлом */etc/hosts*; именно так дела обстоят в системе HP-UX.

Пакетный или диалоговый?

Начнем изучение *nslookup* с запуска и завершения работы с программы. *nslookup* можно запускать в пакетном режиме либо в диалоговом. Если необходимо найти одну запись для доменного имени, воспользуйтесь пакетным режимом. Если планируется выполнение более сложных действий, в частности изменение используемых DNS-серверов или настроек, следует работать в диалоговом режиме.

Чтобы запустить программу в режиме диалога, следует просто набрать *nslookup*:

```
% nslookup
Default Server:  terminator.movie.edu
Address:  0.0.0.0

> ^D
```

Для получения справки можно ввести *?* или *help*.[1] Для завершения работы следует набрать ^*D* (Ctrl-D), либо *exit*. Если попытаться завершить работу с *nslookup* прерыванием по ^*C* (или по другому символу, определенному для прерываний), то желаемого результата не будет. *nslookup* перехватывает сигнал прерывания, прекращает текущую операцию (например, передачу зоны) и выдает приглашение >.

Для поиска в пакетном режиме следует указывать искомое имя в командной строке:

```
% nslookup carrie
Server:  terminator.movie.edu
Address:  0.0.0.0

Name:    carrie.movie.edu
Address: 192.253.253.4
```

Настройка

nslookup имеет собственный набор верньеров и переключателей, то есть *настроек*. Все настройки могут изменяться. Сейчас мы обсудим существующие настройки, а в оставшейся части главы продемонстрируем их использование.

```
% nslookup
Default Server:  bladerunner.fx.movie.edu
Address:  0.0.0.0

> set all
Default Server:  bladerunner.fx.movie.edu
Address:  0.0.0.0
```

[1] Функция справки не реализована в *nslookup* пакета BIND 9 (в версии 9.1.0).

```
Set options:
nodebug         defname         search          recurse
nod2            novc            noignoretc      port=53
querytype=A     class=IN        timeout=5       retry=4
root=a.root-servers.net.
domain=fx.movie.edu
srchlist=fx.movie.edu

> ^D
```

Небольшое вступление, прежде чем мы перейдем непосредственно к настройкам. DNS-сервер по умолчанию – *bladerunner.fx.movie.edu*. Это означает, что *nslookup* будет посылать запросы узлу *bladerunner*, если явным образом не указать другой DNS-сервер. Адрес 0.0.0.0 означает «этот узел». Когда *nslookup* в качестве адреса DNS-сервера использует 0.0.0.0 или 127.0.0.1, речь идет о DNS-сервере, работающем на локальной системе, в данном случае – на узле *bladerunner*.

Существуют настройки двух типов: переключатели и принимающие значения. Переключателям не присваиваются значения с помощью знака равенства. Множество значений переключателя ограничивается положениями «включено» и «выключено». Настройкам, принимающим значения, могут присваиваться... значения. Как определить, какие переключатели включены, а какие выключены? Настройка выключена, если ее имя предваряется словом «no». Так, *nodebug* означает, что отладка выключена. Как можно догадаться, переключатель *search* включен.

Способ изменения значений настроек зависит от того, используется *nslookup* в пакетном режиме или в диалоговом. В диалоговом сеансе настройку можно изменить с помощью команды *set* (*set debug* или *set domain=classics.movie.edu*). В командной строке следует опускать ключевое слово *set* и предварять имя параметра дефисом (*nslookup -debug* или *nslookup -domain=classics.movie.edu*). Параметры могут сокращаться до наиболее краткой, уникальной формы. Так, *nodeb* является допустимым сокращением *nodebug*. Помимо сокращения, имя настройки *querytype* может записываться как *type*.

Рассмотрим каждую из настроек:

[no]debug

Отладка по умолчанию выключена. При включенной отладке DNS-сервер отображает интервалы ожидания и ответные сообщения. См. также описание второго уровня отладки (*[no]d2*).

[no]defname

По умолчанию *nslookup* добавляет локальное доменное имя к именам, не содержащим точек. До того как появился список поиска, клиент BIND добавлял к именам, не содержащим точек, только локальное доменное имя; описываемая настройка отражает этот факт. *nslookup* может вести клиент до появления списка поиска (*se-*

arch выключен, *defname* включен) или клиент со списком поиска (*search* включен).

[no]search

Настройка search заменяет настройку для локального доменного имени (*defname*). То есть *defname* имеет силу только в том случае, если переключатель *search* выключен. По умолчанию *nslookup* добавляет доменные имена из списка поиска (*srchlist*) к именам, которые не заканчиваются точкой.

[no]recurse

По умолчанию *nslookup* создает рекурсивные запросы. В сообщениях устанавливается бит рекурсии. Клиент BIND посылает рекурсивные запросы таким же образом. Но DNS-серверы посылают другим DNS-серверам нерекурсивные запросы.

[no]d2

Отладка второго уровня по умолчанию выключена. Если включить отладку, то в дополнение к стандартной диагностике будут отображаться посылаемые сообщения-запросы. Включение *d2* также автоматически включает *debug*. Выключение *d2* выключает только *d2*; *debug* остается включенным. Выключение *debug* выключает и *debug*, и *d2*.

[no]vc

По умолчанию *nslookup* посылает запросы в UDP-дейтаграммах, а не через TCP-соединение. Большинство клиентов BIND посылают запросы по UDP, поэтому стандартное поведение *nslookup* совпадает с поведением клиента. Клиенту можно предписать использование TCP, то же самое касается программы *nslookup*.

[no]ignoretc

По умолчанию *nslookup* не игнорирует усеченные сообщения. Если бит «усечения» в полученном сообщении установлен – это значит, что DNS-сервер не смог уместить всю значимую информацию в UDP-дейтаграмму ответа – *nslookup* повторно посылает запрос, используя TCP-соединение. Это поведение также соответствует поведению клиента BIND. Смысл повторения запроса с использованием TCP-соединения заключается в том, что размер TCP-ответа может многократно превышать размер UDP-ответа.

port=53

DNS-сервер принимает соединения через порт 53. Для целей отладки можно запустить DNS-сервер на другом порту, а затем предписать программе *nslookup* использовать этот порт для запросов.

querytype=A

По умолчанию *nslookup* производит поиск A (адресных) RR-записей. Помимо этого, если указать IP-адрес (и тип запроса A или

PTR), *nslookup* обращает адрес, добавляет *in-addr.arpa* и производит поиск PTR-записей.

class=IN

Единственный класс, имеющий какой-то смысл, – Интернет (IN). Ну, возможно еще класс Hesiod (HS), для людей из Массачусетского технологического института или пользователей Ultrix.

timeout=5

Если DNS-сервер не ответил в пределах 5 секунд, *nslookup* посылает запрос повторно и удваивает интервал ожидания (до 10, 20, а затем до 40 секунд). Большинство клиентов BIND используют такие же значения при работе с одним DNS-сервером.

retry=4

Посылать запрос не более четырех раз. После каждой попытки значение интервала ожидания удваивается. Это соответствует поведению большинства версий клиента BIND.

root=a.root-servers.net.

Существует удобная команда *root*, которая позволяет изменить DNS-сервер на указанный. Выполнение команды *root* в командной строке современной версии *nslookup* равноценно выполнению команды *server a.root-servers.net*. В более старых версиях указывалось имя корневого DNS-сервера *nic.ddn.mil* (в умеренно старых) или даже *sri-nic.arpa* (в древних). «Корневой» сервер по умолчанию можно изменить с помощью команды *set root=server*.

domain=fx.movie.edu

Стандартное доменное имя, добавляемое при включенном *defname*.

srchlist=fx.movie.edu

Если *search* включен, перечисленные здесь доменные имена добавляются к именам, которые не заканчиваются точкой. Доменные имена перечисляются в порядке предпочтения, и разделяются символом слэша. (В *nslookup* из BIND версии 4.8.3 список поиска по умолчанию содержал бы значение *fx.movie.edu/movie.edu*. Начиная с версии 4.9 список поиска программы *nslookup* стандартно содержит только доменное имя по умолчанию.[1] Следует явно определить список поиска в файле */etc/resolv.conf*, чтобы одновременно использовать *fx.movie.edu* и *movie.edu*.)

[1] Это позволяет легко определить, какая версия *nslookup* используется: следует выполнить *set all* и проверить, содержит список поиска по умолчанию только локальное доменное имя (BIND версии 4.9 или более поздней) или также и доменные имена предков (BIND версии 4.8.3 или более ранней).

Файл .nslookuprc

Собственные стандартные настройки программы *nslookup* можно определить в файле *.nslookuprc*. *nslookup* при запуске производит поиск файла *.nslookuprc* в домашнем каталоге пользователя, это справедливо как для пакетного, так и для диалогового режима. Файл *.nslookuprc* может содержать любые корректные команды *set*, по одной в строке. Это полезно, к примеру, в случае, когда старая версия *nslookup* считает *sri-nic.arpa* корневым DNS-сервером. Корневой DNS-сервер, используемый по умолчанию, можно определить с помощью следующей строки в файле *.nslookuprc*:

```
set root=a.root-servers.net.
```

.nslookuprc можно также использовать для определения списка поиска, отличного от стандартного списка, либо для изменения используемых *nslookup* интервалов ожидания.

Как отключить список поиска

nslookup реализует список поиска, аналогичный существующему в клиенте. Однако при отладке список поиска может просто мешать. Может возникнуть необходимость запретить использование списка поиска (*set nosearch*) либо использовать при поиске абсолютные доменные имена, заканчивающиеся точкой. Как видно из примеров, мы предпочитаем второй вариант.

Основные задачи

Существует несколько повседневных задач, которые приходится практически ежедневно решать с помощью *nslookup*: поиск IP-адреса или MX-записей для определенного доменного имени или запрос данных у конкретного DNS-сервера. Сначала мы рассмотрим основные задачи, а потом перейдем к прочим, которые приходится решать гораздо реже.

Поиск записей различного типа

По умолчанию *nslookup* производит поиск адреса для доменного имени, либо доменного имени для адреса. Запись любого типа можно найти, изменяя значение *querytype*, как показано в следующем примере:

```
% nslookup
Default Server:  terminator.movie.edu
Address:  0.0.0.0

> misery                    — поиск адреса
Server:  terminator.movie.edu
Address:  0.0.0.0
```

```
Name:    misery.movie.edu
Address:  192.253.253.2

> 192.253.253.2            — поиск доменного имени
Server:  terminator.movie.edu
Address:  0.0.0.0

Name:    misery.movie.edu
Address:  192.253.253.2

> set q=mx                 — поиск MX-записей
> wormhole
Server:  terminator.movie.edu
Address:  0.0.0.0

wormhole.movie.edu      preference = 10, mail exchanger = wormhole.movie.edu
wormhole.movie.edu      internet address = 192.249.249.1
wormhole.movie.edu      internet address = 192.253.253.1

> set q=any                — поиск записей любого типа
> diehard
Server:  terminator.movie.edu
Address:  0.0.0.0

diehard.movie.edu       internet address = 192.249.249.4
diehard.movie.edu       preference = 10, mail exchanger = diehard.movie.edu
diehard.movie.edu       internet address = 192.249.249.4
```

Разумеется, существует не так уж и много типов записей DNS. Более полный список приводится в приложении А «Формат сообщений DNS и RR-записей».

Авторитативные ответы против неавторитативных

Те из читателей, кто уже использовал *nslookup* прежде, могли заметить, что при первом поиске по внешнему доменному имени ответ является авторитативным, а при повторном – неавторитативным. Приведем пример:

```
% nslookup
Default Server:  relay.hp.com
Address:  15.255.152.2

> slate.mines.colorado.edu.
Server:  relay.hp.com
Address:  15.255.152.2

Name:     slate.mines.colorado.edu
Address:  138.67.1.3

> slate.mines.colorado.edu.
Server:  relay.hp.com
Address:  15.255.152.2
```

```
Non-authoritative answer:
Name:    slate.mines.colorado.edu
Address:  138.67.1.3
```

И ничего странного в этом нет. Дело в том, что когда локальный DNS-сервер в первый раз производит поиск для имени *slate.mines.colorado.edu*, то связывается с DNS-сервером зоны *mines.colorado.edu*, и сервер *mines.colorado.edu* авторитативно отвечает на вопрос. По сути дела, локальный DNS-сервер возвращает авторитативный ответ прямо программе *nslookup*. Помимо этого, ответ кэшируется. При повторном поиске для *slate.mines.colorado.edu* DNS-сервер извлекает кэшированный ответ и возвращает его в качестве «неавторитативного».[1]

Обратите внимание, что мы завершаем доменные имена точкой при каждом поиске. Если бы мы этого не делали, то получали бы точно такой же ответ. В некоторых случаях важно не забывать использовать абсолютные имена при отладке, а в некоторых это не имеет значения. Поэтому, чтобы не тратить время на выяснение, нужно ли использовать точку в *этом* имени, мы всегда добавляем точку, если известно, что имя абсолютно, за исключением, разумеется, случаев, когда отключено использование списка поиска.

Переключение между DNS-серверами

Иногда появляется необходимость послать прямой запрос другому DNS-серверу – к примеру, если возникло подозрение, что сервер ведет себя некорректно. Изменить используемый DNS-сервер в *nslookup* можно с помощью *lserver*. Разница между *server* и *lserver* в том, что *lserver* посылает запрос «локальному» DNS-серверу – тому, с которого все началось – в целях получения адреса сервера, на который происходит переключение; *server* использует DNS-сервер по умолчанию вместо локального. Это различие важно, поскольку сервер, на который произошло переключение может не отвечать:

```
% nslookup
Default Server:  relay.hp.com
Address:  15.255.152.2
```

При запуске первый DNS-сервер, *relay.hp.com*, становится сервером команды *lserver*. Это имеет значение для описываемого сеанса работы.

```
> server galt.cs.purdue.edu.
Default Server:  galt.cs.purdue.edu
Address:  128.10.2.39

> cs.purdue.edu.
Server:  galt.cs.purdue.edu
```

[1] Интересно, что серверы имен BIND 9 представляют даже первый ответ как неавторитативный.

```
Address:  128.10.2.39

*** galt.cs.purdue.edu can't find cs.purdue.edu.: No response from server
```

В этот момент мы пытаемся переключиться обратно, на исходный DNS-сервер. Но на узле *galt.cs.purdue.edu* нет DNS-сервера, который позволил бы найти адрес *relay.hp.com*:

```
> server relay.hp.com.

*** Can't find address for server relay.hp.com.: No response from server
```

Чтобы не застрять, мы используем команду *lserver*, чтобы с помощью локального DNS-сервера найти адрес *relay.hp.com*:

```
> lserver relay.hp.com.
Default Server:  relay.hp.com
Address:  15.255.152.2

> ^D
```

Поскольку DNS-сервер на узле *galt.cs.purdue.edu* не ответил (ведь DNS-сервер на этом узле вообще отсутствует), то не было возможности найти адрес узла *relay.hp.com*, чтобы переключиться на работу с DNS-сервером на *relay*. И здесь на помощь приходит команда *lserver*: локальный DNS-сервер, *relay*, по-прежнему отвечал на запросы, и мы воспользовались этим обстоятельством. Мы могли бы также не применять *lserver*, а восстановить равновесие прямым указанием IP-адреса узла *relay – server 15.255.152.2*.

Существует также возможность изменять используемые DNS-серверы для отдельных запросов. Чтобы объяснить *nslookup*, что запрос по конкретному доменному имени следует посылать определенному DNS-серверу, следует указать DNS-сервер в качестве второго аргумента, после доменного имени, для которого ведется поиск:

```
% nslookup
Default Server:  relay.hp.com
Address:  15.255.152.2

> saturn.sun.com. ns.sun.com.
Name Server:  ns.sun.com
Address:  192.9.9.3

Name:    saturn.sun.com
Addresses: 192.9.25.2

> ^D
```

И, разумеется, серверы можно переключать при запуске *nslookup* из командной строки. Сервер, которому посылается запрос, можно указать после доменного имени, для которого создается этот запрос:

```
% nslookup -type=mx fisherking.movie.edu. terminator.movie.edu.
```

Такая команда является указанием *nslookup* посылать запрос DNS-серверу *terminator.movie.edu* на предмет получения MX-записей для имени *fisherking.movie.edu*.

Наконец, чтобы указать альтернативный DNS-сервер и перейти в диалоговый режим, можно передать программе *nslookup* дефис вместо доменного имени:

```
% nslookup - terminator.movie.edu.
```

Прочие задачи

Перейдем к фокусам, которые используются реже, но иногда оказываются довольно полезными. Большинство из них будут полезны при диагностировании проблем DNS или BIND; они позволят покопаться в сообщениях, доступных клиенту, а также притвориться DNS-сервером BIND, посылающим запрос другому серверу, или производящим передачу зоны.

Отображение сообщений-запросов и сообщений-ответов

При необходимости можно предписать *nslookup* отображение посылаемых запросов и получаемых ответных сообщений. Для отображения ответов необходимо включить *debug*. Для отображения запросов и ответов – *d2*. При необходимости отключить отладку полностью следует использовать *set nodebug*, поскольку *set nod2* отключает только отладку второго уровня. Для приводимого фрагмента мы сделаем несколько комментариев. Особенно любопытные читатели могут стряхнуть пыль со своих копий документа RFC 1035, открыть страницу 25 и читать ее параллельно с нашими объяснениями.

```
% nslookup
Default Server:  terminator.movie.edu
Address:  0.0.0.0

> set debug
> wormhole
Server:  terminator.movie.edu
Address:  0.0.0.0

------------
Got answer:
    HEADER:
        opcode = QUERY, id = 6813, rcode = NOERROR
        header flags:  response, auth. answer, want recursion,
        recursion avail.  questions = 1,  answers = 2,
        authority records = 2,  additional = 3
```

```
    QUESTIONS:
        wormhole.movie.edu, type = A, class = IN
    ANSWERS:
    ->  wormhole.movie.edu
        internet address = 192.253.253.1
        ttl = 86400 (1D)
    ->  wormhole.movie.edu
        internet address = 192.249.249.1
        ttl = 86400 (1D)
    AUTHORITY RECORDS:
    ->  movie.edu
        nameserver = terminator.movie.edu
        ttl = 86400 (1D)
    ->  movie.edu
        nameserver = wormhole.movie.edu
        ttl = 86400 (1D)
    ADDITIONAL RECORDS:
    ->  terminator.movie.edu
        internet address = 192.249.249.3
        ttl = 86400 (1D)
    ->  wormhole.movie.edu
        internet address = 192.253.253.1
        ttl = 86400 (1D)
    ->  wormhole.movie.edu
        internet address = 192.249.249.1
        ttl = 86400 (1D)

------------
Name:    wormhole.movie.edu
Addresses:  192.253.253.1, 192.249.249.1

> set d2
> wormhole
Server:  terminator.movie.edu
Address:  0.0.0.0
```

This time the query is also shown.

```
------------
SendRequest(), len 36
    HEADER:
        opcode = QUERY, id = 6814, rcode = NOERROR
        header flags:  query, want recursion
        questions = 1,  answers = 0,  authority records = 0,
       additional = 0

    QUESTIONS:
        wormhole.movie.edu, type = A, class = IN

------------
------------
Got answer (164 bytes):
```

The answer is the same as above.

Строки дефисов разделяют сообщения-запросы и сообщения-ответы. Сейчас, как и было обещано, мы разберем содержимое сообщений. Пакеты DNS состоят из пяти разделов: заголовка (Header), вопроса (Question), ответа (Answer), авторитативности (Authority) и вторичного (Additional).

Раздел заголовка

Раздел заголовка присутствует в каждом запросе или ответном сообщении. Код операции, о котором сообщает *nslookup*, всегда имеет значение QUERY. Существуют другие коды операций: для асинхронных уведомлений об изменении зональных данных (NOTIFY) и динамических обновлений (UPDATE), но *nslookup* их не встречает, поскольку посылает обычные запросы и получает стандартные ответы.

Поле ID в заголовке используется для связывания ответного сообщения с запросом и обнаружения дублирующихся запросов или ответов. Чтобы определить, какое сообщение является запросом, а какое ответом, следует изучить флаг заголовка. Строка *want recursion* означает, что запрос рекурсивный. В ответе значение этого флага копируется. Строка *auth. answer* идентифицирует авторитативный ответ. Другими словами, ответ получен из данных авторитативного DNS-сервера, а не из кэша. Код ответного сообщения, *rcode*, может иметь значения: *no error* (нет ошибок), *server failure* (ошибка сервера), *name error* (ошибка в имени, известная также как *nxdomain* или *nonexistent domain* – несуществующий домен), *not implemented* (реализация отсутствует) или *refused* (отказ). Коды *server failure*, *name error*, *not implemented* и *refused* приводят к выдаче программой *nslookup* сообщений «Server failed», «Nonexistent domain», «Not implemented» и «Query refused», соответственно. Последние четыре записи в разделе заголовка представляют собой счетчики, определяющие, сколько RR-записей содержится в каждом из четырех последующих разделов.

Раздел вопроса

В сообщении DNS всегда присутствует один вопрос; он состоит из доменного имени, типа запрошенных данных и класса. В сообщении DNS не может присутствовать более одного вопроса – возможность обработки нескольких вопросов потребовала бы переработки формата сообщений. Для начала потребовалось бы заменить единственный бит авторитативности другой структурой данных, поскольку раздел ответа мог бы содержать набор авторитативных и неавторитативных ответов. В существующей структуре установка бита авторитативного ответа означает, что сервер является авторитативным для зоны, которая содержит доменное имя, указанное в вопросе.

Раздел ответа

Данные раздел содержит RR-записи, которые являются ответом на вопрос. В ответе может содержаться несколько RR-записей. К при-

меру, если узел входит в несколько сетей, может присутствовать несколько адресных записей.

Раздел авторитативности

В раздел авторитативности возвращаются записи DNS-серверов (NS-записи). Когда ответ является перенаправлением к другим DNS-серверам, координаты этих DNS-серверов перечислены в данном разделе.

Вторичный раздел

Данный раздел содержит информацию, дополняющую записи других разделов. К примеру, если DNS-сервер упомянут в разделе авторитативности, его адрес может присутствовать в дополнительном разделе. В конце концов, чтобы связаться с DNS-сервером, необходимо иметь его адрес.

Специально для педантичных читателей отметим, что существует вариант, при котором число вопросов в сообщении DNS не равно одному: в инверсных запросах это число равно нулю. Инверсный запрос содержит один ответ, а раздел вопроса пуст. DNS-сервер заполняет раздел вопроса. Но как мы уже говорили, инверсные запросы занесены в красную книгу.

Маскировка под DNS-сервер BIND

Можно заставить *nslookup* посылать точно такие же запросы, какие посылает DNS-сервер. Начнем с того, что запросы DNS-сервера не так уж сильно отличаются от сообщений-запросов клиента. Основное различие заключается в том, что клиент запрашивает рекурсивное разрешение, а DNS-сервер практически никогда этого не делает. Запрос рекурсии по умолчанию вставляется в сообщения, создаваемые программой *nslookup*, поэтому его необходимо будет отключить явным образом. Различие в работе клиента и DNS-сервера состоит в том, что клиент использует список поиска, а DNS-сервер не использует. Список поиска используется в *nslookup* по умолчанию, поэтому его также следует отключить явным образом. Разумеется, продуманное использование точки в конце имени будет иметь тот же результат.

В приземленных терминах *nslookup* все это означает, что создание запросов в стиле клиента происходит при применении программы со стандартными настройками. Чтобы предписать создание запросов в стиле DNS-сервера, следует использовать команды *set norecurse* и *set nosearch*. Для командной строки это будет выглядеть следующим образом: *nslookup -norecurse -nosearch*.

Когда DNS-сервер BIND получает запрос, то ищет ответ в данных, для которых он авторитативен, а также в кэше. Если ответа на запрашиваемые данные нет ни в кэше, ни в авторитативных данных, то DNS-сервер отвечает, что имя не существует либо что отсутствуют записи за-

прошенного типа. Если DNS-сервер не кэшировал ответ и *не* является авторитативным для зоны, о которой идет речь, то он начинает прочесывать пространство имен в поиске NS-записей. Всегда существуют NS-записи, расположенные выше в пространстве имен. В качестве последнего средства DNS-сервер использует NS-записи высшего уровня – корневой зоны.

При получении нерекурсивного запроса DNS-сервер отвечает найденными NS-записями. С другой стороны, если исходный запрос был рекурсивным, DNS-сервер посылает запросы удаленным DNS-серверам, в соответствие с найденными NS-записями. Когда DNS-сервер получает ответ от одного из удаленных DNS-серверов, то кэширует этот ответ и, при необходимости, повторяет процесс. Ответ удаленного сервера может содержать ответ на вопрос либо перечень DNS-серверов, расположенных ниже в пространстве имен и ближе к ответу.

Для следующего примера предположим, что мы пытаемся удовлетворить рекурсивный запрос и не нашли никаких NS-записей вплоть до зоны *gov*. И это действительно может иметь место, если мы поинтересуемся у DNS-сервера *relay.hp.com* о данных по имени *www.whitehouse.gov*; он не найдет подходящих NS-записей, пока не доберется до зоны *gov*. Затем мы переключаем DNS-сервер на DNS-сервер зоны *gov* и задаем тот же вопрос. Получаем направление к DNS-серверам *whitehouse.gov*. Переключаем DNS-сервер на DNS-сервер зоны *whitehouse.gov* и задаем тот же вопрос еще раз:

```
% nslookup
Default Server:  relay.hp.com
Address:  15.255.152.2

> set norec            – запрос в стиле DNS-сервера: выключаем рекурсию
> set nosearch         – не использовать список поиска
> www.whitehouse.gov   – последнюю точку можно опустить, поскольку
                         список поиска отключен

Server:  relay.hp.com
Address:  15.255.152.2

Name:    www.whitehouse.gov
Served by:
- I.ROOT-SERVERS.NET
          192.36.148.17
          gov
- E.ROOT-SERVERS.NET
          192.203.230.10
          gov
- D.ROOT-SERVERS.NET
          128.8.10.90
          gov
- B.ROOT-SERVERS.NET
          128.9.0.107
          gov
```

```
- C.ROOT-SERVERS.NET
          192.33.4.12
          gov
- A.ROOT-SERVERS.NET
          198.41.0.4
          gov
- H.ROOT-SERVERS.NET
          128.63.2.53
          gov
- G.ROOT-SERVERS.NET
          192.112.36.4
          gov
- F.ROOT-SERVERS.NET
          192.5.5.241
          gov
```

Переключение на DNS-сервер *gov* (возможно, придется временно включить рекурсию, если ваш DNS-сервер не хранит в кэше адреса DNS-сервера *gov*):

```
> server e.root-servers.net
Default Server:  e.root-servers.net
Address: 192.203.230.10
```

Задаем тот же вопрос DNS-серверу *gov*. Он перенаправит нас к DNS-серверам, расположенным ближе к искомому ответу:

```
> www.whitehouse.gov.
Server:  e.root-servers.net
Address:  192.203.230.10

Name:    www.whitehouse.gov
Served by:
- DNSAUTH1.SYS.GTEI.NET

          whitehouse.gov
- DNSAUTH2.SYS.GTEI.NET

          whitehouse.gov
- DNSAUTH3.SYS.GTEI.NET

          whitehouse.gov
```

Выбираем один из DNS-серверов *whitehouse.gov* – подойдет любой:

```
> server dnsauth2.sys.gtei.net.
Default Server:  dnsauth2.sys.gtei.net
Address:  4.2.49.3

> www.whitehouse.gov.
Server:  sec1.dns.psi.net
Address:  38.8.92.2

Name:    www.whitehouse.gov
Addresses:  198.137.240.91, 198.137.240.92
```

Надеемся, этот пример дал читателям представление о том, как DNS-серверы производят поиск по доменным именам. Если необходимо освежить картинку в памяти, обратитесь к рис. 2.12 и 2.13.

Прежде чем двинуться дальше, хотим обратить внимание читателей на тот факт, что всем DNS-серверам мы задавали один и тот же вопрос: «Какой адрес у *www.whitehouse.gov*?» Как вы думаете, что произошло бы в случае, если бы DNS-сервер зоны *gov* сам уже кэшировал адрес *www.whitehouse.gov*? Этот DNS-сервер ответил бы на наш вопрос данными из кэша, вместо того чтобы направлять нас к DNS-серверам *whitehouse.gov*. Почему это имеет значение? Предположим, мы что-то напутали с адресом одного из узлов нашей зоны. Нам указывают на ошибку, и мы ее исправляем. Наш DNS-сервер хранит корректные данные, но некоторые из удаленных DNS-серверов возвращают неправильные данные, когда им задают вопрос об адресе этого узла. Один из DNS-серверов, обслуживающих зону более высокого уровня, например корневой DNS-сервер, кэшировал неправильные данные; получив запрос адреса узла, он отвечает неправильными данными, вместо того чтобы направить автора запроса к нашему DNS-серверу. Эту проблему довольно трудно выявить, поскольку только один из серверов «более высокого уровня» кэшировал неправильные данные, а значит, лишь некоторые запросы будут приводить к получению неправильных данных, а именно – запросы, адресованные этому DNS-серверу. Весело, правда? Конечно, в конце концов, неправильная запись в кэше DNS-сервера «более высокого уровня» устареет и будет удалена. Если мы не хотим ждать, можно связаться с администраторами удаленного сервера и попросить их перезапустить *named* с целью очистки кэша. Разумеется, если удаленный DNS-сервер является важным и хорошо загруженным, администраторы, весьма вероятно, объяснят, куда мы можем отправляться с такими предложениями.

Передача зоны

nslookup может использоваться для передачи целой зоны с помощью команды *ls*. Эта возможность полезна в плане диагностики, если необходимо узнать доменное имя удаленного узла либо просто сосчитать, сколько узлов содержится во внешней зоне. Поскольку вывод может быть объемным, *nslookup* позволяет перенаправлять его в файл. Чтобы прервать процесс передачи зоны, можно использовать стандартный символ прерывания.

Внимание: некоторые DNS-серверы не позволят вам получить копию зоны либо из соображений безопасности, либо в целях сокращения нагрузки. Интернет – мирная сеть, но администраторы должны защищать свою территорию.

Взглянем на зону *movie.edu*. Как можно видеть из следующего фрагмента, доступны все данные зоны – SOA-запись даже дважды, что является артефактом, появление которого связано с обменом данными в

процессе передачи зоны. Поскольку некоторые версии *nslookup* по умолчанию отображают только адрес и NS-записи, мы использовали ключ *-d* для получения всей зоны:

```
% nslookup
Default Server:  terminator.movie.edu
Address:  0.0.0.0

> ls -d movie.edu.
[terminator.movie.edu]
$ORIGIN movie.edu.
@                       1D IN SOA       terminator al.robocop (
                                        2000091400      ; порядковый номер
                                        3H              ; обновление
                                        1H              ; повторение
                                        4W2D            ; устаревание
                                        1H )            ; минимальное

                        1D IN NS        terminator
                        1D IN NS        wormhole
wormhole                1D IN A         192.249.249.1
                        1D IN A         192.253.253.1
wh249                   1D IN A         192.249.249.1
robocop                 1D IN A         192.249.249.2
bigt                    1D IN CNAME     terminator
cujo                  1D IN TXT       "Location:" "machine" "room" "dog" "house"
wh253                   1D IN A         192.253.253.1
wh                      1D IN CNAME     wormhole
shining                 1D IN A         192.253.253.3
terminator              1D IN A         192.249.249.3
localhost               1D IN A         127.0.0.1
fx                      1D IN NS        bladerunner.fx
bladerunner.fx          1D IN A         192.253.254.2
fx                      1D IN NS        outland.fx
outland.fx              1D IN A         192.253.254.3
fx                      1D IN NS        huskymo.boulder.acmebw.com.
                        1D IN NS        tornado.acmebw.com.
dh                      1D IN CNAME     diehard
carrie                  1D IN A         192.253.253.4
diehard                 1D IN A         192.249.249.4
misery                  1D IN A         192.253.253.2
@                       1D IN SOA       terminator al.robocop (
                                        2000091400      ; порядковый номер
                                        3H              ; обновление
                                        1H              ; повторение
                                        4W2D            ; устаревание
                                        1H )            ; минимальное
```

Предположим, мы не успели прочитать запись в начале данных зоны, поскольку она исчезла за верхней границей экрана. *nslookup* позволяет сохранить записи зоны в файл:

```
> ls -d movie.edu  > /tmp/movie.edu  — перечислить все данные
                                       в файле /tmp/movie.edu
 [terminator.movie.edu]
Received 25 answers (25 records).
```

Некоторые из версий *nslookup* даже реализуют встроенную команду *view*, которая сортирует и отображает содержимое зоны в диалоговом режиме. Однако в последних изданиях BIND 8 команда *view* не работает, а в BIND 9 – на момент существования версии 9.1.0 – не поддерживается.

Разрешение проблем с nslookup

Последнее, чего можно пожелать, так это проблем с инструментом, предназначенным для диагностирования проблем. К сожалению, некоторые ошибки делают *nslookup* практически бесполезной программой. Прочие случаи ошибок *nslookup* (в лучшем случае) могут озадачить, поскольку при их возникновении пользователю не предоставляется информация, на основе которой он мог бы продолжить работу. Собственно *nslookup* является причиной проблем далеко не во всех случаях, в основном проблемы можно отнести на счет настройки и работы DNS-серверов. Проблемы такого рода мы и рассмотрим далее.

Поиск правильных данных

По сути, это не проблема, но может серьезно озадачить. Если использовать *nslookup* для поиска записи определенного типа для доменного имени в случае, когда доменное имя существует, а записи искомого типа – нет, обнаруживается следующая ошибка:

```
% nslookup
Default Server:  terminator.movie.edu
Address:  0.0.0.0

> movie.edu.

*** No address (A) records available for movie.edu.
```

Тогда какого типа записи существуют? Чтобы узнать это, выполним команду *set type=any*:

```
> set type=any
> movie.edu.
Server:  terminator.movie.edu
Address:  0.0.0.0

movie.edu
      origin = terminator.movie.edu
      mail addr = al.robocop.movie.edu
      serial = 42
      refresh = 10800 (3H)
      retry   = 3600 (1H)
```

```
        expire  = 604800 (7D)
        minimum ttl = 86400 (1D)
movie.edu    nameserver = terminator.movie.edu
movie.edu    nameserver = wormhole.movie.edu
movie.edu    nameserver = zardoz.movie.edu
movie.edu    preference = 10, mail exchanger = postmanrings2x.movie.edu
postmanrings2x.movie.edu        internet address = 192.249.249.66
```

Сервер не отвечает

Что случилось, если наш сервер имен не может найти собственное имя?

```
% nslookup
Default Server:  terminator.movie.edu
Address:  0.0.0.0

> terminator
Server:  terminator.movie.edu
Address:  0.0.0.0

*** terminator.movie.edu can't find terminator: No response from server
```

Сообщение об ошибке «no response from server» следует понимать буквально: клиент не получил ответа. Совершенно не факт, что *nslookup* производит поиск чего-либо при запуске. Если вы видите, что адрес вашего DNS-сервера – 0.0.0.0, это означает, что *nslookup* воспользовался именем узла системы (которое возвращается командой *hostname*) в качестве значения поля *Default Server* (сервер по умолчанию), а затем выдал приглашение. И только при попытке найти какие-либо данные выясняется, что сервер не отвечает. В этом случае довольно очевидно, что DNS-сервер просто не запущен – потому что DNS-сервер должен смочь найти информацию по собственному имени. Однако если вы ищете информацию из внешнего мира, DNS-сервер мог не ответить, поскольку просто не успел найти данные до того, как *nslookup* потерял надежду получить ответ. Как определить разницу между DNS-сервером, который просто не был запущен, и DNS-сервером, который работает, но не ответил на запрос? Можно воспользоваться командой *ls*:

```
% nslookup
Default Server:  terminator.movie.edu
Address:  0.0.0.0

> ls foo.      – пытаемся получить список записей несуществующей зоны
*** Can't list domain foo.: No response from server
```

В данном случае DNS-сервер не запущен.[1] Если до узла невозможно достучаться, сообщение об ошибке будет «timed out» (истек интервал

[1] Вообще-то такая ошибка может означать также и то, что где-то в сети фильтруются TCP-соединения на данный сервер – причем как в одном (no response) так и в другом (timeout) случае. – *Примеч. науч. ред.*

ожидания). В случае, когда DNS-сервер запущен, будет получено следующее сообщение об ошибке:

```
% nslookup
Default Server:  terminator.movie.edu
Address:  0.0.0.0

> ls foo.
 [terminator.movie.edu]
*** Can't list domain foo.: No information
```

Разумеется, если в вашем мире действительно не существует зоны высшего уровня *foo*.

Отсутствует PTR-запись для адреса DNS-сервера

Вот одна из самых неприятных ошибок программы *nslookup*: что-то пошло не так, и *nslookup* завершил работу сразу после запуска:

```
% nslookup

*** Can't find server name for address 192.249.249.3: Non-existent host/domain
*** Default servers are not available
```

Сообщение «nonexistent domain» (несуществующий домен) означает, что имя *3.249.249.192.in-addr.arpa* не существует. Иначе говоря, программе *nslookup* не удалось отобразить 192.249.249.3, адрес своего DNS-сервера, в доменное имя. Но разве мы не сказали только что, что *nslookup* ничего не ищет при запуске? В приведенной ранее конфигурации программа *nslookup* действительно ничего не искала при запуске. Но это не правило. Если создать файл *resolv.conf*, содержащий одну или более инструкций *nameserver*, *nslookup* попытается произвести обратное отображение адреса, чтобы получить доменное имя DNS-сервера. В предыдущем примере *присутствует* DNS-сервер по адресу 192.249.249.3, но упоминается, что не существует PTR-записей для адреса 192.249.249.3. Очевидно, проблемы связаны с ошибками зоны обратного отображения, по крайней мере там, где дело касается доменного имени *3.49.249.192.in-addr.arpa*.

Сообщение «default servers are not available» (DNS-серверы по умолчанию недоступны) вводит пользователя в заблуждение. В конце концов, присутствует DNS-сервер, который может сообщить, что доменное имя не существует. Чаще, если сервер не запущен, либо до него невозможно достучаться, встречается сообщение об ошибке «no response from server» (сервер не ответил). Только в этом случае сообщение «default servers are not available» приобретает смысл.

Запрос отвергнут

Отказ в выполнении запросов может быть источником проблем при запуске, а также причиной нерезультативного поиска в сеансе работы.

Вот так выглядит завершение *nslookup* после запуска по причине получения отказа:

```
% nslookup
*** Can't find server name for address 192.249.249.3: Query refused
*** Default servers are not available
%
```

Существуют две возможные причины такого поведения. DNS-сервер не поддерживает инверсные запросы (более старые версии *nslookup*) либо существующий список доступа не позволяет произвести поиск.

Более старые версии *nslookup* (до версии 4.8.3) выполняли при запуске инверсные запросы. Инверсные запросы никогда не применялись широко – и программа *nslookup* являлась одним из тех приложений, в которых инверсные запросы использовались. Начиная с BIND версии 4.9 поддержка инверсных запросов прекратилась, что привело к появлению проблем со старыми версиями *nslookup*. Чтобы справиться с этими проблемами были созданы новые параметры настройки.

В BIND 4 инструкция выглядит следующим образом:

```
options fake-iquery
```

Соответствующая инструкция для BIND 8:

```
options { fake-iquery yes; };
```

(BIND 9 не поддерживает *fake-iquery* на момент существования версии 9.1.0.)

Эта инструкция предписывает DNS-серверу отвечать на инверсные запросы «поддельным» ответом, который достаточно корректен, чтобы программа *nslookup* продолжала работу.[1]

Проблемы при запуске также можно отнести на счет списков доступа. Когда *nslookup* пытается найти доменное имя своего DNS-сервера (посылая не инверсный, а PTR-запрос), то может получить отказ. Если вы думаете, что проблема в списке доступа, убедитесь, что дали разрешение узлу, на котором происходит работа, посылать запросы DNS-серверу. Проверьте TXT-записи *secure_zone* и предписания *allow-query*, связанные с IP-адресом локального узла, либо адресом loopback-интерфейса, если *nslookup* выполняется на той же машине, что и DNS-сервер.

Списки доступа могут не только предотвратить успешный запуск *nslookup*. Они могут предотвращать успешное завершение поиска или передачи зоны в самый разгар сеанса работы, когда *nslookup* переключается на удаленный DNS-сервер. Это выглядит примерно так:

```
% nslookup
```

[1] Поддельный ответ на инверсный запрос, скажем, доменного имени с адресом 192.249.249.3 – тот же самый адрес в квадратных скобках, [192.249.249.3].

```
Default Server:  hp.com
Address:  15.255.152.4

> server terminator.movie.edu
Default Server:  terminator.movie.edu
Address:  192.249.249.3

> carrie.movie.edu.
Server:  terminator.movie.edu
Address:  192.249.249.3

*** terminator.movie.edu can't find carrie.movie.edu.: Query refused

> ls movie.edu          — попытка передать зону
 [terminator.movie.edu]
*** Can't list domain movie.edu: Query refused
>
```

Первый из DNS-серверов resolv.conf не отвечает

Еще одна вариация последней проблемы:

```
% nslookup
*** Can't find server name for address 192.249.249.3: No response from server
Default Server:  wormhole.movie.edu
Address:  192.249.249.1
```

Первый сервер из списка, приводимого в файле *resolv.conf*, не ответил. В *resolv.conf* присутствовала вторая инструкция *nameserver*, и второй DNS-сервер отозвался на запрос. С этого момента *nslookup* будет посылать запросы только узлу *wormhole.movie.edu*; но не будет посылать последующие запросы серверу по адресу 192.249.249.3.

Как узнать, что происходит

В последних примерах мы размахивали руками, утверждая, что *nslookup* ищет адрес DNS-сервера, но никак не доказали этот факт. Вот наше доказательство. На этот раз, при запуске *nslookup* мы включили отладку *d2* с помощью ключа командной строки. На втором уровне отладки *nslookup* печатает посылаемые сообщения, а также сообщает об истечении интервалов ожидания и переключениях между серверами:

```
% nslookup -d2
------------
SendRequest(  ), len 44
    HEADER:
        opcode = QUERY, id = 1, rcode = NOERROR
        header flags:  query, want recursion
        questions = 1,  answers = 0,  authority records = 0,
        additional = 0

    QUESTIONS:
        3.249.249.192.in-addr.arpa, type = PTR, class = IN
```

```
------------
timeout (5 secs)
timeout (10 secs)
timeout (20 secs)
timeout (40 secs)
SendRequest failed

*** Can't find server name for address 192.249.249.3: No response from server
*** Default servers are not available
```

Как можно видеть (строки timeout), программа *nslookup* потратила 75 секунд на ожидание ответов, перед тем как окончательно сдаться. Без включения отладки экран был бы пуст в течение 75 секунд; это выглядело бы так, словно программа повисла.

Неопределенная ошибка

Существует возможность встретиться с довольно неприятной ошибкой, которая называется «неопределенной». Ниже мы приводим пример такой ошибки. Здесь присутствует лишь окончание фрагмента, поскольку нас на данный момент интересует лишь собственно ошибка (полностью сеанс работы с *nslookup*, приведший к получению этой ошибки, содержится в главе 14 «Разрешение проблем DNS и BIND»):

```
Authoritative answers can be found from:
 (root)  nameserver = NS.NIC.DDN.MIL
 (root)  nameserver = B.ROOT-SERVERS.NET
 (root)  nameserver = E.ROOT-SERVERS.NET
 (root)  nameserver = D.ROOT-SERVERS.NET
 (root)  nameserver = F.ROOT-SERVERS.NET
 (root)  nameserver = C.ROOT-SERVERS.NET
 (root)  nameserver =
*** Error: record size incorrect (1050690 != 65519)

*** relay.hp.com can't find .: Unspecified error
```

Дело в том, что объем ответной информации оказался слишком большим и не поместился в UDP-дейтаграмму. DNS-сервер перестал заполнять ответ, когда кончилось место. Бит усечения в ответном сообщении не был установлен, потому что в этом случае программа *nslookup* повторила бы запрос через TCP-соединение; скорее всего, DNS-сервер решил, что включил достаточное количество «важной» информации. Ошибки такого рода встречаются не слишком часто. Это может происходить при создании слишком большого числа NS-записей для зоны, так что следует умерять свои аппетиты. (Подозреваем, советы вроде этого заставляют читателей спрашивать себя, зачем они купили эту книгу.) Сколько записей «слишком много» – зависит от того, насколько хорошо могут «упаковываться» доменные имена в пакете, а это, в свою очередь, зависит от того, сколько имен DNS-серверов, заканчиваются одинаковым доменным именем. Корневые DNS-серверы были пе-

реименованы и имеют окончание *root-servers.net* именно по этой причине – в результате корневых серверов в сети Интернет может быть больше (13). Основное правило – старайтесь не создавать более десяти NS-записей. Чтобы узнать причину ошибки, показанной выше, придется просто прочесть главу 14. Те из читателей, кто только что прочитал главу 9 «Материнство», уже могут догадаться самостоятельно.

Лучшие в сети

Труд системного администратора неблагодарный. Существуют определенные вопросы, обычно довольно простые, которые задаются пользователями снова и снова. И иногда, пребывая в творческом настроении, администраторы изобретают эффективные способы помочь своим пользователям. Когда мы обнаруживаем, насколько их решения гениальны, то можем только восхищаться ими, жалея, что сами об этом не подумали. Вот один из таких случаев, системный администратор нашел способ решить досадную проблему с завершением сеанса *nslookup*:

```
% nslookup
Default Server:  envy.ugcs.caltech.edu
Address:  131.215.134.135

> quit
Server:  envy.ugcs.caltech.edu
Addresses:  131.215.134.135, 131.215.128.135

Name:    ugcs.caltech.edu
Addresses:  131.215.128.135, 131.215.134.135
Aliases:  quit.ugcs.caltech.edu
          use.exit.to.leave.nslookup.-.-.-.ugcs.caltech.edu

> exit
%
```

Работа с dig

Это один из способов справиться с так называемым недостатком *nslookup*. Второй способ – просто выбросить *nslookup* и применять *dig*, Domain Information Groper – искатель доменной информации (название появилось раньше, чем расшифровка сокращения).

Ранее мы говорили, что *dig* не столь распространен, как *nslookup*, поэтому начнем с рассказа о том, где его достать. Исходные тексты *dig* находятся в каталоге *tools* (BIND 4), в каталоге *src/bin/dig* (BIND 8) или в каталоге *bin/dig* (BIND 9) дистрибутива BIND. Если пакет собирается из исходных текстов, будет собрана также и новая копию программы *dig*.

При использовании *dig* все аспекты поведения и создания запросов определяются в командной строке, поскольку диалоговый режим работы

в *dig* не реализован. Доменное имя, для которого производится поиск, указывается в качестве первого аргумента, тип запроса (например, *a* при поиске адресных записей, *mx* при поиске MX-записей) – в качестве второго аргумента; по умолчанию происходит поиск адресных записей. DNS-сервер, которому следует посылать запросы, указывается после символа «@», причем можно использовать доменное имя или IP-адрес. По умолчанию запросы посылаются DNS-серверам из *resolv.conf*.

dig очень хорошо разбирается в аргументах. Их можно указывать в любом порядке, а *dig* самостоятельно поймет, что *mx* – это, скорее всего, тип записей, а не доменное имя, для которого ведется поиск.[1]

Одно из серьезных различий между *nslookup* и *dig* заключается в том, что *dig* не использует список поиска, а значит – в качестве аргументов доменных имен всегда должны набираться абсолютные доменные имена. Так:

```
% dig plan9.fx.movie.edu
```

производит поиск адресных записей для *plan9.fx.movie.edu*; запросы отправляются первому DNS-серверу из перечисленных в *resolv.conf*. При этом команда:

```
% dig acmebw.com mx
```

производит поиск MX-записей для *acmebw.com* через тот же DNS-сервер, а команда:

```
% dig @wormhole.movie.edu. movie.edu. soa
```

посылает DNS-серверу *wormhole.movie.edu* запрос SOA-записи для *movie.edu*.

Формат вывода dig

dig отображает полные ответные сообщения DNS, включая специально выделяемые разделы (заголовка, вопроса, ответа, авторитативности и вторичный), а также представляет RR-записи в формате мастер-файла. Это удобно, если существует необходимость воспользоваться выводом инструмента диагностики для создания файла данных зоны, либо файла корневых указателей. К примеру, вывод, полученный при выполнении команды:

```
% dig @a.root-servers.net ns .
```

выглядит следующим образом:

---

[1] В действительности ранние версии пакета BIND 9 (до версии 9.1.0) содержат ущербную в этом смысле реализацию *dig*, которая требует, чтобы аргумент доменного имени предшествовал аргументу типа. При этом DNS-сервер, с которым ведется работа, может быть указан в любой позиции.

```
; <<>> DiG 8.3 <<>> @a.root-servers.net . ns
; (1 server found)
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6
;; flags: qr aa rd; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 13
;; QUERY SECTION:
;;      ., type = NS, class = IN

;; ANSWER SECTION:
.                       6D IN NS        A.ROOT-SERVERS.NET.
.                       6D IN NS        H.ROOT-SERVERS.NET.
.                       6D IN NS        C.ROOT-SERVERS.NET.
.                       6D IN NS        G.ROOT-SERVERS.NET.
.                       6D IN NS        F.ROOT-SERVERS.NET.
.                       6D IN NS        B.ROOT-SERVERS.NET.
.                       6D IN NS        J.ROOT-SERVERS.NET.
.                       6D IN NS        K.ROOT-SERVERS.NET.
.                       6D IN NS        L.ROOT-SERVERS.NET.
.                       6D IN NS        M.ROOT-SERVERS.NET.
.                       6D IN NS        I.ROOT-SERVERS.NET.
.                       6D IN NS        E.ROOT-SERVERS.NET.
.                       6D IN NS        D.ROOT-SERVERS.NET.

;; ADDITIONAL SECTION:
A.ROOT-SERVERS.NET.     6D IN A         198.41.0.4
H.ROOT-SERVERS.NET.     6D IN A         128.63.2.53
C.ROOT-SERVERS.NET.     6D IN A         192.33.4.12
G.ROOT-SERVERS.NET.     6D IN A         192.112.36.4
F.ROOT-SERVERS.NET.     6D IN A         192.5.5.241
B.ROOT-SERVERS.NET.     6D IN A         128.9.0.107
J.ROOT-SERVERS.NET.     5w6d16h IN A    198.41.0.10
K.ROOT-SERVERS.NET.     5w6d16h IN A    193.0.14.129
L.ROOT-SERVERS.NET.     5w6d16h IN A    198.32.64.12
M.ROOT-SERVERS.NET.     5w6d16h IN A    202.12.27.33
I.ROOT-SERVERS.NET.     6D IN A         192.36.148.17
E.ROOT-SERVERS.NET.     6D IN A         192.203.230.10
D.ROOT-SERVERS.NET.     6D IN A         128.8.10.90

;; Total query time: 116 msec
;; FROM: terminator.movie.edu to SERVER: a.root-servers.net  198.41.0.4
;; WHEN: Fri Sep 15 09:47:26 2000
;; MSG SIZE  sent: 17  rcvd: 436
```

Рассмотрим этот фрагмент по разделам.

Первая строка начинается с комментария <<>> *DiG 8.3* <<>> и повторяет аргументы командной строки, то есть отражает факт, что были запрошены NS-записи для корневой зоны у DNS-сервера *a.root-servers.net*.

Следующая строка, (*1 server found*), отражает тот факт, что *dig*, при поиске адресов, связанных с доменным именем, указанным после сим-

вола @, то есть *a.root-servers.net*, нашел ровно один адрес. (Если *dig* находит более трех адресов, то сообщает о трех найденных адресах, поскольку это максимальное число, с которым работает большинство клиентов и DNS-серверов.)

Строка, которая начинается с ->> *HEADER* <<-, является первой частью заголовка ответного сообщения, полученного от удаленного DNS-сервера. Код операции в заголовке всегда QUERY, как и в случае с *nslookup*. Состояние представлено значением NOERROR; но может быть представлено любым из значений, упомянутых в разделе «Отображение сообщений-запросов и сообщений-ответов», выше по тексту. ID – это идентификатор сообщения, 16-битное число, которое используется для связи ответных сообщений с отправленными запросами.

Флаги (flags) содержат дополнительные сведения об ответе. *qr* отражает тот факт, что сообщение является ответом, а не запросом. *dig* декодирует ответы, но не запросы, поэтому *qr* присутствует всегда. Однако это не справедливо для флагов *aa* и *rd*. *aa* отражает авторитативность ответа, а *rd* – запрос рекурсии, бит, который был установлен в запросе (DNS-сервер просто копирует этот бит в ответное сообщение). В большинстве случаев, когда бит *rd* установлен в запросе, ответ будет содержать флаг *ra*, который сообщает нам, что рекурсия на удаленном DNS-сервере была разрешена. Однако сервер *a.root-servers.net* является корневым DNS-сервером, рекурсия на нем запрещена, как мы показывали в главе 11 «Безопасность», поэтому рекурсивные запросы обрабатываются точно так же, как итеративные. Так что сервер игнорирует бит *rd* и показывает, что рекурсия недоступна, сбрасывая флаг *ra*.

Последнее поле заголовка отражает тот факт, что *dig* задал один вопрос и получил 13 записей в разделе ответа, нуль записей в разделе авторитативности и 13 записей в дополнительном разделе.

Следующая за *QUERY SECTION:* строка содержит отправленный запрос: были запрошены NS-записи класса IN для корневой зоны. За строкой *ANSWER SECTION:* следуют 13 NS-записей для корневых DNS-серверов, а за строкой *ADDITIONAL SECTION:* – 13 A-записей, которые соответствуют тем самым 13 корневым DNS-серверам. Если бы ответное сообщение содержало раздел авторитативности, мы бы увидели его содержимое после строки *AUTHORITY SECTION:*.

В самый конец *dig* добавляет сводную информацию о запросе и ответном сообщении. Первая строка содержит данные о том, через сколько времени после отправки запроса удаленный DNS-сервер вернул ответ. Вторая строка показывает, с какого узла был отправлен запрос и какому DNS-серверу. Третья строка содержит временную отметку получения ответного сообщения. Четвертая – размеры запроса и ответа в байтах.

dig: передача зоны

Как и *nslookup*, *dig* может использоваться для передачи зональных данных. Но в отличие от *nslookup dig* не имеет специальной команды, позволяющей запросить передачу зоны. Чтобы инициировать передачу зоны следует указать *axfr* (тип запроса) и доменное имя зоны в качестве аргументов командной строки. Следует помнить, что получить зону можно только от DNS-сервера, который является для зоны авторитативным.

Для получения зоны *movie.edu* от DNS-сервера *wormhole.movie.edu* следует выполнить команду:

```
$ dig @wormhole.movie.edu movie.edu axfr

; <<>> DiG 8.3 <<>> @wormhole.movie.edu movie.edu axfr
; (1 server found)
$ORIGIN movie.edu.
@                    1D IN SOA       terminator al.robocop (
                                     2000091402      ; порядковый номер
                                     3H              ; обновление
                                     1H              ; повторение
                                     1W              ; устаревание
                                     1H )            ; минимальное

                     1D IN NS        terminator
                     1D IN NS        wormhole
                     1D IN NS        outland.fx
outland.fx           1D IN A         192.253.254.3
wormhole             1D IN A         192.249.249.1
                     1D IN A         192.253.253.1
wh249                1D IN A         192.249.249.1
robocop              1D IN A         192.249.249.2
bigt                 1D IN CNAME     terminator
cujo                1D IN TXT       "Location:" "machine" "room" "dog" "house"
wh253                1D IN A         192.253.253.1
wh                   1D IN CNAME     wormhole
shining              1D IN A         192.253.253.3
terminator           1D IN A         192.249.249.3
localhost            1D IN A         127.0.0.1
fx                   1D IN NS        bladerunner.fx
bladerunner.fx       1D IN A         192.253.254.2
fx                   1D IN NS        outland.fx
outland.fx           1D IN A         192.253.254.3
dh                   1D IN CNAME     diehard
carrie               1D IN A         192.253.253.4
diehard              1D IN A         192.249.249.4
misery               1D IN A         192.253.253.2
@                    1D IN SOA       terminator al.robocop (
                                     2000091402      ; порядковый номер
                                     3H              ; обновление
                                     1H              ; повторение
```

```
                                        1W              ; устаревание
                                        1H )            ; минимальное

;; Received 25 answers (25 records).
;; FROM: terminator.movie.edu to SERVER: wormhole.movie.edu
;; WHEN: Fri Sep 22 11:02:45 2000
```

Заметим, что, как и в случае с *nslookup*, SOA-запись присутствует в результате дважды, в начале и в конце данных зоны. Как и все прочие данные, отображаемые в выводе *dig*, результат передачи зоны представляется в формате мастер-файла, поэтому в случае необходимости этот результат можно использовать для создания файлов данных зоны.[1]

Ключи dig

Ключей командной строки *dig* слишком много, чтобы описывать их здесь, поэтому мы ограничимся наиболее важными, а читателей отсылаем к страницам руководства по *dig*.

-x адрес

Программа *nslookup* достаточно сообразительна, чтобы опознавать IP-адрес и находить соответствующее доменное имя в *in-addr.arpa*. *dig* ничем не хуже. Если использовать ключ *-x*, *dig* предполагает, что аргумент доменного имени в действительности является IP-адресом, обращает октеты и добавляет имя *in-addr.arpa*. Использование ключа *-x* также изменяет тип записей, поиск которых ведется по умолчанию, на ANY, так что произвести обратное отображение для IP-адреса можно командой *dig -x 10.0.0.1*.

-p порт

Посылать запросы через указанный порт, а не через стандартный порт 53.

+norec[urse]

Выключить рекурсию (по умолчанию включена).

+vc

Посылать TCP-запросы (по умолчанию посылаются UDP-запросы).

[1] Хотя перед этим придется удалить лишнюю SOA-запись.

- *Уровни отладки*
- *Включение отладки*
- *Чтение отладочной диагностики*
- *Алгоритм работы DNS-клиента и отрицательное кэширование (BIND 8)*
- *Алгоритм работы DNS-клиента и отрицательное кэширование (BIND 9)*
- *Инструменты*

13

Чтение отладочного вывода BIND

– Ах, Лилия, – сказала Алиса, глядя на Тигровую Лилию, легонько покачивающуюся на ветру.
– Как жалко, что вы не умеете говорить!
– Говорить-то мы умеем, – ответила Лилия.
– Было бы с кем!

Одним из важных инструментов в наборе для решения проблем является отладочный вывод DNS-серверааа. Если DNS-сервер был собран с ключом DEBUG, его работа может наблюдаться на уровне отдельных запросов. Получаемые сообщения часто довольно загадочные; они предназначаются для того, кто будет сверяться при чтении с исходными текстами сервера. В этой главе мы расскажем про некоторые аспекты отладочного вывода. Мы ставим целью рассказать ровно столько, чтобы читатели могли понять, что делает DNS-сервер; у нас и в мыслях нет приводить здесь полную энциклопедию отладочных сообщений.

Читая приводимые здесь объяснения, вспоминайте материал из предыдущих глав. Рассмотрение одной и той же информации в различных контекстах поможет более полно понять принцип работы DNS-сервера.

Уровни отладки

Объем информации, предоставляемой DNS-сервером, зависит от уровня отладки. Чем ниже уровень отладки, тем меньше объем отладочной информации. Более высокие уровни отладки приводят к получению более подробной информации, но при этом быстрее съедают дисковое

пространство. Прочитав приличный объем отладочной информации, читатели получат представление о том, какой уровень отладки необходим для решения той или иной проблемы. Разумеется, если существует возможность воссоздать проблему, можно начать с уровня 1 и увеличивать уровень отладки, пока не будет получено достаточное количество информации. Для самой простой проблемы – невозможности получения информации по имени – первого уровня в большинстве случаев будет вполне достаточно, так что начинать следует с него.

Информация, доступная на различных уровнях

Ниже приводится список уровней отладки для серверов BIND 8 и BIND 9. Отладочная информация обладает свойством кумулятивности: уровень 2 содержит всю отладочную информацию уровня 1. Данные делятся на следующие основные классы: запуск, обновление базы данных, обработка запросов и работа с зонами. Мы не рассматриваем обновление внутренней базы данных сервера, поскольку проблемы обычно не связаны с этой областью. Тем не менее, как станет ясно в главе 14 «Разрешение проблем DNS и BIND», проблему может представлять природа данных, которые DNS-сервер добавляет или удаляет из внутренней базы данных.

В BIND 8 и 9 существует 99 уровней отладки, но большая часть поступающих в log-файл отладочных сообщений содержится в пределах нескольких уровней, которые мы сейчас и рассмотрим.

Уровни отладки BIND 8

Уровень 1

Информация на этом уровне отладки обязательно краткая. DNS-серверы могут обрабатывать *огромное* число запросов, которые могут создавать *огромный* объем отладочной информации. Поскольку на данном уровне отладки сообщения минимизируются в объеме, существует возможность собрать данные за длительный период времени. Уровень 1 следует применять для получения основной информации по запуску сервера и наблюдения за транзакциями запросов. Некоторые из ошибок, в частности ошибки синтаксиса и ошибки формата пакетов DNS, записываются в log-файл на этом уровне. На этом же уровне отражаются ссылки (referrals).

Уровень 2

Уровень 2 содержит много ценной информации: приводятся IP-адреса внешних DNS-серверов, которые использовались при поиске, а также их RTT-метрики; некорректные ответы; для каждого ответа приводится тип исходного запроса – SYSTEM (sysquery) или USER. Этот уровень можно применять в погоне за проблемой загрузки зоны вторичным DNS-сервером, поскольку он содержит информацию о зоне: порядковый номер, время обновления, повторения попыт-

ки, устаревания и оставшееся время – эти значения используются вторичным DNS-сервером для проверки соответствия зоне, хранимой основным сервером.

Уровень 3

Отладочная информация 3 уровня намного более многословна, поскольку содержит сообщения, связанные с обновлением базы данных DNS-сервера. При применении третьего и более высоких уровней следует позаботиться о наличии достаточного объема свободного дискового пространства. На третьем уровне видны дублирующиеся запросы, генерируемые системные запросы (sysquery), имена удаленных DNS-серверов, использованных при поиске, и число адресов, найденных для каждого сервера.

Уровень 4

Уровень 4 следует применять при необходимости отследить пакеты запросов и ответов, *получаемые* DNS-сервером. На этом уровне также доступна информация о достоверности кэшированных данных.

Уровень 5

На уровне 5 существует большое число различных сообщений, но они не являются особенно полезными для отладки в целом. Уровень также содержит некоторые сообщения об ошибках, к примеру, об ошибочном завершении вызова *malloc( )* или прекращении попыток DNS-сервера обработать запрос.

Уровень 6

Уровень 6 отображает ответы, посылаемые отправителям запросов.

Уровень 7

Уровень 7 содержит несколько сообщений, связанных с настройкой либо с синтаксическим анализом.

Уровень 8

На этом уровне нет значимой отладочной информации.

Уровень 9

На этом уровне нет значимой отладочной информации.

Уровень 10

Уровень 10 следует применять при необходимости отследить пакеты запросов и ответов, *посылаемые* DNS-сервером. Формат пакетов точно такой же, как и для уровня 4. Этот уровень отладки используется не слишком часто, поскольку ответы DNS-сервера можно увидеть с помощью программ *nslookup* и *dig*.

Уровень 11

На этом и более высоких уровнях отладки присутствует лишь пара отладочных сообщений, и эти сообщения являются частью кода, который редко выполняется.

Уровни отладки BIND 9

Уровень 1

Уровень 1 отражает основные операции DNS-сервера: загрузку зоны, служебные операции (запросы SOA-записей, передачу зоны, очистку кэша), NOTIFY-сообщения, полученные запросы и порожденные высокоуровневые задачи (такие как поиск адресов DNS-сервера).

Уровень 2

Уровень 2 отражает мультакастовые запросы.

Уровень 3

Уровень 3 отображает информацию о создании и выполнении задач и операций низкого уровня. К сожалению, большинство этих задач имеют не очень прозрачные имена (что вам говорит *requestmgr_detach*?), а их аргументы и вовсе совершенно загадочны. Уровень 3 также содержит сообщения, связанные с процессом ведения log-файла зоны; к примеру, сообщение поступает всякий раз, когда DNS-сервер вносит запись изменения зоны в log-файл зоны либо когда log-файл применяется к зоне при запуске. Работа валидатора DNSSEC и проверка TSIG-подписей также отражаются на третьем уровне.

Уровень 4

Уровень 4 регистрирует переход DNS-сервера к использованию AXFR в случае, если журнал полученной зоны недоступен.

Уровень 5

Уровень 5 содержит информацию о видах, использованных при ответе на запрос.

Уровень 6

Уровень 6 содержит целый набор сообщений, связанных с передачей зоны другому DNS-серверу, а также с проверкой запросов на передачу.

Уровень 7

На этом уровне содержится лишь пара дополнительных сообщений (обновление и удаление записей из журнала зоны, счетчик числа байтов, полученных при передаче зоны).

Уровень 8

Многие из сообщений, связанных с динамическим обновлением, отражаются на уровне 8: проверка предварительных требований, запись в журнал зоны и откаты. Помимо этого, уровень содержит несколько сообщений низкого уровня, связанных с передачей зоны, в частности регистрацию RR-записей, которые были отправлены при передаче зоны.

Уровень 10

Уровень 10 содержит несколько сообщений, связанных с активностью таймера зоны.

Уровень 20

Уровень 20 отражает установку таймера обновления зоны.

Уровень 90

На этом уровне представлены операции низкого уровня диспетчера задач BIND 9.

В BIND 8 и 9 существует возможность настроить DNS-сервер таким образом, чтобы отладочные сообщения содержали информацию об уровне отладки. Достаточно установить параметр *print-severity* (см. раздел «Ведение log-файла в BIND 8 и 9» в главе 7 «Работа с BIND»).

Следует помнить, что информация является именно отладочной – она использовалась авторами пакета BIND для отладки кода, и, как следствие, она не настолько прозрачна, как нам, возможно, хотелось бы. Информацию можно изучать, чтобы понять, почему DNS-сервер работает не так, как предполагалось, либо просто узнать, как работает DNS-сервер; но не следует ожидать красиво оформленного и хорошо продуманного вывода.

Включение отладки

Отладку в DNS-сервере можно включить при запуске из командной строки либо посылкой управляющих сообщений. Если для диагностирования проблемы необходимо иметь перед глазами информацию, выдаваемую при запуске сервера, нужно указать ключ командной строки. В случае необходимости включить или отключить отладку при уже работающем сервере надо воспользоваться управляющими сообщениями. DNS-сервер записывает отладочный вывод в файл *named.run*. DNS-серверы BIND 4 создают *named.run* в каталоге */usr/tmp* (либо */var/tmp*), а DNS-серверы BIND 8 и 9 – в рабочем каталоге DNS-сервера.

Ключи командной строки

При разрешении проблем иногда бывает необходимо видеть значение *sortlist*, знать, с каким интерфейсом связан файловый дескриптор, либо выяснить, в какой момент на стадии инициализации DNS-сервер завершил работу (в случае, если сообщение об ошибке, записанное демоном *syslog*, было недостаточно прозрачным). Чтобы получить отладочную информацию такого рода, необходимо запустить сервер в режиме отладки со специальным ключом командной строки; потому что ко времени посылки управляющего сообщения уже будет слишком поздно. Ключ командной строки, включающий отладку, выглядит

так: *-d уровень*. DNS-сервер BIND 4 при использовании этого ключа командной строки не переходит в фоновый режим работы, как обычно; чтобы добиться этого, необходимо добавить символ & в конец командной строки. Следующая командная строка запускает DNS-сервер BIND 4 с уровнем отладки 1:

```
# /etc/named -d 1 &
```

DNS-серверы BIND 8 и 9 переходят в фоновый режим работы даже при использовании *-d*, и для них символ & можно не включать в команду.

Изменение уровня отладки с помощью управляющих сообщений

Если нет необходимости изучать инициализацию DNS-сервера, следует запустить сервер без соответствующего ключа командной строки. Позже отладку можно включить или выключить, используя *ndc* для отправления соответствующих управляющих сообщений процессу DNS-сервера. Следующие две команды устанавливают уровень отладки 3, а затем выключают отладку:

```
# ndc trace 3
# ndc notrace
```

Как можно догадаться, если включить отладку ключом командной строки, то *ndc* все так же может применяться для изменения уровня отладки.

В программе *rndc* из пакета BIND 9.1.0 команды *trace* и *notrace* пока не реализованы (их нет и в демоне *named* версии 9.1.0), но будут присутствовать в будущих версиях. Таким образом, при работе с BIND 9 следует использовать ключ командной строки *-d*.

Чтение отладочной диагностики

Мы рассмотрим пять примеров отладочной диагностики. В первом показан запуск DNS-сервера. Следующие два примера связаны с успешным поиском адресов. Четвертый пример – для вторичного DNS-сервера, синхронизирующего хранимую зону. Последний пример относится не к поведению DNS-сервера, но к поведению DNS-клиента, и речь пойдет об алгоритме поиска. В каждом случае (кроме последнего) DNS-сервер запускался заново, практически полностью очищая кэш.

Читатели, возможно, спросят, почему мы решили отразить во всех примерах нормальное поведение сервера – ведь глава посвящена отладке. Дело в том, что необходимо знать, как выглядит нормальная работа, прежде чем начинать охоту за ошибками. Вторая причина – мы стараемся более глубоко объяснить понятия, использованные в предыдущих главах (повторные запросы, время передачи сигнала и т. д.).

Запуск DNS-сервера (BIND 8, уровень отладки 1)

Мы начнем изучение примеров отладочного вывода с фрагмента инициализации DNS-сервера. Первым DNS-сервером будет BIND 8. Был использован ключ командной строки *-d 1*, и вот вывод, полученный в файле *named.run*:

```
1) Debug level 1
2) Version = named 8.2.3-T7B Mon Aug 21 19:21:21 MDT 2000
3) cricket@abugslife.movie.edu:/usr/local/src/bind-8.2.3-T7B/src/bin/named
4) conffile = ./named.conf
5) starting.  named 8.2.3-T7B Mon Aug 21 19:21:21 MDT 2000
6) cricket@abugslife.movie.edu:/usr/local/src/bind-8.2.3-T7B/src/bin/named
7) ns_init(./named.conf)
8) Adding 64 template zones
9) update_zone_info('0.0.127.in-addr.arpa', 1)
10) source = db.127.0.0
11) purge_zone(0.0.127.in-addr.arpa,1)
12) reloading zone
13) db_load(db.127.0.0, 0.0.127.in-addr.arpa, 1, Nil, Normal)
14) purge_zone(0.0.127.in-addr.arpa,1)
15) master zone "0.0.127.in-addr.arpa" (IN) loaded (serial 2000091500)
16) zone[1] type 1: '0.0.127.in-addr.arpa' z_time 0, z_refresh 0
17) update_zone_info('.', 3)
18) source = db.cache
19) reloading hint zone
20) db_load(db.cache, , 2, Nil, Normal)
21) purge_zone(,1)
22) hint zone "" (IN) loaded (serial 0)
23) zone[2] type 3: '.' z_time 0, z_refresh 0
24) update_pid_file(  )
25) getnetconf(generation 969052965)
26) getnetconf: considering lo [127.0.0.1]
27) ifp->addr [127.0.0.1].53 d_dfd 20
28) evSelectFD(ctx 0x80d8148, fd 20, mask 0x1, func 0x805e710, uap 0x40114344)
29) evSelectFD(ctx 0x80d8148, fd 21, mask 0x1, func 0x8089540, uap 0x4011b0e8)
30) listening on [127.0.0.1].53 (lo)
31) getnetconf: considering eth0 [192.249.249.3]
32) ifp->addr [192.249.249.3].53 d_dfd 22
33) evSelectFD(ctx 0x80d8148, fd 22, mask 0x1, func 0x805e710, uap 0x401143b0)
34) evSelectFD(ctx 0x80d8148, fd 23, mask 0x1, func 0x8089540, uap 0x4011b104)
35) listening on [206.168.194.122].53 (eth0)
36) fwd ds 5 addr [0.0.0.0].1085
37) Forwarding source address is [0.0.0.0].1085
38) evSelectFD(ctx 0x80d8148, fd 5, mask 0x1, func 0x805e710, uap 0)
39) evSetTimer(ctx 0x80d8148, func 0x807cbe8, uap 0x40116158, due
969052990.812648000, inter 0.000000000)
40) exit ns_init(  )
41) update_pid_file(  )
42) Ready to answer queries.
43) prime_cache: priming = 0, root = 0
```

```
44) evSetTimer(ctx 0x80d8148, func 0x805bc30, uap 0, due 969052969.000000000,
inter 0.000000000)
45) sysquery: send -> [192.33.4.12].53 dfd=5 nsid=32211 id=0 retry=969052969
46) datagram from [192.33.4.12].53, fd 5, len 436
47) 13 root servers
```

Мы добавили к отладочному выводу номера строк, естественно, в обычной ситуации вы их не увидите. Строки со второй по шестую отражают используемую версию BIND и имя файла настройки. Версия 8.2.3-T 7B была выпущена консорциумом ISC (Internet Software Consortium) в августе 2000 года. Для данного случая мы использовали файл настройки, расположенный в текущем каталоге *./named.conf*.

Строки с 7 по 23 отражают чтение файла настройки и файлов данных зоны сервером BIND. Данный DNS-сервер является только кэширующим, соответственно, читаются файлы *db.127.0.0* (строки с 9 по 16) и *db.cache* (строки 17–23). Строка 9 информирует нас об обновлении зоны (*0.0.127.IN-ADDR.ARPA*), а строка 10 – о файлах, содержащих данные для зоны (*db.127.0.0*). Строка 11 говорит о том, что любые старые данные удаляются перед загрузкой новых. Строка 12 уведомляет о перезагрузке зоны, которая производится, несмотря на то обстоятельство, что зона в действительности загружается впервые. Загрузка данных отражена в строках с 13 по 15. В строках 16 и 23 *z_time* – это время проверки актуальности данных зоны, *z_refresh* – время обновления зоны. Эти значения имеют смысл только в случае, когда DNS-сервер является вторичным для зоны.

Строки с 25 по 39 отражают процесс инициализации файловых дескрипторов. (В данном случае речь идет на самом деле о дескрипторах сокетов.) Файловые дескрипторы 20 и 21 (строки 27–29) связываются с адресом loopback-интерфейса, 127.0.0.1. Дескриптор 20 – сокет дейтаграмм, а дескриптор 21 – потоковый. Файловые дескрипторы 22 и 23 (строки 32–34) связываются с интерфейсом 192.249.249.3. Адрес каждого интерфейса был исследован и использован; адрес не используется, если он уже присутствует в списке либо если интерфейс не был инициализирован. Файловый дескриптор 5 (строки 36–39) связывается с адресом по маске, 0.0.0.0. Большинство сетевых демонов используют только один сокет – связанный с этим адресом, а не сокеты, связанные с конкретными интерфейсами. Адрес по маске принимает пакеты, посылаемые любому из интерфейсов узла. Мы сделаем небольшое отступление и объясним, по какой причине *named* использует одновременно сокет, связанный с адресом маски, и сокеты, связанные с конкретными интерфейсами.

Когда *named* получает запрос от приложения, либо от другого DNS-сервера, запрос поступает через один из сокетов, связанных с конкретными интерфейсами. Если бы *named* не имел сокетов, связанных с конкретными интерфейсами, то получал бы запросы через сокет, связанный с адресом маски. Когда *named* посылает ответ, то использует

тот же дескриптор сокета, через который был получен запрос. Почему *named* поступает именно так? Когда ответы посылаются через сокет, связанный с адресом маски, ядро подставляет в качестве адреса отправителя адрес интерфейса, через который в действительности отправляется ответ. Этот адрес может совпадать или не совпадать с адресом, которому был направлен исходный запрос. Когда ответы посылаются через сокет, связанный с конкретным адресом, ядро подставляет в качестве адреса отправителя этот самый конкретный адрес – тот же адрес, которому был направлен исходный запрос. Если DNS-сервер получает ответ с IP-адреса, о котором ему ничего не известно, этот ответ получает статус «пришельца» и игнорируется. *named* старается избежать создания ответов-пришельцев, отправляя ответы через дескрипторы, связанные с конкретными интерфейсами, чтобы адрес отправителя совпадал с адресом получателя исходного запроса. Однако при посылке *запросов named* использует дескриптор маски, поскольку нет необходимости использовать конкретный IP-адрес.

Строки с 43 по 47 отражают посылку DNS-сервером системного запроса с целью выяснения, какие DNS-серверы в настоящий момент обслуживают корневую зону. Это действие известно как «первичное заполнение кэша». Первый же DNS-сервер возвращает ответ, содержащий информацию о 13 DNS-серверах.

Итак, DNS-сервер инициализирован и готов выполнять запросы.

Запуск DNS-сервера (BIND 9, уровень отладки 1)

Отладочный вывод, полученный при запуске DNS-сервера BIND 9:

```
1)  Sep 15 15:34:53.878 starting BIND 9.1.0 -d1
2)  Sep 15 15:34:53.883 using 1 CPU
3)  Sep 15 15:34:53.899 loading configuration from './named.conf'
4)  Sep 15 15:34:53.920 the default for the 'auth-nxdomain' option is now 'no'
5)  Sep 15 15:34:54.141 no IPv6 interfaces found
6)  Sep 15 15:34:54.143 listening on IPv4 interface lo, 127.0.0.1#53
7)  Sep 15 15:34:54.151 listening on IPv4 interface eth0, 192.249.249.3#53
8)  Sep 15 15:34:54.163 command channel listening on 0.0.0.0#953
9)  Sep 15 15:34:54.180 now using logging configuration from config file
10) Sep 15 15:34:54.181 dns_zone_load: zone 0.0.127.in-addr.arpa/IN: start
11) Sep 15 15:34:54.188 dns_zone_load: zone 0.0.127.in-addr.arpa/IN: loaded
12) Sep 15 15:34:54.189 dns_zone_load: zone 0.0.127.in-addr.arpa/IN:
dns_journal _rollforward: no journal
13) Sep 15 15:34:54.190 dns_zone_maintenance: zone 0.0.127.in-addr.arpa/IN:
enter
14) Sep 15 15:34:54.190 dns_zone_maintenance: zone version.bind/CHAOS: enter
15) Sep 15 15:34:54.190 running
```

Читатели, вероятно, уже отметили основное различие между фрагментами отладочного вывода BIND версий 8 и 9. Диагностика BIND 9 весьма лаконична. Дело в том, что BIND 8 существует уже около трех

лет, так что у авторов была масса времени, чтобы добавить к коду отладочные сообщения. BIND 9 только что сошел с конвейера, так что его набор сообщений пока еще не очень велик.

Помимо этого, в каждом отладочном сообщении BIND 9 присутствует отметка времени, что очень удобно, если существует необходимость сопоставить отладочные сообщения с некоторыми внешними событиями.

Строки 1 и 2 содержат информацию об используемой версии BIND (9.1.0) и имя файла настройки. Как и в предыдущем примере, это файл *named.conf*, расположенный в текущем каталоге. В строке 3 содержится информация о том, что используется один процессор, чего и следовало ожидать на машине с одним процессором.

Строка четыре содержит простое предупреждение об изменении стандартного значения предписания *auth-nxdomain* (которое рассматривается в главе 10 «Дополнительные возможности»). Строка 5 напоминает, что на нашем узле отсутствуют сетевые интерфейсы IP версии 6; в противном случае BIND 9 мог бы принимать запросы через эти интерфейсы.

Строки 6 и 7 отражают тот факт, что сервер ожидает поступления запросов через два сетевых интерфейса: *lo*, loopback-интерфейс, и *eth0*, Ethernet-интерфейс. BIND 9 отображает адрес и порт в формате *адрес#порт*, тогда как в BIND 8 используется формат *[адрес].порт*. Строка 8 сообщает, что *named* принимает управляющие сообщения, поступающие через стандартный порт 953.

Строки 10–12 отражают загрузку сервером зоны *0.0.127.in-addr.arpa*. Смысл сообщений *start* (запуск) и *loaded* (загрузка завершена) очевиден. Сообщение *no journal* отражает отсутствие журнала. (Журнал, описанный в главе 10, представляет собой запись динамических обновлений, полученных сервером для зоны.)

Наконец, строки 13 и 14 отражают выполнение сервером служебных операций для зон *0.0.127.in-addr.arpa* и *version.bind*. (*version.bind* – это встроенная CHAOSNET-зона, которая содержит единственную TXT-запись, связанную с доменным именем *version.bind*.) Выполнение служебных операций для зоны – это процесс, который планирует выполнение регулярных задач, таких как запросы SOA-записей для вторичного сервера и зон-заглушек, или отправка NOTIFY-сообщений.

Успешный поиск (BIND 8, уровень отладки 1)

Предположим, существует необходимость отследить процесс поиска сервером данных для доменного имени. Сервер был запущен без применения соответствующего ключа командной строки. Вызовем *ndc*, чтобы включить отладку, выполним операцию для конкретного имени, и затем выключим отладку, следующим образом:

```
# ndc trace 1
```

```
# /etc/ping galt.cs.purdue.edu.
# ndc notrace
```

Вот полученный в результате файл *named.run*:

```
datagram from [192.249.249.3].1162, fd 20, len 36

req: nlookup(galt.cs.purdue.edu) id 29574 type=1 class=1
req: missed 'galt.cs.purdue.edu' as '' (cname=0)
forw: forw -> [198.41.0.10].53 ds=4 nsid=40070 id=29574 2ms retry 4sec
datagram from [198.41.0.10].53, fd 4, len 343

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 40070
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 9, ADDITIONAL: 9
;;              galt.cs.purdue.edu, type = A, class = IN
EDU.                        6D IN NS    A.ROOT-SERVERS.NET.
EDU.                        6D IN NS    H.ROOT-SERVERS.NET.
EDU.                        6D IN NS    B.ROOT-SERVERS.NET.
EDU.                        6D IN NS    C.ROOT-SERVERS.NET.
EDU.                        6D IN NS    D.ROOT-SERVERS.NET.
EDU.                        6D IN NS    E.ROOT-SERVERS.NET.
EDU.                        6D IN NS    I.ROOT-SERVERS.NET.
EDU.                        6D IN NS    F.ROOT-SERVERS.NET.
EDU.                        6D IN NS    G.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.             5w6d16h IN A    198.41.0.4
H.ROOT-SERVERS.NET.             5w6d16h IN A    128.63.2.53
B.ROOT-SERVERS.NET.             5w6d16h IN A    128.9.0.107
C.ROOT-SERVERS.NET.             5w6d16h IN A    192.33.4.12
D.ROOT-SERVERS.NET.             5w6d16h IN A    128.8.10.90
E.ROOT-SERVERS.NET.             5w6d16h IN A    192.203.230.10
I.ROOT-SERVERS.NET.             5w6d16h IN A    192.36.148.17
F.ROOT-SERVERS.NET.             5w6d16h IN A    192.5.5.241
G.ROOT-SERVERS.NET.             5w6d16h IN A    192.112.36.4
resp: nlookup(galt.cs.purdue.edu) qtype=1
resp: found 'galt.cs.purdue.edu' as 'edu' (cname=0)
resp: forw -> [192.36.148.17].53 ds=4 nsid=40071 id=29574 1ms
datagram from [192.36.148.17].53, fd 4, len 202

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 40071
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 4, ADDITIONAL: 4
;;   galt.cs.purdue.edu, type = A, class = IN
PURDUE.EDU.                 2D IN NS    NS.PURDUE.EDU.
PURDUE.EDU.                 2D IN NS    MOE.RICE.EDU.
PURDUE.EDU.                 2D IN NS    PENDRAGON.CS.PURDUE.EDU.
PURDUE.EDU.                 2D IN NS    HARBOR.ECN.PURDUE.EDU.
NS.PURDUE.EDU.                  2D IN A     128.210.11.5
MOE.RICE.EDU.                   2D IN A     128.42.5.4
PENDRAGON.CS.PURDUE.EDU.              2D IN A  128.10.2.5
HARBOR.ECN.PURDUE.EDU.              2D IN A     128.46.199.76
resp: nlookup(galt.cs.purdue.edu) qtype=1
resp: found 'galt.cs.purdue.edu' as 'cs.purdue.edu' (cname=0)
resp: forw -> [128.46.199.76].53 ds=4 nsid=40072 id=29574 8ms
datagram from [128.46.199.76].53, fd 4, len 234
```

```
send_msg -> [192.249.249.3].1162 (UDP 20) id=29574
Debug off
```

Во-первых, обратите внимание, что в диагностике присутствуют IP-адреса, а не доменные имена, что довольно странно для DNS-сервера. В действительности в этом нет ничего особенно странного. Если мы пытаемся решить проблему, связанную с поиском адреса по имени, то следует ограничить поиск по именам, потому что дополнительные операции делают диагностику менее удобной для чтения и затрудняют отладку. Ни один из уровней отладки не предусматривает преобразование IP-адресов в доменные имена. Поэтому придется воспользоваться инструментом, который сделает это. Такой инструмент мы представим читателям позже.

Разберем отладочную диагностику построчно. Столь подробное рассмотрение необходимо, если мы хотим понять, что означает каждая из строк. Если включается отладка, то речь, как правило, идет о том, чтобы выяснить, почему не происходит разрешение для некоторых имен; чтобы разобраться в происходящем, необходимо понимать смысл диагностики.

```
datagram from [192.249.249.3].1162, fd 20, len 36
```

Дейтаграмма поступила от узла с IP-адресом 192.249.249.3 (*terminator.movie.edu*). Дейтаграмма может поступить с адреса 127.0.0.1 в случае, если отправитель пользуется тем же узлом, на котором работает DNS-сервер. Приложение, отправившее дейтаграмму, использовало порт 1162. DNS-сервер получил дейтаграмму через файловый дескриптор (fd) 20. Диагностика запуска DNS-сервера, пример которой мы приводили ранее, позволяет определить, с каким интерфейсом связан файловый дескриптор 20. Длина (len) дейтаграммы составила 36 байтов.

```
req: nlookup(galt.cs.purdue.edu) id 29574 type=1 class=1
```

Поскольку следующая строка начинается с подстроки *req*, становится ясно, что дейтаграмма являлась запросом. Имя, о котором шла речь в запросе, – *galt.cs.purdue.edu*. Идентификатор запроса – 29574. Подстрока *type=1* означает, что речь в запросе идет об адресной информации. Подстрока *class=1* – класс IN. Полный перечень типов запросов и классов содержится в заголовочном файле */usr/ include/arpa/nameser.h*.

```
req: missed 'galt.cs.purdue.edu' as '' (cname=0)
```

DNS-сервер произвел поиск указанного имени и не нашел адреса. Затем была сделана безуспешная попытка найти внешний DNS-сервер, которому имело бы смысл послать запрос; и так вплоть до корневой зоны (пустая строка в кавычках). Подстрока *cname=0* означает, что DNS-сервер не нашел CNAME-записей. Если найдена CNAME-запись, производится поиск по каноническому имени вместо исходного, а *cname* получает ненулевое значение.

```
forw: forw -> [198.41.0.10].53 ds=4 nsid=40070 id=29574 2ms retry 4sec
```

Запрос был передан (через порт 53) DNS-серверу на узле 198.41.0.10 (*j.root-servers.net*). Для отправки запроса DNS-сервер воспользовался дескриптором 4 (который связан с адресом маски). DNS-сервер присвоил запросу идентификационный номер 40070 (*nsid=40070*), чтобы впоследствии идентифицировать ответ на этот запрос. Приложение присвоило запросу идентификационный номер 29574 (*id=29574*), как можно видеть из строки *nlookup*. DNS-сервер ожидает ответа в течение четырех секунд перед отправкой запроса следующему DNS-серверу.

```
datagram from [198.41.0.10].53, fd 4, len 343
```

Ответил DNS-сервер узла *j.root-servers.net*. Поскольку ответ представляет собой делегирование, он полностью отображается в отладочной диагностике.

```
resp: nlookup(galt.cs.purdue.edu) qtype=1
```

После того как информация, полученная в ответе, кэширована, поиск по имени повторяется. Как мы уже сказали, подстрока *qtype=1* означает, что поступил запрос адресной информации.

```
resp: found 'galt.cs.purdue.edu' as 'edu' (cname=0)
resp: forw -> [192.36.148.17].53 ds=4 nsid=40071 id=29574 1ms
datagram from [192.36.148.17].53, fd 4, len 202
```

Корневой сервер ответил перенаправлением к серверам зоны *edu*. Тот же запрос отправляется адресу 192.36.148.17 (*i.root-servers.net*), который соответствует одному из серверов *edu*. *i.root-servers.net* возвращает информацию о серверах зоны *purdue.edu*.

```
resp: found 'galt.cs.purdue.edu' as 'cs.purdue.edu' (cname=0)
```

На этот раз присутствует информация уровня имени *cs.purdue.edu*.

```
resp: forw -> [128.46.199.76].53 ds=4 nsid=40072 id=29574 8ms
```

DNS-серверу по адресу 128.46.199.76 (*harbor.ecn.purdue.edu*) был отправлен запрос. Идентификатор запроса, присвоенный сервером, – 40072.

```
datagram from [128.46.199.76].53, fd 4, len 234
```

DNS-сервер *harbor.ecn.purdue.edu* ответил. Чтобы идентифицировать содержимое ответа, следует посмотреть на последующие события.

```
send_msg -> [192.249.249.3].1162 (UDP 20) id=29574
```

Последний ответ, вероятнее всего, содержал запрошенный адрес, поскольку DNS-сервер ответил приложению (которое использовало при запросе порт 1162, что можно увидеть в строке диагностики исходного запроса). Ответ был отправлен в виде UDP-пакета (а не через TCP-соединение) через файловый дескриптор 20.

DNS-сервер не был нагружен в момент создания вышеприведенного фрагмента; он не обрабатывал параллельно другие запросы. При создании log-файла диагностики на активном DNS-сервере дела обстоят не столь радужно. Придется прочесывать отладочный вывод в поисках частей, относящихся к поиску по имени, о котором идет речь. Но это не особенно трудно. Запустите свой любимый текстовый редактор, найдите строку *nlookup*, в которой идет речь об исходном имени. После этого останется лишь выбрать строки с таким же *nsid*-идентификатором. В следующем фрагменте диагностики для BIND 8 мы покажем, как отследить *nsid*-идентификаторы.

Успешный поиск (BIND 9, уровень отладки 1)

Ниже приводится отладочная диагностика для поиска по тому же доменному имени при использовании DNS-сервера BIND 9 на уровне отладки 1, но она до смешного краткая. Тем не менее, как мы уже говорили, очень важно знать, как выглядит диагностика при правильной работе. Итак:

```
Sep 16 17:20:57.193 client 192.249.249.3#1090: query: galt.cs.purdue.edu A
Sep 16 17:20:57.194 createfetch: galt.cs.purdue.edu. A
```

Первая строка сообщает нам, что клиент с IP-адреса 192.249.249.3 (то есть локальный узел), работающий через порт 1090, отправил нам запрос адреса для имени *galt.cs.purdue.edu*. Авторство второй строки принадлежит той части DNS-сервера, которая занимается разрешением имен; смысл заключается в том, чтобы дать нам понять, что происходит.

Успешный поиск с повторными запросами (BIND 8, уровень отладки 1)

Не всякий поиск проходит так же «гладко», как последний – иногда для выполнения запроса требуются повторные запросы. В случае успешного окончания поиска для пользователя нет никакой разницы, хотя запросы, выполняемые с необходимостью выполнять повторные запросы, выполняются дольше. Ниже приводится фрагмент отладочного вывода, который содержит информацию о повторных запросах. После получения этого фрагмента мы преобразовали IP-адреса в имена. Обратите внимание, насколько проще читать вывод с именами!

```
1)  Debug turned ON, Level 1
2)
3)  datagram from terminator.movie.edu port 3397, fd 20, len 35
4)  req: nlookup(ucunix.san.uc.edu) id 1 type=1 class=1
5)  req: found 'ucunix.san.uc.edu' as 'edu' (cname=0)
6)  forw: forw -> i.root-servers.net port 53  ds=4 nsid=2 id=1 0ms retry 4 sec
7)
8)  datagram from i.root-servers.net port 53, fd 4, len 240
```

```
     <delegation lines removed>
9)  resp: nlookup(ucunix.san.uc.edu) qtype=1
10) resp: found 'ucunix.san.uc.edu' as 'san.uc.edu' (cname=0)
11) resp: forw -> uceng.uc.edu port 53 ds=4 nsid=3 id=1 0ms
12) resend(addr=1 n=0) - > ucbeh.san.uc.edu port 53 ds=4 nsid=3 id=1 0ms
13)
14) datagram from terminator.movie.edu port 3397, fd 20, len 35
15) req: nlookup(ucunix.san.uc.edu) id 1 type=1 class=1
16) req: found 'ucunix.san.uc.edu' as 'san.uc.edu' (cname=0)
17) resend(addr=2 n=0) - > uccba.uc.edu port 53 ds=4 nsid=3 id=1 0ms
18) resend(addr=3 n=0) - > mail.cis.ohio-state.edu port 53 ds=4 nsid=3 id=1 0ms
19)
20) datagram from mail.cis.ohio-state.edu port 53, fd 4, len 51
21) send_msg -> terminator.movie.edu (UDP 20 3397) id=1
```

Этот фрагмент начинается точно так же, как предыдущий (строки с 1 по 11): DNS-сервер получает запрос для имени *ucunix.san.uc.edu*, посылает запрос DNS-серверу *edu* (*i.root-servers.net*), получает ответ, содержащий перечень DNS-серверов для *uc.edu*, и посылает запрос одному из них (*uceng.uc.edu*).

Новыми в этом фрагменте являются строки *resend* (строки 12, 17 и 18). *Forw* в строке 11 считается в качестве *resend(addr=0 n=0)* – мы, компьютерные незнайки, всегда начинаем считать с нуля. Поскольку сервер *uceng.uc.edu* не ответил, DNS-сервер отправил запрос на *ucbeh.san.uc.edu* (строка 12), *uccba.uc.edu* (строка 17) и *mail.cis.ohio-state.edu* (строка 18). Наконец, отозвался внешний DNS-сервер *mail.cis.ohio-state.edu* (строка 20). Обратите внимание, что все повторные запросы можно найти поиском по подстроке *nsid=3*; и это важно помнить, поскольку другие многочисленные запросы могут вклиниваться между повторными.

Также интересно обратить внимание на вторую дейтаграмму, полученную от узла *terminator.movie.edu* (строка 14). Порт, дескриптор файла, длина, идентификатор и тип запроса полностью идентичны тем, что можно видеть в запросе в строке 3. Приложение не получило ответ вовремя, поэтому оно повторило исходный запрос. Поскольку DNS-сервер все еще обрабатывает первый из полученных запросов, второй запрос является дублирующим. DNS-сервер определил дублирующийся запрос и проигнорировал его, хотя это не отображено в выводе. Мы же знаем это наверняка, поскольку отсутствует строка *forw:* после строк *req:*, в отличие от того, что можно видеть на строках с четвертой по шестую.

Попробуйте угадать, как будет выглядеть этот фрагмент, если DNS-сервер будет испытывать сложности с поиском адресов? Многочисленные повторные запросы, свидетельствующие о попытках DNS-сервера найти адрес (эти строки можно найти поиском подстроки *nsid=*). Приложение отправит несколько дополнительных запросов, полагая, что предыдущие не были получены DNS-сервером. В конце концов DNS-сервер прекратит поиск, и скорее всего, это произойдет уже после того, как собственно приложение потеряет надежду получить ответ.

Если речь идет о DNS-сервере BIND 9.1.0, повторно отправляемые запросы не отображаются на уровнях отладки ниже третьего, а на третьем уровне их довольно нелегко различить в лавине прочих отладочных сообщений BIND 9. Ко всему, даже на уровне отладки 3 BIND 9.1.0 не упоминает о том, *каким* именно DNS-серверам отправляются повторные запросы.

Вторичный DNS-сервер производит проверку зоны (BIND 8, уровень отладки 1)

Помимо проблем, связанных с поиском адресов, могут встречаться проблемы загрузки зоны вторичным DNS-сервером. Источник этой проблемы можно зачастую определить просто сравнением порядковых номеров в SOA-записи зоны на двух серверах, основном и дополнительном, посредством использования *nslookup* или *dig*, что мы и продемонстрируем в главе 14. Если рассматриваемая проблема не поддается решению таким методом, можно прибегнуть к поиску причин в отладочной информации. Мы покажем, как должна выглядеть отладочная информация при нормальной работе сервера.

Приводимый ниже фрагмент отладочного вывода был получен на «спокойном» DNS-сервере – не получающем никаких запросов – чтобы было ясно видно, какие строки относятся к выполнению служебных операций для зоны. Вспомним, что дополнительные DNS-серверы BIND 4 и 8 используют порождаемые процессы для передачи зональных данных на локальный диск перед чтением этих данных. Вторичный DNS-сервер записывает отладочную информацию в файл *named.run*, а порожденный процесс записывает свою в файл *xfer.ddt.PID*. Суффикс *PID* – по умолчанию идентификатор порожденного процесса, может быть изменен в целях обеспечения уникальности имени. Будьте бдительны – включение отладки на дополнительном DNS-сервере приведет к появлению многочисленных файлов с именами *xfer.ddt.PID*, даже если речь идет всего лишь о том, чтобы отследить выполнение запроса адреса. Мы проводим эксперимент на уровень отладки 1, со включенным параметром log-файлирования *print-time* (BIND 8). Уровень отладки 3 для случаев, когда передача зоны происходит, как правило, содержит больше информации, чем необходимо. Передача зоны размером в несколько сотен RR-записей может привести к созданию файла *xfer.ddt.PID* размером в несколько мегабайт.

```
21-Feb 00:13:18.026 do_zone_maint for zone movie.edu (class IN)
21-Feb 00:13:18.034 zone_maint('movie.edu')
21-Feb 00:13:18.035 qserial_query(movie.edu)
21-Feb 00:13:18.043 sysquery: send -> [192.249.249.3].53 dfd=5
                          nsid=29790 id=0 retry=888048802
21-Feb 00:13:18.046 qserial_query(movie.edu) QUEUED
21-Feb 00:13:18.052 next maintenance for zone 'movie.edu' in 2782 sec
21-Feb 00:13:18.056 datagram from [192.249.249.3].53, fd 5, len 380
```

```
21-Feb 00:13:18.059 qserial_answer(movie.edu, 26739)
21-Feb 00:13:18.060 qserial_answer: zone is out of date
21-Feb 00:13:18.061 startxfer(  ) movie.edu
21-Feb 00:13:18.063 /usr/etc/named-xfer -z movie.edu -f db.movie
                     -s 26738 -C 1 -P 53 -d 1 -l xfer.ddt 192.249.249.3
21-Feb 00:13:18.131 started xfer child 390
21-Feb 00:13:18.132 next maintenance for zone 'movie.edu' in 7200 sec

21-Feb 00:14:02.089 endxfer: child 390 zone movie.edu returned
                          status=1 termsig=-1
21-Feb 00:14:02.094 loadxfer(  ) "movie.edu"
21-Feb 00:14:02.094 purge_zone(movie.edu,1)

21-Feb 00:14:30.049 db_load(db.movie, movie.edu, 2, Nil)
21-Feb 00:14:30.058 next maintenance for zone 'movie.edu' in 1846 sec

21-Feb 00:17:12.478 slave zone "movie.edu" (IN) loaded (serial 26739)
21-Feb 00:17:12.486 no schedule change for zone 'movie.edu'

21-Feb 00:42:44.817 Cleaned cache of 0 RRs

21-Feb 00:45:16.046 do_zone_maint for zone movie.edu (class IN)
21-Feb 00:45:16.054 zone_maint('movie.edu')
21-Feb 00:45:16.055 qserial_query(movie.edu)
21-Feb 00:45:16.063 sysquery: send -> [192.249.249.3].53 dfd=5
                          nsid=29791 id=0 retry=888050660
21-Feb 00:45:16.066 qserial_query(movie.edu) QUEUED
21-Feb 00:45:16.067 next maintenance for zone 'movie.edu' in 3445 sec
21-Feb 00:45:16.074 datagram from [192.249.249.3].53, fd 5, len 380
21-Feb 00:45:16.077 qserial_answer(movie.edu, 26739)
21-Feb 00:45:16.078 qserial_answer: zone serial is still OK
21-Feb 00:45:16.131 next maintenance for zone 'movie.edu' in 2002 sec
```

В отличие от предыдущих фрагментов, этот содержит указатели времени в каждой строке. Датирование позволяет ясно увидеть группировку отладочных операторов.

Этот DNS-сервер является вторичным для единственной зоны, *movie.edu*. Строка, датированная временем 00:13:18.026, показывает, что наступил момент синхронизации с основным сервером. Вторичный сервер запрашивает SOA-запись зоны и сравнивает порядковые номера перед принятием решения о загрузке зоны. Строки начиная с датированной 00:13:18.059 по 00:13:18.131 содержат порядковый номер зоны (26739), информацию о том, что хранимая зоны устарела, и информацию о запуске порожденного процесса (pid 390) для передачи зоны. В момент 00:13:18.132 таймеру присваивается значение в 7200 секунд. Это время, за которое должна завершиться передача зоны. В момент 00:14:02.089 мы видим код завершения порожденного процесса. Код завершения 1 говорит о том, что данные зоны были успешно получены. Старые данные зоны удаляются (time 00:14:02.094), а новые загружаются.

Следующая проверка (время 00:14:30.058) намечена через 1846 секунд. Для данной зоны интервал обновления составляет 3600 секунд, так почему же сервер наметил проверку через 1846? DNS-сервер пытается избежать синхронизации таймера обновления. Вместо того чтобы использовать интервал, равный в точности 3600 секундам, он использует случайно выбранное значение, большее половины интервала обновления (1800), но меньшее полного интервала (3600). В 00:45:16.046 происходит повторная проверка зоны, и на этот раз обновление не требуется.

Если взять фрагмент отладочного вывода за достаточно большой промежуток времени, можно увидеть много строк, похожих на строку 00:42:44.817, – по одной на каждый час. А происходит вот что: сервер перебирает кэшированную информацию, удаляя устаревшие данные, чтобы сократить объем занимаемой памяти.

Основной DNS-сервер данной зоны – сервер BIND версии 4. Если бы основной сервер принадлежал пакету BIND версии 8, вторичный сервер получал бы уведомление при каждом изменении зоны, не дожидаясь окончания интервала обновления. Отладочный вывод вторичного сервера в этом случае выглядел бы практически так же, за исключением того, что сигналом к обновлению зоны было бы сообщение NOTIFY:

```
rcvd NOTIFY(movie.edu, IN, SOA) from [192.249.249.3].1059
qserial_query(movie.edu)
sysquery: send -> [192.249.249.3].53 dfd=5
           nsid=29790 id=0 retry=888048802
```

Вторичный DNS-сервер производит проверку зоны (BIND 9, уровень отладки 1)

Эквивалентный отладочный вывод DNS-сервера BIND 9.1.0 на уровне 1, как всегда, более лаконичен. Выглядит он так:

```
Sep 18 15:05:00.059 zone_timer: zone movie.edu/IN: enter
Sep 18 15:05:00.059 dns_zone_maintenance: zone movie.edu/IN: enter
Sep 18 15:05:00.059 queue_soa_query: zone movie.edu/IN: enter
Sep 18 15:05:00.059 soa_query: zone movie.edu/IN: enter
Sep 18 15:05:00.061 refresh_callback: zone movie.edu/IN: enter
Sep 18 15:05:00.062 refresh_callback: zone movie.edu/IN: Serial: new
2000010923, old 2000010922
Sep 18 15:05:00.062 queue_xfrin: zone movie.edu/IN: enter
Sep 18 15:05:00.070 zone_xfrdone: zone movie.edu/IN: success
Sep 18 15:05:00.070 transfer of 'movie.edu' from 192.249.249.3#53: end of
transfer
Sep 18 15:05:01.089 zone_timer: zone movie.edu/IN: enter
Sep 18 15:05:01.089 dns_zone_maintenance: zone movie.edu/IN: enter
Sep 18 15:05:19.121 notify_done: zone movie.edu/IN: enter
Sep 18 15:05:19.621 notify_done: zone movie.edu/IN: enter
```

Сообщение, датированное 15:05:00.059, показывает срабатывание таймера обновления, что приводит к выполнению DNS-сервером служебных действий для зоны (в следующей строке). Во-первых, DNS-сервер ставит в очередь запрос SOA-записи для зоны класса IN – *movie.edu* (*queue_soa_query* для того же времени) и посылает его. В 15:05:00.062 сервер обнаруживает, что зона основного DNS-сервера имеет больший порядковый номер, чем хранимая (2000010923 и 2000010922), и выполняет входящую передачу зоны (*queue_xfrin*). Через восемь миллисекунд (15:05:00.070) передача зоны закончена, а в момент 15:05:01.089 DNS-сервер обнуляет таймер обновления (*zone_timer*).

Следующие три строки показывают выполнение сервером служебных операций для *movie.edu*. К примеру, если бы некоторые DNS-серверы *movie.edu* были расположены вне зоны *movie.edu*, DNS-сервер воспользовался бы этим моментом для поиска их адресов (не только A-, но также A6- и AAAA-записей!), чтобы позже включать их в ответы. В последних двух строках мы видим отправку DNS-сервером NOTIFY-сообщений – двух, если быть точными, – DNS-серверам, перечисленным в NS-записях *movie.edu*.

Алгоритм работы DNS-клиента и отрицательное кэширование (BIND 8)

С помощью приводимых записей мы покажем, что такое алгоритм поиска и отрицательное кэширование клиента BIND версии 4.9 и более поздних с точки зрения DNS-сервера BIND 8. Мы могли бы выполнить поиск адреса для *galt.cs.purdue.edu*, как в предыдущем случае, но это не позволило бы нам оценить алгоритм поиска. Поэтому мы выполним запрос для несуществующего имени *foo.bar*. Причем дважды:

```
1)  datagram from cujo.horror.movie.edu 1109, fd 6, len 25
2)  req: nlookup(foo.bar) id 19220 type=1 class=1
3)  req: found 'foo.bar' as '' (cname=0)
4)  forw: forw -> D.ROOT-SERVERS.NET 53 ds=7 nsid=2532 id=19220 0ms retry 4sec
5)
6)  datagram from D.ROOT-SERVERS.NET 53, fd 5, len 25
7)  ncache: dname foo.bar, type 1, class 1
8)  send_msg -> cujo.horror.movie.edu 1109 (UDP 6) id=19220
9)
10) datagram from cujo.horror.movie.edu 1110, fd 6, len 42
11) req: nlookup(foo.bar.horror.movie.edu) id 19221 type=1 class=1
12) req: found 'foo.bar.horror.movie.edu' as 'horror.movie.edu' (cname=0)
13) forw: forw -> carrie.horror.movie.edu 53 ds=7 nsid=2533 id=19221 0ms
                                                                retry 4sec
14) datagram from carrie.horror.movie.edu 53, fd 5, len 42
15) ncache: dname foo.bar.horror.movie.edu, type 1, class 1
16) send_msg -> cujo.horror.movie.edu 1110 (UDP 6) id=19221
```

И еще раз поиск адреса *foo.bar*:

```
17) datagram from cujo.horror.movie.edu 1111, fd 6, len 25
18) req: nlookup(foo.bar) id 15541 type=1 class=1
19) req: found 'foo.bar' as 'foo.bar' (cname=0)
20) ns_req: answer -> cujo.horror.movie.edu 1111 fd=6 id=15541 size=25 Local
21)
22) datagram from cujo.horror.movie.edu 1112, fd 6, len 42
23) req: nlookup(foo.bar.horror.movie.edu) id 15542 type=1 class=1
24) req: found 'foo.bar.horror.movie.edu' as 'foo.bar.horror.movie.edu'
(cname=0)
25) ns_req: answer -> cujo.horror.movie.edu 1112 fd=6 id=15542 size=42 Local
```

Давайте взглянем на клиент и его алгоритм поиска. Первое имя, для которого производится поиск адреса (строка 2), в точности совпадает с тем именем, которое мы набрали. Поскольку имя содержало по меньшей мере одну точку, поиск для него был выполнен без предварительных модификаций. Этот поиск вернул отрицательный результат, к имени был добавлен домен *horror.movie.edu*, и поиск повторился. (Клиенты BIND версий до 4.9 попытались бы добавить одновременно *horror.movie.edu* и *movie.edu*.)

В седьмой строке отображено кэширование отрицательного ответа (*ncache*). Если то же имя используется в поиске в ближайшие несколько минут (строка 19), сервер может дать немедленный ответ, что имя не существует, поскольку этот отрицательный ответ все еще хранится в кэше. (Если слова не убеждают, сравните строки 3 и 19. Строка 3: ничего не найдено по имени *foo.bar*, а в строке 19 это имя неожиданно находится.)

Алгоритм работы DNS-клиента и отрицательное кэширование (BIND 9)

Вот так выглядит отладочный вывод DNS-сервера BIND 9.1.0 при дважды повторенном поиске для *foo.bar*:

```
Sep 18 15:45:42.944 client cujo.horror.movie.edu#1044: query: foo.bar A
Sep 18 15:45:42.945 createfetch: foo.bar. A
Sep 18 15:45:42.945 createfetch: . NS
Sep 18 15:45:43.425 client cujo.horror.movie.edu#1044: query: foo.bar.
horror.movie.edu A
Sep 18 15:45:43.425 createfetch: foo.bar.horror.movie.edu. A
```

Эта диагностика намного более краткая, чем в BIND 8, но в ней содержится необходимая информация. Первая строка, датированная 15:45:42.944, показывает первый запрос адреса *foo.bar*, поступивший от клиента *cujo.horror.movie.edu* (помните, что мы пропустили вывод через наш волшебный фильтр, преобразующий IP-адреса в имена, речь о котором пойдет ниже). Следующие две строки показывают, что

DNS-сервер выполнил две задачи (*createfetch*) для поиска адреса *foo.bar*: во-первых, собственно задачу поиска адреса для имени *foo.bar*, а во-вторых – вспомогательную задачу поиска NS-записи для корневой зоны, которая необходима для завершения поиска по имени *foo.bar*. Получив текущую NS-запись для корневой зоны, DNS-сервер запрашивает у корневого DNS-сервера адрес для имени *foo.bar* и получает ответ, что домен высшего уровня с именем *bar* не существует. Но мы, к сожалению, этого не видим.

Строка, датированная 15:45:43.425, показывает использование списка поиска узлом *cujo.horror.movie.edu* и поиск имени *foo.bar.horror.movie.edu*. DNS-сервер порождает задачу (*createfetch*) поиска адреса этого доменного имени.

При повторном поиске *foo.bar* мы видим следующее:

```
Sep 18 15:45:46.557 client cujo.horror.movie.edu#1044: query: foo.bar A
Sep 18 15:45:46.558 client cujo.horror.movie.edu#1044: query:
foo.bar.horror.movie.edu A
```

Все заметили отсутствие строк *createfetch*? Наш DNS-сервер кэширует отрицательные ответы.

Инструменты

Подведем итоги. Мы рассказывали о программе, преобразующей IP-адреса в имена с целью облегчения чтения отладочного вывода. Вот эта программа на языке Perl:

```
#!/usr/bin/perl -n

use "Socket";

if (/\b)(\d+\.\d+\.\d+\.\d+)\b/) {
     $addr = pack('C4', split(/\./, $1));
     ($name, $rest) = gethostbyaddr($addr, &AF_INET);
     if($name) {s/$1/$name/;
}

print;
```

Не рекомендуется передавать вывод *named.run* этому сценарию через конвейер при включенной отладке, поскольку сценарий выполняет собственные запросы к DNS-серверу.

14

- *Виновата ли служба NIS?*
- *Инструменты и методы*
- *Перечень возможных проблем*
- *Проблемы перехода на новую версию*
- *Проблемы сосуществования и версий*
- *Ошибки TSIG*
- *Симптомы проблем*

Разрешение проблем DNS и BIND

– Конечно! Ведь без него будет очень скучно жить на свете! – сказал Деликатес.
– У тебя есть свой конек?
– Нет, у меня есть кошка, – сказала Алиса.
– Ее зовут…
– Прекрасно! – обрадовался Грифон.
– Давно пора тебе рассказать нам о себе и о своих приключениях!

В последних двух главах мы рассказали о применениях инструментов *nslookup* и *dig*, а также о том, как интерпретировать отладочную информацию, предоставляемую DNS-сервером. В этой главе мы покажем, как применять эти программы совместно с традиционными сетевыми инструментами Unix, в частности верным старым *ping*, для диагностирования и разрешения встречающихся на практике проблем DNS и BIND.

Разрешение проблем – дело, которому сложно научить. Начинать следует с симптомов и пытаться по ним найти причину. Нельзя рассмотреть всю гамму проблем, которые могут встретиться при использовании Интернета, но мы, разумеется, приложим все усилия, чтобы показать, как следует диагностировать наиболее распространенные. В процессе мы надеемся обучить читателей технике диагностирования и разрешения проблем, которая пригодится при борьбе с более редкими неприятностями, о которых мы здесь не упоминаем.

Виновата ли служба NIS?

Прежде чем начать обсуждение даигностики и разрешения конкретных проблем DNS и BIND, мы должны удостовериться, что читатели понимают, как отличить проблему, связанную с NIS, от проблемы, связанной с DNS. На узлах с работающей службой NIS может быть непросто понять, какая из служб является источником проблем. К примеру, классический BSD-*nslookup* не обращает внимания на NIS. Можно работать с программой *nslookup* на системе Sun, бесконечно посылая запросы DNS-серверам, а все прочие службы при этом будут использовать NIS.

Как узнать, кто виноват? Некоторые поставщики изменяют *nslookup* на предмет использования NIS в качестве службы имен, если существует настроенная NIS. Так, *nslookup* HP-UX сообщает, что намеревается посылать запросы серверу NIS – при запуске:

```
% nslookup
Default NIS Server:  terminator.movie.edu
Address:  192.249.249.3

>
```

На узлах с обычной версией *nslookup* чаще всего можно использовать команду *ypmatch*, чтобы понять, что используется – DNS или NIS. *ypmatch* печатает пустую строку после информации об узле, полученной от DNS-сервера. В следующем примере ответ получен от NIS:

```
% ypmatch ruby hosts
140.186.65.25   ruby ruby.ora.com
%
```

А в этом – от DNS-сервера:

```
% ypmatch harvard.harvard.edu hosts
128.103.1.1     harvard.harvard.edu

%
```

Следует помнить, что это работает в SunOS 4.1.1, но может не работать в любой из более поздних версий SunOS. Насколько нам известно, это просто ошибка/особенность, которая может исчезнуть в следующей же версии.

Наиболее правильный способ понять, что ответ получен от NIS, – использовать команду *ypcat* для просмотра содержимого базы данных *hosts*. К примеру, чтобы узнать, содержится ли узел *andrew.cmu.edu* в карте узлов NIS, можно воспользоваться командой:

```
% ypcat hosts | grep andrew.cmu.edu
```

Если ответ найден в NIS (и известно, что эта служба запрашивается первой), то найдена и причина проблемы.

И наконец, в системах Unix, использующих файл *nsswitch.conf*, можно узнать порядок опроса служб, найдя в этом файле запись для базы данных *hosts*. Следующая строка показывает, что в первую очередь запрашивается служба NIS:

```
hosts:    nis dns files
```

а эта – что клиент прежде всего посылает запрос службе доменных имен:

```
hosts:    dns nis files
```

Более подробная информация по синтаксису и семантике файла *nsswitch.conf* содержится в главе 6 «Конфигурирование узлов».

Приведенные советы должны помочь определить виновника проблем либо по меньшей мере исключить из рассмотрения одного из подозреваемых. Если после этого DNS все еще находится под подозрением – просто читайте эту главу.

Инструменты и методы

В последних двух главах мы изучали *nslookup*, *dig* и отладочный вывод DNS-сервера. Прежде чем продолжить, возьмем на вооружение несколько новых инструментов, которые могут быть полезны при отладке: *named-xfer*, дампы базы данных и регистрация запросов.

Как применять named-xfer

named-xfer – программа, вызываемая DNS-серверами BIND 4 и 8 для выполнения передачи зоны. (Как, вероятно, помнят читатели, DNS-серверы BIND 9 – многопотоковые, им не нужна отдельная программа для получения зон: они просто выполняют эту операцию в паралелльном потоке.) *named-xfer* проверяет, является ли актуальной копия зональных данных, хранимая вторичным сервером, и при необходимости производит передачу зоны. (В версиях 4.9 и 8 *named* самостоятельно проверяет актуальность данных, чтобы избежать порождения ненужного процесса.)

В главе 13 «Чтение отладочного вывода BIND» мы приводили отладочный вывод вторичного DNS-сервера BIND 8, созданный при проверке хранимой зоны. При получении зоны вторичный сервер создал порожденный процесс (*named-xfer*) для копирования данных в локальную файловую систему. Мы умолчали о том, что *named-xfer* можно запускать и вручную, не дожидаясь, когда это сделает *named*, и что можно предписать этой программе создание собственной отладочной диагностики (без вовлечения *named*).

Это может быть полезно, если речь идет о проблеме, связанной с передачей зоны, но нет времени ждать, когда сервер *named* сам ее инициии-

рует. Чтобы вручную изучить процесс передачи зоны, следует указать ряд ключей командной строки:

```
% /usr/sbin/named-xfer
Usage error: no domain
Usage: named-xfer
        -z zone_to_transfer
        -f db_file
        [-i ixfr_file]
        [-s serial_no]
        [-d debug_level]
        [-l debug_log_file]
        [-t trace_file]
        [-p port]
        [-S] [-Z]
        [-C class]
        [-x axfr-src]
        [-T tsig_info_file]
        servers [-ixfr|-axfr]...
```

Это вывод *named-xfer* из пакета BIND 8.2.3. В более ранних версиях *named-xfer* набор ключей меньше.

named при вызове *named-xfer* использует ключ *-z* (для указания зоны, для которой следует производить проверку), ключ *-f* (для указания имени файла данных этой зоны, приводимого в *named.boot* или *named.conf*), ключ *-s* (для указания порядкового номера зоны из хранимой вторичным сервером SOA-записи), а также перечисляет адреса серверов, с которыми предписано сверяться вторичному серверу (IP-адреса предписания *masters* в операторе *zone* в файле *named.conf* либо инструкции *secondary* в *named.boot*). Если *named* работает в режиме отладки, используется также ключ *-d* для указания уровня отладки для *named-xfer*. Все остальные ключи предназначены для диагностики проблем, они связаны с инкрементальной передачей зоны, TSIG-подписями и другими подобными вещами.

При запуске *named-xfer* пользователем уровень отладки также может быть задан ключом командной строки *-d*. (Но следует помнить, что уровни отладки выше третьего приводят к получению огромного объема отладочной информации при успешной передаче зоны!) Альтернативное имя для файла отладочного вывода можно указать с помощью ключа *-l*. По умолчанию запись сообщений происходит в файл */var/tmp/xfer.ddt.XXXXXX*, где *XXXXXX* – суффикс, добавляемый в целях обеспечения уникальности имени файла, либо файл с тем же именем в */usr/tmp*. Можно также указать имя узла, к которому происходит обращение, вместо его IP-адреса.

К примеру, следующая командная строка позволяет понять, происходит ли передача зоны с узла *terminator.movie.edu*:

```
% /usr/sbin/named-xfer -z movie.edu -f /tmp/db.movie -s 0 terminator
```

```
% echo $?
4
```

В данной команде был использован нулевой порядковый номер (serial), чтобы гарантировать попытку передачи зоны *named-xfer*, даже если обновление не требуется. Нуль – это специальный порядковый номер, при указании которого *named-xfer* произведет передачу зоны, не обращая внимания на реально существующий порядковый номер. Кроме того, мы предписали *named-xfer* помещать новые файлы данных зоны в каталог */tmp*, не перезаписывая хранимые копии.

Успешно ли прошла передача? Это можно определить по коду, возвращаемому программой *named-xfer*. В BIND версии 8.1.2 или более ранней код завершения может принимать следующие значения:

0 Данные зоны соответствуют хранимым на основном сервере, передача не требуется.

1 Зона успешно получена.

2 Узел (или узлы), которым программа *named-xfer* отправила запросы, недоступны, либо произошла ошибка, возможно, соответствующее сообщение записано в log-файл *syslog*.

3 Произошла ошибка, соответствующее сообщение записано в log-файл *syslog*.

В BIND 8.2 были добавлены новые значения для инкрементальной передачи зоны:

4 Успешная AXFR-передача (полная) зоны.

5 Успешная IXFR-передача (инкрементальная) зоны.

6 Основной DNS-сервер вернул AXFR *named-xfer* на запрос IXFR.

Для DNS-сервера, даже поддерживающего IXFR, является абсолютно допустимым реагировать полной передачей зоны на запрос инкрементальной передачи. К примеру, на мастер-сервере может отсутствовать часть информации об изменениях, произошедших в данных зоны.

Обратите внимание, что в BIND версии 8.2 и более поздних *named-xfer* никогда не возвращает значение 1. Оно было заменено значениями с 4 по 6.

Как обойтись без named-xfer?

Если было произведено обновление до BIND 9 и исполняемый файл *named-xfer* отсутствует, можно по-прежнему использовать *nslookup* или *dig* для получения зоны. Любой из инструментов, умеющих делать запросы, позволяет получить ту же информацию, что и *named-xfer*.

Для идентичной передачи зоны можно исопльзовать *dig* следующим образом:

```
% dig @terminator.movie.edu movie.edu axfr
```

В *nslookup* можно изменить используемый DNS-сервер и выполнить команду *ls -d* в диалоговом режиме.

К сожалению, и *dig*, и *nslookup* не столь изящны в сообщениях об ошибках, как *named-xfer*. Если *nslookup* не может получить зону, то обычно рапортует о «неопределенной ошибке»:

```
> ls movie.edu
 [terminator.movie.edu]
*** Can't list domain movie.edu: Unspecified error
```

Ошибка может быть вызвана списком доступа *allow-transfer*, тем фактом, что *terminator.movie.edu* в действительности не является авторитативным для *movie.edu* и целым рядом других проблем. Чтобы понять причину ошибки, можно выполнить дополнительные запросы либо свериться с сообщениями DNS-мастер-сервера, доступными в log-файле демона *syslog*.

Читаем дамп базы данных

Пристальное изучения дампа внутренней базы данных DNS-сервера, включающей кэшированную информацию, также может быть полезно при поиске ошибок. Команды *ndc dumpdb* и *rndc dumpdb* приводят к созданию сервером *named* записи авторитативных данных, кэшированных данных и данных корневых указателей в файл *named_dump.db* в рабочем каталоге BIND (либо в файл */usr/tmp/named_dump.db*, или */var/tmp/named_dump.db*, в случае использования BIND 4).[1] Ниже приводится пример файла *named_dump.db*. Авторитативные данные и кэшированные записи отображены в начале файла без определенного порядка, за ними следуют данные корневых указателей:

```
; Dumped at Tue Jan  6 10:49:08 1998
;; ++zone table++
; 0.0.127.in-addr.arpa (type 1, class 1, source db.127.0.0)
;   time=0, lastupdate=0, serial=1,
;   refresh=0, retry=3600, expire=608400, minimum=86400
;   ftime=884015430, xaddr=[0.0.0.0], state=0041, pid=0
;; --zone table--
; Note: Cr=(auth,answer,addtnl,cache) tag only shown for non-auth RR's
; Note: NT=milliseconds for any A RR which we've used as a nameserver
; --- Cache & Data ---
$ORIGIN .
.   518375  IN      NS  G.ROOT-SERVERS.NET.   ;Cr=auth [128.8.10.90]
    518375  IN      NS  J.ROOT-SERVERS.NET.   ;Cr=auth [128.8.10.90]
    518375  IN      NS  K.ROOT-SERVERS.NET.   ;Cr=auth [128.8.10.90]
    518375  IN      NS  L.ROOT-SERVERS.NET.   ;Cr=auth [128.8.10.90]
```

[1] BIND 9.1.0 является первой версией пакета BIND 9, в которой поддерживается создание образа (дампа) базы данных.

```
    518375  IN      NS  M.ROOT-SERVERS.NET.   ;Cr=auth [128.8.10.90]
    518375  IN      NS  A.ROOT-SERVERS.NET.   ;Cr=auth [128.8.10.90]
    518375  IN      NS  H.ROOT-SERVERS.NET.   ;Cr=auth [128.8.10.90]
    518375  IN      NS  B.ROOT-SERVERS.NET.   ;Cr=auth [128.8.10.90]
    518375  IN      NS  C.ROOT-SERVERS.NET.   ;Cr=auth [128.8.10.90]
    518375  IN      NS  D.ROOT-SERVERS.NET.   ;Cr=auth [128.8.10.90]
    518375  IN      NS  E.ROOT-SERVERS.NET.   ;Cr=auth [128.8.10.90]
    518375  IN      NS  I.ROOT-SERVERS.NET.   ;Cr=auth [128.8.10.90]
    518375  IN      NS  F.ROOT-SERVERS.NET.   ;Cr=auth [128.8.10.90]
EDU  86393  IN      SOA A.ROOT-SERVERS.NET.  hostmaster.INTERNIC.NET. (
         1998010500 1800 900 604800 86400 )   ;Cr=addtnl [128.63.2.53]
$ORIGIN  0.127.in-addr.arpa.
0        IN    SOA cujo.movie.edu. root.cujo.movie.edu. (
         1998010600 10800 3600 608400 86400 )        ;Cl=5
         IN    NS  cujo.movie.edu.   ;Cl=5
$ORIGIN  0.0.127.in-addr.arpa.
1        IN    PTR localhost.    ;Cl=5
$ORIGIN EDU.
PURDUE   172787  IN  NS  NS.PURDUE.EDU.           ;Cr=addtnl [192.36.148.17]
         172787  IN  NS  MOE.RICE.EDU.            ;Cr=addtnl [192.36.148.17]
         172787  IN  NS  PENDRAGON.CS.PURDUE.EDU.  ;Cr=addtnl [192.36.148.17]
         172787  IN  NS  HARBOR.ECN.PURDUE.EDU.    ;Cr=addtnl [192.36.148.17]
$ORIGIN  movie.EDU.
;cujo    593     IN  SOA  A.ROOT-SERVERS.NET. hostmaster.INTERNIC. NET. (
;        1998010500 1800 900 604800 86400 );EDU.; NXDOMAIN  ;-$
   ;Cr=auth [128.63.2.53]
$ORIGIN   RICE.EDU.
MOE      172787  IN  A   128.42.5.4        ;NT=84 Cr=addtnl [192.36.148.17]
$ORIGIN   PURDUE.EDU.
CS       86387   IN  NS  pendragon.cs.PURDUE.edu.    ;Cr=addtnl [128.42.5.4]
         86387   IN  NS  ns.PURDUE.edu.              ;Cr=addtnl [128.42.5.4]
         86387   IN  NS  harbor.ecn.PURDUE.edu.      ;Cr=addtnl [128.42.5.4]
         86387   IN  NS  moe.rice.edu.               ;Cr=addtnl [128.42.5.4]
NS       172787   IN  A  128.210.11.5        ;NT=4 Cr=addtnl [192.36.148.17]
$ORIGIN   ECN.PURDUE.EDU.
HARBOR   172787  IN   A  128.46.199.76       ;NT=6 Cr=addtnl [192.36.148.17]
$ORIGIN   CS.PURDUE.EDU.
galt     86387   IN   A  128.10.2.39                ;Cr=auth [128.42.5.4]
PENDRAGON  172787  IN  A  128.10.2.5         ;NT=20 Cr=addtnl [192.36.148.17]
$ORIGIN   ROOT-SERVERS.NET.
K        604775    IN  A  193.0.14.129       ;NT=10 Cr=answer [128.8.10.90]
A        604775    IN  A  198.41.0.4         ;NT=20 Cr=answer [128.8.10.90]
L        604775    IN  A  198.32.64.12       ;NT=8 Cr=answer [128.8.10.90]
B        604775    IN  A  128.9.0.107        ;NT=9 Cr=answer [128.8.10.90]
M        604775    IN  A  202.12.27.33       ;NT=20 Cr=answer [128.8.10.90]
C        604775    IN  A  192.33.4.12        ;NT=17 Cr=answer [128.8.10.90]
D        604775    IN  A  128.8.10.90        ;NT=11 Cr=answer [128.8.10.90]
E        604775    IN  A  192.203.230.10     ;NT=9 Cr=answer [128.8.10.90]
F        604775    IN  A  192.5.5.241        ;NT=73 Cr=answer [128.8.10.90]
G        604775    IN  A  192.112.36.4       ;NT=14 Cr=answer [128.8.10.90]
H        604775    IN  A  128.63.2.53        ;NT=160 Cr=answer [128.8.10.90]
```

```
I        604775    IN  A  192.36.148.17      ;NT=102 Cr=answer [128.8.10.90]
J        604775    IN  A  198.41.0.10        ;NT=21 Cr=answer [128.8.10.90]
; --- Hints ---
$ORIGIN .
.   3600           IN  NS  A.ROOT-SERVERS.NET.     ;Cl=0
    3600           IN  NS  B.ROOT-SERVERS.NET.     ;Cl=0
    3600           IN  NS  C.ROOT-SERVERS.NET.     ;Cl=0
    3600           IN  NS  D.ROOT-SERVERS.NET.     ;Cl=0
    3600           IN  NS  E.ROOT-SERVERS.NET.     ;Cl=0
    3600           IN  NS  F.ROOT-SERVERS.NET.     ;Cl=0
    3600           IN  NS  G.ROOT-SERVERS.NET.     ;Cl=0
    3600           IN  NS  H.ROOT-SERVERS.NET.     ;Cl=0
    3600           IN  NS  I.ROOT-SERVERS.NET.     ;Cl=0
    3600           IN  NS  J.ROOT-SERVERS.NET.     ;Cl=0
    3600           IN  NS  K.ROOT-SERVERS.NET.     ;Cl=0
    3600           IN  NS  L.ROOT-SERVERS.NET.     ;Cl=0
    3600           IN  NS  M.ROOT-SERVERS.NET.     ;Cl=0
$ORIGIN   ROOT-SERVERS.NET.
K     3600         IN   A  193.0.14.129      ;NT=11 Cl=0
L     3600         IN   A  198.32.64.12      ;NT=9 Cl=0
A     3600         IN   A  198.41.0.4        ;NT=10 Cl=0
M     3600         IN   A  202.12.27.33      ;NT=11 Cl=0
B     3600         IN   A  128.9.0.107       ;NT=1288 Cl=0
C     3600         IN   A  192.33.4.12       ;NT=21 Cl=0
D     3600         IN   A  128.8.10.90       ;NT=1288 Cl=0
E     3600         IN   A  192.203.230.10    ;NT=19 Cl=0
F     3600         IN   A  192.5.5.241       ;NT=23 Cl=0
G     3600         IN   A  192.112.36.4      ;NT=18 Cl=0
H     3600         IN   A  128.63.2.53       ;NT=11 Cl=0
I     3600         IN   A  192.36.148.17     ;NT=21 Cl=0
J     3600         IN   A  198.41.0.10       ;NT=13 Cl=0
```

DNS-сервер, создавший этот файл, являлся авторитативным только для зоны *0.0.127.in-addr.arpa*. Этим сервером был произведен поиск для двух имен: *galt.cs.purdue.edu* и *cujo.movie.edu*. В процессе поиска для *galt.cs.purdue.edu* сервер кэшировал не только адрес узла *galt*, но также список DNS-серверов для зоны *purdue.edu* и их адреса. Имя *cujo.movie.edu* в действительности не существует (как и зона *movie.edu*, используемая только в наших примерах), поэтому сервер кэшировал отрицательный ответ. В файле дампа отрицательный ответ закомментирован (строка начинается с символа точки с запятой), и вместо данных приводится причина получения отрицательного ответа (NXDOMAIN). Обратите внимание, что значение TTL довольно низкое (593). В BIND 8.2 и более поздних версиях DNS-сервера отрицательные ответы кэшируются на время, определяемое последним полем SOA-записи, и это время обычно существенно меньше, чем стандартное значение TTL для всей зоны.

В разделе указателей – в конце файла – содержатся данные из файла *db.cache*. TTL для данных указателей уменьшается и может обратиться в нуль, но указатели никогда не удаляются.

После некоторых RR-записей присутствует точка с запятой и строка *NT*=. Это адресные записи DNS-серверов. Число определяет время передачи сигнала для конкретного DNS-сервера, оно хранится DNS-сервером для ранжирования «собеседников» по скорости реагирования; при следующем запросе будет использован сервер с наименьшим показателем RTT.

Кэшированные данные легко отличить от всех остальных – их записи дополняются отметкой *достоверности* (*credibility*, *Cr*=) и иногда – IP-адресом сервера, от которого получены данные.[1] Данные зоны и данные указателей дополняются отметкой *Cl*=, которая содержит номер уровня (*count of level*) в дереве доменов (корневой уровень имеет номер 0, *foo* будет иметь уровень 1, *foo.foo* – уровень 2 и т. д.). Сейчас мы сделаем отступление и объясним понятие достоверности.

Разница между версиями 4.8.3 и 4.9 включает появление метрики достоверности. Она позволяет DNS-серверу принимать более продуманные решения относительно того, что следует делать с новыми данными, полученными от удаленного сервера.

У серверов версии 4.8.3 было только два уровня достоверности – локально-авторитативные данные и все остальные. Локально-авторитативными назывались данные в файлах данных зоны – DNS-серверу прекрасно известно, когда стоит, а когда не стоит обновлять хранимую копию. Данные, получаемые от всех прочих DNS-серверов, имели одинаковую достоверность.

Вот реальная ситуация и действия сервера версии 4.8.3 в этой ситуации. Предположим, DNS-сервер производил поиск адреса для *terminator.movie.edu* и получил авторитативный ответ от DNS-сервера зоны *movie.edu*. (Авторитативный ответ – наилучший из существующих.) Некоторое время спустя, при поиске для имени *foo.oreilly.com*, DNS-сервер получает еще одну адресную запись для имени *terminator.movie.edu*, но на этот раз – в качестве части информации о делегировании для *oreilly.com* (*terminator.movie.edu* является для этой зоны вторич-

[1] DNS-сервер отображает IP-адрес удаленного сервера, если это возможно. В BIND 8.2 и более поздних версиях DNS-серверов IP-адрес доступен только в том случае, если использовано соответствующее предписание – *host-statistics*, которое было описано в главе 8 «Развитие домена». В более ранних DNS-серверах BIND 4.9 и BIND 8 режим включен по умолчанию. *host-statistics* сохраняет впечатляющую статистику по каждому DNS-серверу и клиенту, с которым когда-либо контактировал данный DNS-сервер, что в некоторых случаях очень полезно (скажем, позволяет определить, от какого DNS-сервера получена определенная запись), но связано с потреблением повышенных объемов памяти.

ным DNS-сервером). DNS-сервер версии 4.8.3 обновил бы кэшированную адресную запись для узла *terminator.movie.edu*, несмотря на то, что данные поступили от DNS-сервера *com*, а не от авторитативного DNS-сервера *movie.edu*. Но ведь DNS-серверы *com* и *movie.edu* возвращают одинаковую информацию по *terminator.movie.edu*, и проблемы никакой нет? Да-да, а в южной Калифорнии никогда не идут дожди.

DNS-серверы версии 4.9 и более поздних проявляют некоторую разборчивость. Как и серверы версии 4.8.3, они считают хранимые данные зоны неприкосновенными – в принципе. При этом проводится различие между различными типами данных, получаемых от удаленных DNS-серверов. Вот иерархия достоверности данных от удаленных серверов, в порядке понижения достоверности:

auth

Записи извлечены из ответов авторитативных DNS-серверов (из раздела ответа сообщений с установленным битом авторитативности).

answer

Записи извлечены из неавторитативных либо кэшированных ответов (из раздела ответа сообщений со сброшенным битом авторитативности).

addtnl

Записи извлечены из оставшейся части сообщения – разделов авторитативности и дополнительного. Раздел авторитативности сообщения содержит NS-записи, делегирующие зону авторитативному DNS-серверу. Дополнительный раздел содержит адресные записи, которые, возможно, дополняют информацию из других разделов (скажем, это адресные записи, которые сопутствуют NS-записям из раздела авторитативности).

Существует одно исключение из этого правила: когда DNS-сервер производит начальное заполнение кэша корневых DNS-серверов, записи, имеющие достоверность *addtnl* получают достоверность *answer*, чтобы снизить вероятность их случайного изменения. Обратите внимание, в приведенном дампе адресные записи корневых DNS-серверов имеют достоверность *answer*, но при этом адресные записи для DNS-серверов *purdue.edu* имеют достоверность *addtnl*.

В описанной только что ситуации DNS-сервер версии 4.9 или более поздней не станет заменять авторитативные данные (достоверность *auth*) для *terminator.movie.edu* данными делегирования (достоверность *addtnl*), поскольку авторитативный ответ имеет большую достоверность.

Регистрация запросов

В BIND версии 4.9 появилась *регистрация запросов* (*query logging*), которая может облегчить диагностирование некоторых проблем. Когда регистрация запросов включена, работающий DNS-сервер записы-

вает каждый запрос в log-файл демона *syslog*. Такая возможность полезна для поиска ошибок в настройках клиента, поскольку появляется способ проверить, что имя, для которого выполняется запрос, идентично тому, о котором думал пользователь.

Прежде всего следует убедиться, что сообщения LOG_INFO регистрируются демоном *syslog* в потоке *daemon*. После этого необходимо включить регистрацию запросов одним из существующих способов: в BIND 4.9 применить настройку *options query-log* в загрузочном файле DNS-сервера; в BIND 4.9 или BIND 8 запустить DNS-сервер с ключом командной строки *-q* или послать команду *ndc querylog* работающему DNS-серверу. В BIND версии 9.1.0 и более поздних версий (в более ранних версиях BIND 9 регистрация запросов не поддерживается) следует использовать команду *rndc querylog*. В log-файле *syslog* начнут появляться сообщения примерно следующего характера:

```
Feb 20 21:43:25 terminator named[3830]:
                    XX+ /192.253.253.2/carrie.movie.edu/A
Feb 20 21:43:32 terminator named[3830]:
                    XX+ /192.253.253.2/4.253.253.192.in-addr.arpa/PTR
```

Либо, в случае BIND 9, такие:

```
Jan 13 18:32:25 terminator named[13976]: info: client 192.253.253.2#1702:
query: carrie.movie.edu IN A
Jan 13 18:32:42 terminator named[13976]: info: client 192.253.253.2#1702:
query: 4.253.253.192.in-addr.arpa IN PTR
```

Сообщения включают IP-адрес узла, сделавшего запрос, а также собственно запрос. Поскольку первый пример относится к DNS-серверу BIND 8.2.3, а запросы являются рекурсивными, сообщения в этом примере начинаются с подстроки XX+. Итеративные запросы начинаются с подстроки XX. (DNS-серверы версий до 8.2.1 не различают в сообщениях рекурсивные и нерекурсивные запросы.) Инверсные запросы включают дефис перед типом записи (для инверсного запроса адресной записи присутствует подстрока «-A» вместо «A»). После регистрации достаточного количества запросов выключить регистрацию можно с помощью повторного выполнения команды *ndc querylog* или *rndc querylog*.

Если приходится использовать более старую версию DNS-сервера BIND 9, получаемые запросы можно увидеть в отладочном выводе *named* первого уровня 1.

Перечень возможных проблем

Теперь, когда мы дали читателям рабочий инструмент, поговорим о том, как использовать этот инструмент для диагностирования реальных проблем. Отдельные проблемы очень легко выявить и решить.

Мы рассмотрим их в рабочем порядке, поскольку они встречаются довольно часто и вызваны стандартными ошибками. Итак, участники конкурса в порядке общей очереди. Мы называем эти ошибки «Чертовой дюжиной».

1. Не был увеличен порядковый номер зоны

Главным симптомом этой проблемы является нежелание DNS-серверов получать изменения, которые были внесены в данные зоны на первичном мастер-сервере. Вторичные серверы считают, что зона не изменилась, поскольку порядковый номер остался тем же.

Как проверить, был ли изменен порядковый номер зоны? К сожалению, это непросто. Если вы не помните, каким был последний порядковый номер, а сам по себе номер не содержит информации о том, когда он был обновлен, прямого способа ответить на этот вопрос не существует.[1] При перезагрузке первичного DNS-сервера происходит загрузка обновленного файла зоны, вне зависимости от того, был ли изменен порядковый номер. Сервер читает временную отметку для файла, и если файл изменялся с момента последней загрузки зоны, загружает файл. Лучшее, что можно сделать, – воспользоваться *nslookup* для сравнения данных, возвращаемых первичным и вторичным серверами. Если данные различаются, то, вероятно, вы забыли увеличить порядковый номер. Если вы можете вспомнить, какие изменения были внесены, просмотрите зоны в поиске этих изменений. В противном случае можно попробовать получить зону с основного и вторичного DNS-серверов, отсортировать результаты и использовать программу *diff* для поиска различий.

Есть и хорошая новость. Иногда определить, была ли синхронизирована зона, – непросто, но можно довольно легко произвести синхронизацию. Достаточно увеличить порядковый номер зоны в копии данных, которая хранится на первичном DNS-мастер-сервере, и перезагрузить зону на этом сервере. Вторичные серверы должны синхронизировать зону в пределах интервалов обновления либо быстрее, если используется уведомление NOTIFY. Чтобы убедиться, что вторичные серверы получили новые данные, можно вручную выполнить *named-xfer* (само собой – на вторичных серверах):

```
# /usr/sbin/named-xfer -z movie.edu -f db.movie -s 0 terminator.movie.edu
# echo $?
```

Если *named-xfer* возвращает значение 1 или 4, зона была успешно получена. Прочие значения означают, что зона не была получена либо

[1] С другой стороны, если использовать для создания порядкового номера дату, как делают многие (скажем, 2001010500 – первое обновление данных пятого января 2001 года), на вопрос с обновлением и датой обновления порядкового номера можно ответить буквально за пару секунд.

из-за ошибки, либо потому, что вторичный сервер считает хранимую зону актуальной. (Подробности см. выше, в разделе «Как применять named-xfer».)

Существует еще одна вариация темы. Она встречается, когда администратор применяет инструмент вроде *h2n* для автоматического создания файлов данных зоны на основе таблицы узлов. В этом случае очень соблазнительно бывает удалить старые файлы зоны и создать новые – с нуля. Некоторые администраторы время от времени так поступают, ошибочно полагая, что данные из старых файлов зоны могут самостоятельно перебраться в новые. Проблема с удалением файлов данных зоны заключается в том, что в отсутствие старого файла данных, из которого можно прочитать старый порядковый номер, *h2n* начинает нумерацию с порядкового номера 1. Если порядковый номер зоны на первичном DNS-мастер-сервере внезапно становится единицей (предыдущее значение, к примеру, 598), дополнительные DNS-серверы (версии 4.8.3 или более ранних) не будут жаловаться, считая, что хранимые зоны актуальны и нет необходимости производить передачу. Серверы версии 4.9 и более поздних очень бдительны и запишут соответствующее предупреждающее сообщение в log-файл *syslog*:

```
Jun  7 20:14:26 wormhole named[29618]: Zone "movie.edu"
                (class 1) SOA serial# (1) rcvd from [192.249.249.3]
                 is < ours (112)
```

Так что если порядковый номер на первичном DNS-мастер-сервере выглядит подозрительно маленьким, следует выяснить, какие порядковые номера на дополнительных серверах, и сравнить результаты:

```
% nslookup
Default Server:  terminator.movie.edu
Address:  192.249.249.3

> set q=soa
> movie.edu.
Server:  terminator.movie.edu
Address:  192.249.249.3

movie.edu
        origin = terminator.movie.edu
        mail addr = al.robocop.movie.edu
        serial = 1
        refresh = 10800 (3 hours)
        retry   = 3600 (1 hour)
        expire  = 604800 (7 days)
        minimum ttl = 86400 (1 day)
> server wormhole.movie.edu.
Default Server:  wormhole.movie.edu
Addresses:  192.249.249.1, 192.253.253.1

> movie.edu.
Server:  wormhole.movie.edu
```

```
Addresses:  192.249.249.1, 192.253.253.1

movie.edu
        origin = terminator.movie.edu
        mail addr = al.robocop.movie.edu
        serial = 112
        refresh = 10800 (3 hours)
        retry   = 3600 (1 hour)
        expire  = 604800 (7 days)
        minimum ttl = 86400 (1 day)
```

wormhole.movie.edu, будучи вторичным DNS-сервером для *movie.edu*, не должен иметь порядковый номер больший, чем у первичного DNS-мастер-сервера, так что здесь явно что-то не в порядке.

Кстати говоря, проблему довольно легко обнаружить с помощью программы, которую мы напишем в главе 15 «Программирование с помощью функций библиотеки клиента».

2. Не был перезагружен первичный DNS-мастер-сервер

Иногда, изменив файл настройки или файл данных зоны, администратор забывает перезагрузить первичный DNS-мастер-сервер. В процессе работы DNS-сервер не производит автоматической проверки изменения временных отметок файлов и, соответственно, не догадается самостоятельно о том, что произошли изменения. Так что любые внесенные изменения не будут отражены в данных, поставляемых DNS-сервером: новые зоны не будут загружены и новые записи не будут передаваться вторичным серверам.

Чтобы выяснить, когда в последний раз был перезагружен DNS-сервер, можно обратиться к log-файлу демона *syslog* в поиске последней записи такого рода (DNS-сервер BIND 9):

```
Mar  8 17:22:08 terminator named[22317]: loading configuration from '/etc/
named.conf'
```

Для BIND 4.9 или BIND 8 соответствующая запись выгляди так:

```
Mar  8 17:22:08 terminator named[22317]: reloading nameserver
```

Эти сообщения позволяют определить, когда в последний раз выполнялась команда перезагрузки для DNS-сервера. Если DNS-сервер был аварийно (принудительно) завершен, а затем повторно запущен, для сервера BIND 9 будет присутствовать такое сообщение:

```
Mar  8 17:22:08 terminator named[22317]: starting BIND 9.1.0
```

Оно же для DNS-сервера BIND 8 выглядит так:

```
Mar  8 17:22:08 terminator named[22317]: restarted
```

И наконец, для сервера версии 4.9:

```
Mar  8 17:22:08 terminator named[22317]: starting
```

Если время перезапуска или перезагрузки не соотносится со временем внесения последних изменений, следует повторно перезагрузить DNS-сервер. Следует также убедиться, что при изменении файлов данных зоны был увеличен порядковый номер этой зоны. Если время редактирования файла данных зоны неизвестно, можно свериться с временем изменения файла, которое доступно по команде *ls -l*.

3. Вторичный сервер не может загрузить данные зоны

Если вторичный DNS-сервер не может получить текущий порядковый номер зоны от основного, в log-файл демона *syslog* попадает сообщение примерно следующего содержания (BIND 9):

```
Sep 25 22:02:38 wormhole named[21246]: refresh_callback: zone movie.edu/IN:
failure for 192.249.249.3#53: timed out
```

Либо в случае BIND 8:

```
Jan  6 11:55:25 wormhole named[544]: Err/TO getting serial# for "movie.edu"
```

И для BIND 4:

```
Mar  3 8:19:34 wormhole named[22261]: zoneref: Masters for secondary
       zone movie.edu unreachable
```

Если не обратить на проблему внимания, в конце концов произойдет устаревание зоны на вторичном сервере. Сообщение DNS-сервера BIND 9 в таком случае:

```
Sep 25 23:20:20 wormhole named[21246]: zone_expire: zone movie.edu/IN:
expired
```

Или в случае BIND 4.9 или BIND 8:

```
Mar  8 17:12:43 wormhole named[22261]: secondary zone
       "movie.edu" expired
```

Когда зона устаревает, при запросе у DNS-сервера данных из этой зоны возвращается ошибка SERVFAIL:

```
% nslookup robocop wormhole.movie.edu.
Server:  wormhole.movie.edu
Addresses:  192.249.249.1, 192.253.253.1

*** wormhole.movie.edu can't find robocop.movie.edu: Server failed
```

Существуют три основные причины для этой ошибки: потеря соединения с основным сервером из-за сбоя в сети, неправильный IP-адрес ос-

новного сервера в файле настройки или синтаксическая ошибка в файле данных зоны на первичном DNS-сервере. Прежде всего следует проверить запись для зоны в файле настройки и понять, какой IP-адрес используется вторичным сервером при попытке получить данные:

```
zone "movie.edu" {
                type slave;
                masters { 192.249.249.3; };
                file "bak.movie.edu";
};
```

Для серверов BIND 4 эта инструкция выглядит следующим образом:

```
secondary        movie.edu        192.249.249.3        bak.movie.edu
```

Убедитесь, что использованный адрес действительно является IP-адресом основного DNS-сервера. В случае совпадения адресов, проверьте наличие связи с этим IP-адресом:

```
% ping 192.249.249.3 -n 10
PING 192.249.249.3: 64 byte packets

----192.249.249.3 PING Statistics----
10 packets transmitted, 0 packets received, 100% packet loss
```

Если связь с основным DNS-сервером не может быть установлена, убедитесь, что узел, на котором запущен этот сервер, находится в рабочем состоянии (питание включено и т. д.), либо ищите проблему в работе сети. Если узел доступен, проверьте, что сервер *named* запущен на этом узле, и возможность получения зоны в пилотируемом режиме:

```
# /usr/sbin/named-xfer -z movie.edu -f /tmp/db.movie.edu -s 0 192.249.249.3
# echo $?
2
```

Код завершения 2 означает, что произошла ошибка. Обратитесь к log-файлу *syslog* на предмет появления в нем соответствующего сообщения. Для нашего примера было получено сообщение:

```
Jan  6 14:56:07 zardoz named-xfer[695]: record too short from
[192.249.249.3], zone movie.edu
```

На первый взгляд – похоже на ошибку усечения. Проблема на самом деле несколько меньше, что и можно увидеть, воспользовавшись программой *nslookup*:

```
% nslookup - terminator.movie.edu
Default Server:  terminator.movie.edu
Address:  192.249.249.3

> ls movie.edu      – Попытка получения зоны
 [terminator.movie.edu]
*** Can't list domain movie.edu: Query refused
```

Происходит следующее: *named* отказывается разрешить передачу зоны. Удаленный сервер обезопасил зональные данные предписанием *allow-transfer*, записью ресурсов *secure_zone* либо инструкцией *xfrnets* в загрузочном файле.

Если основной сервер отвечает, что не является авторитативным для зоны, DNS-сервер BIND 9 выдаст примерно следующее сообщение:

```
Sep 26 13:29:23 zardoz named[21890]: refresh_callback: zone movie.edu/IN:
non-authoritative answer from 192.249.249.3#53
```

DNS-сервер BIND 8:

```
Jan  6 11:58:36 zardoz named[544]: Err/TO getting serial# for "movie.edu"
Jan  6 11:58:36 zardoz named-xfer[793]: [192.249.249.3] not authoritative for
     movie.edu, SOA query got rcode 0, aa 0, ancount 0, aucount 0
```

Если мастер-сервер является правильным, то он *должен* быть авторитативным для зоны. В данном случае мастер-сервер, вероятно, испытывает проблемы с загрузкой данных зоны из-за синтаксической ошибки в файле данных. Свяжитесь с администратором основного сервера и попросите проверить log-файл *syslog* на предмет сообщений о синтаксических ошибках (см. проблему за номером 5, которую мы скоро изучим).

4. К файлу данных зоны добавлено имя, но не создана соответствующая PTR–запись

Поскольку в DNS отображение имен узлов в IP-адреса отделено от преобразования IP-адресов в имена узлов, очень легко забыть добавить PTR-запись для нового узла. Добавление A-записи – действие, производимое почти рефлекторно, но многие люди, привыкшие к использованию таблиц узлов, предполагают, что добавление адресной записи решает и проблему обратного отображения. Но это не так – следует добавлять PTR-запись для нового узла в соответствующую зону обратного отображения.

Отсутствие PTR-записи для адреса узла обычно приводит к невозможности производить идентификацию узла. Так, пользователи не смогут вызвать программу *rlogin* без указания пароля для доступа к другим узлам, а *rsh* и *rcp* просто не будут работать. Серверы, с которыми общаются эти программы, должны иметь возможность преобразовать IP-адрес клиента в доменное имя, чтобы произвести проверку по файлам *.rhosts* и *hosts.equiv*. Попытки соединений для этих пользователей будут приводить к появлению в log-файле демона *syslog* примерно таких записей:

```
Aug 15 17:32:36 terminator inetd[23194]: login/tcp:
       Connection from unknown (192.249.249.23)
```

Помимо этого, многие крупные FTP-архивы, включая *ftp.uu.net*, отказывают в анонимном доступе узлам, IP-адреса которых не могут быть отображены в доменные имена. Сервер *ftp.uu.net* выдает сообщение, фрагмент которого приводится ниже:

```
530- Sorry, we're unable to map your IP address 140.186.66.1 to a hostname
530- in the DNS.  This is probably because your nameserver does not have a
530- PTR record for your address in its tables, or because your reverse
530- nameservers are not registered.  We refuse service to hosts whose
530- names we cannot resolve.
```

В этом случае причина запрещения использовать анонимный FTP-доступ довольно очевидна. Однако другие FTP-серверы не утруждаются отображением информативных сообщений, а просто отказывают в предоставлении доступа к службе.

Чтобы проверить, создана ли PTR-запись, полезна программа *nslookup*:

```
% nslookup
Default Server:  terminator.movie.edu
Address:  192.249.249.3

> beetlejuice        — Проверка отображения имени в адрес
Server:  terminator.movie.edu
Address:  192.249.249.3

Name:    beetlejuice.movie.edu
Address:  192.249.249.23

> 192.249.249.23    — Проверка соответствующего обратного отображения
Server:  terminator.movie.edu
Address:  192.249.249.3

*** terminator.movie.edu can't find 192.249.249.23: Non-existent domain
```

На первичном DNS-мастер-сервере зоны *249.249.192.in-addr.arpa* быстрое изучение файла *db.192.249.249* позволяет понять, что PTR-запись не была добавлена к файлу данных зоны, или что DNS-сервер просто не был перезагружен. Если DNS-сервер, о котором идет речь, является вторичным для зоны, убедитесь, что порядковый номер был увеличен на первичном DNS-мастер-сервере и что у вторичного сервера было достаточно времени для загрузки новой зоны.

5. Ошибка синтаксиса в файле настройки или файле данных зоны

Ошибки синтаксиса довольно часто (частота зависит, в основном, от опыта администратора) встречаются в файлах настройки DNS-сервера и файлах данных зон. Обычно ошибка в файле настройки приводит к тому, что DNS-сервер не может загрузить одну или несколько зон. Некоторые опечатки в операторе *options* могут приводить к тому, что

DNS-сервер просто не запустится, записав подобное сообщение в log-файл демона *syslog* (BIND 9):

```
Sep 26 13:39:30 terminator named[21924]: change directory to '/var/name'
failed: file not found
Sep 26 13:39:30 terminator named[21924]: options configuration failed: file
not found
Sep 26 13:39:30 terminator named[21924]: loading configuration: failure
Sep 26 13:39:30 terminator named[21924]: exiting (due to fatal error)
```

Для DNS-сервера BIND 8:

```
Jan  6 11:59:29 terminator named[544]: can't change directory to /var/name: No
     such file or directory
```

Следует помнить, что при запуске *named* из командной строки или при загрузке системы, сообщения об ошибках не отображаются, но *named* работает очень недолго.

Если ошибка синтаксиса присутствует в менее важной строке файла настройки – скажем, в операторе *zone* – проблемы будут возникать только с этой зоной. Как правило, DNS-сервер просто не будет способен загрузить зону (скажем, неправильно написано слово «masters» или имя файла данных зоны, либо отсутствуют кавычки при указании имени файла или доменного имени). Для сервера BIND 9 это приведет примерно к следующему выводу в log-файле *syslog*:

```
Sep 26 13:43:03 terminator named[21938]: /etc/named.conf:80: parse error near
'masters'
Sep 26 13:43:03 terminator named[21938]: loading configuration: failure
Sep 26 13:43:03 terminator named[21938]: exiting (due to fatal error)
```

Для BIND 8:

```
Jan  6 12:01:36 terminator named[841]: /etc/named.conf:10: syntax error near
     'movie.edu'
```

Если ошибка синтаксиса присутствует в файле данных зоны, но DNS-сервер успешно загружает зону, это приведет либо к отправке неавторитативных ответов для *всех* данных зоны, либо к возврату ошибки SERVFAIL на запросы данных из зоны:

```
% nslookup carrie
Server:  terminator.movie.edu
Address:  192.249.249.3

Non-authoritative answer:
Name:    carrie.movie.edu
Address:  192.253.253.4
```

Вот *syslog*-сообщения DNS-сервера BIND 9, полученные в результате синтаксической ошибки, приведшей к подобной проблеме:

```
Sep 26 13:45:40 terminator named[21951]: error: dns_rdata_fromtext:
db.movie.edu:11: near 'postmanrings2x': unexpected token
```

```
Sep 26 13:45:40 terminator named[21951]: error: dns_zone_load: zone
movie.edu/IN: database db.movie.edu: dns_db_load failed: unexpected token
Sep 26 13:45:40 terminator named[21951]: critical: loading zones: unexpected
token
Sep 26 13:45:40 terminator named[21951]: critical: exiting (due to fatal
error)
```

Ошибки для BIND 8:

```
Jan  6 15:07:46 terminator named[693]: db.movie.edu:11: Priority error
     (postmanrings2x.movie.edu.)
Jan  6 15:07:46 terminator named[693]: master zone "movie.edu" (IN) rejected
due
     to errors (serial 1997010600)
```

Взглянув на файл данных зоны, являющийся источником проблемы, мы увидим вот такую запись:

```
postmanrings2x     IN     MX     postmanrings2x.movie.edu.
```

В MX-записи отсутствует приоритет, это и является причиной проблем.

Следует помнить, что синтаксическую ошибку можно не заметить, если не читать очень внимательно log-файл *syslog*, либо не соотносить проблему с тем фактом, что сервер возвращает неавторитативный ответ (хотя должен быть авторитативным для зоны).

Начиная с BIND 4.9.4, «неправильное» имя узла может являться синтаксической ошибкой:

```
Jan  6 12:04:10 terminator named[841]: owner name "ID_4.movie.edu" IN
(primary)
     is invalid - rejecting
Jan  6 12:04:10 terminator named[841]: db.movie.edu:11: owner name error
Jan  6 12:04:10 terminator named[841]: db.movie.edu:11: Database error near (A)
Jan  6 12:04:10 terminator named[841]: master zone "movie.edu" (IN) rejected
     due to errors (serial 1997010600)
```

В BIND 9 (в версии 9.1.0) проверка имен не реализована, хотя в будущих версиях может вновь появиться.

6. Отсутствует точка в конце доменного имени в файле данных зоны

Невероятно легко забыть последнюю точку в имени при редактировании файла данных зоны. Поскольку правила применения последней точки так часто меняются (не использовать в файле настройки, не использовать в файле *resolv.conf*, напротив – использовать в файлах данных зоны для замещения значения $ORIGIN...), не всегда удается за ними уследить. Следующие RR-записи:

```
zorba         IN     MX     10 zelig.movie.edu
movie.edu     IN     NS     terminator.movie.edu
```

выглядят обычно для неопытного глаза, но они, скорее всего, не работают так, как подразумевалось. В файле *db.movie.edu* они были бы эквивалентны следующим записям:

```
zorba.movie.edu.        IN    MX    10 zelig.movie.edu.movie.edu.
movie.edu.movie.edu.    IN    NS    terminator.movie.edu.movie.edu.
```

если суффикс по умолчанию не был явным образом изменен.

Если в разделе данных RR-записи забыть точку в конце данных RR-записи (а это не то же самое, что забыть точку в конце *доменного имени* RR-записи), это обычно приводит к появлению разных идиотских NS- или MX-записей:

```
% nslookup -type=mx zorba.movie.edu.
Server:  terminator.movie.edu
Address:  192.249.249.3

zorba.movie.edu      preference = 10, mail exchanger
                     = zelig.movie.edu.movie.edu
zorba.movie.edu      preference = 50, mail exchanger
                     = postmanrings2x.movie.edu.movie.edu
```

Причина поведения должна быть вполне понятна из вывода *nslookup*. Но если забыть последнюю точку в доменном имени из правой части записи (как в только что показанной NS-записи для *movie.edu*), то найти эту ошибку будет непросто. Если попытаться найти эту запись с помощью *nslookup*, она не будет найдена для искомого доменного имени. Дамп базы данных DNS-сервера может помочь:

```
$ORIGIN edu.movie.edu.
movie    IN    NS    terminator.movie.edu.movie.edu.
```

Строка $ORIGIN выглядит достаточно странно, чтобы обратить на нее внимание.

7. Отсутствуют данные корневых указателей

Если по какой-либо причине не был создан файл корневых указателей для DNS-сервера либо этот файл был случайно удален, DNS-сервер не сможет производить разрешение имен, которые выходят за пределы его авторитативности. Такое поведение легко обнаружить с помощью *nslookup*, но при этом следует указывать абсолютные доменные имена, иначе использование списка поиска может привести к ложным результатам:

```
% nslookup
Default Server:  terminator.movie.edu
Address:  192.249.249.3

> ftp.uu.net.     – Поиск имени за пределами авторитативности
                    DNS-сервера приводит к ошибке SERVFAIL...
Server:  terminator.movie.edu
```

```
Address:  192.249.249.3

*** terminator.movie.edu can't find ftp.uu.net.: Server failed
```

Поиск имени в пределах авторитативности DNS-сервера приводит к получению результата:

```
> wormhole.movie.edu.
Server:  terminator.movie.edu
Address:  192.249.249.3

Name:    wormhole.movie.edu
Addresses:  192.249.249.1, 192.253.253.1

> ^D
```

Чтобы подтвердить подозрение, что отсутствуют данные корневых указателей, сверьтесь с log-файлом *syslog* на предмет наличия следующей ошибки:

```
Jan  6 15:10:22 terminator named[764]: No root nameservers for class IN
```

Напомним, что класс 1 – это класс IN, или Интернет. Ошибка показывает, что по причине отсутствия данных корневых указателей корневые DNS-серверы не могут быть найдены.

Маловероятно, что эта проблема проявится при использовании BIND 9, поскольку в этой ветви пакета реализованы встроенные указатели корневых серверов.

8. Отсутствие сетевого соединения

Хотя Интернет сегодня стал гораздо более надежной системой, чем во времена освоения Дикого Запада и существования ARPAnet, но выход из строя сетей все еще встречается достаточно часто. Если не заглядывать «под капот» на предмет изучения отладочного вывода, то эти поломки обычно похожи на падение производительности:

```
% nslookup nisc.sri.com.
Server:  terminator.movie.edu
Address:  192.249.249.3

*** Request to terminator.movie.edu timed out ***
```

Если включить отладку на DNS-сервере, то можно увидеть, что он вполне здоров. Он получил запрос от клиента, послал необходимые запросы и терпеливо ждал ответы. Но ни одного не получил. Вот так может выглядеть в этом случае отладочный вывод сервера BIND 8:

```
Debug turned ON, Level 1
```

nslookup посылает первый запрос локальному DNS-серверу на предмет получения IP-адреса для имени *nisc.sri.com*. Запрос был передан другому DNS-серверу, а затем еще одному – из-за отсутствия ответа от первого:

```
datagram from [192.249.249.3].1051, fd 5, len 30
req: nlookup(nisc.sri.com) id 18470 type=1 class=1
req: missed 'nisc.sri.com' as 'com' (cname=0)
forw: forw -> [198.41.0.4].53 ds=7 nsid=58732 id=18470 0ms retry 4 sec
resend(addr=1 n=0) -> [128.9.0.107].53 ds=7 nsid=58732 id=18470 0ms
```

nslookup теряет терпение и повторяет запрос к локальному DNS-серверу. Обратите внимание, что используется тот же исходный порт. Локальный DNS-сервер игнорирует дублирующийся запрос и пробует ретранслировать первый запрос еще два раза:

```
datagram from [192.249.249.3].1051, fd 5, len 30
req: nlookup(nisc.sri.com) id 18470 type=1 class=1
req: missed 'nisc.sri.com' as 'com' (cname=0)
resend(addr=2 n=0) -> [192.33.4.12].53 ds=7 nsid=58732 id=18470 0ms
resend(addr=3 n=0) -> [128.8.10.90].53 ds=7 nsid=58732 id=18470 0ms
```

nslookup снова посылает запрос локальному DNS-серверу, и сервер продолжает посылать новые запросы:

```
datagram from [192.249.249.3].1051, fd 5, len 30
req: nlookup(nisc.sri.com) id 18470 type=1 class=1
req: missed 'nisc.sri.com' as 'com' (cname=0)
resend(addr=4 n=0) -> [192.203.230.10].53 ds=7 nsid=58732 id=18470 0ms
resend(addr=0 n=1) -> [198.41.0.4].53 ds=7 nsid=58732 id=18470 0ms
resend(addr=1 n=1) -> [128.9.0.107].53 ds=7 nsid=58732 id=18470 0ms
resend(addr=2 n=1) -> [192.33.4.12].53 ds=7 nsid=58732 id=18470 0ms
resend(addr=3 n=1) -> [128.8.10.90].53 ds=7 nsid=58732 id=18470 0ms
resend(addr=4 n=1) -> [192.203.230.10].53 ds=7 nsid=58732 id=18470 0ms
resend(addr=0 n=2) -> [198.41.0.4].53 ds=7 nsid=58732 id=18470 0ms
Debug turned OFF
```

В случае DNS-сервера BIND 9 на первом уровне отладки значительно меньше подробностей. Тем не менее можно видеть, что DNS-сервер постоянно пытается произвести поиск для имени *nisc.sri.com*:

```
Sep 26 14:33:27.486 client 192.249.249.3#1028: query: nisc.sri.com A
Sep 26 14:33:27.486 createfetch: nisc.sri.com. A
Sep 26 14:33:32.489 client 192.249.249.3#1028: query: nisc.sri.com A
Sep 26 14:33:32.490 createfetch: nisc.sri.com. A
Sep 26 14:33:42.500 client 192.249.249.3#1028: query: nisc.sri.com A
Sep 26 14:33:42.500 createfetch: nisc.sri.com. A
Sep 26 14:34:02.512 client 192.249.249.3#1028: query: nisc.sri.com A
Sep 26 14:34:02.512 createfetch: nisc.sri.com. A
```

На более высоких уровнях отладки можно видеть интервалы ожидания, но в BIND 9.1.0 по-прежнему не отображаются адреса удаленных DNS-серверов, которым посылаются запросы.

Из отладочного вывода DNS-сервера BIND 8 можно извлечь перечень IP-адресов удаленных DNS-серверов и проверить существование маршрутов до этих серверов. Очень вероятно, что программе *ping* не повезет точно так же, как и DNS-серверу:

```
% ping 198.41.0.4 -n 10          – ping для первого из DNS-серверов
PING 198.41.0.4: 64 byte packets

----198.41.0.4 PING Statistics----
10 packets transmitted, 0 packets received, 100% packet loss
% ping 128.9.0.107 -n 10         – ping для второго из DNS-серверов
PING 128.9.0.107: 64 byte packets

----128.9.0.107 PING Statistics----
10 packets transmitted, 0 packets received, 100% packet loss
```

Если же маршрут существует, следует убедиться, что удаленные серверы запущены и работают. Можно также уточнить, не блокирует ли местный интернет-брандмауэр запросы DNS-сервера. Если недавно был осуществлен переход с BIND 8 на BIND 9, обратитесь к врезке «Хитрости совместного использования BIND 8/9 и брандмауэров с фильтрацией пакетов» в главе 11 «Безопасность»; вполне возможно, что информация окажется полезной.

Если же *ping* не в состоянии достучаться до серверов по указанным адресам, остается только заняться поисками поломки в сети. Для локализации проблемы (в местной сети, конечной сети, либо на пути между ними) может быть полезна программа *traceroute* и использование *ping* по кратчайшему маршруту.

Помимо этого для поиска проблемы следует применять здравый смысл. В приведенном примере все удаленные DNS-серверы являются корневыми. (Их PTR-записи могли быть где-то кэшированы, так что можно узнать и доменные имена.) Маловероятно, что каждая из сетей, содержащих корневые DNS-серверы, вышла из строя, как и то, что магистральные сети Интернет в одночасье просто развалились. Принцип бритвы Оккама подсказывает, что наиболее простая из причин, которые могли привести к такому поведению программ, – а именно разрыв подключения местной сети к сети Интернет – и является действительной причиной.

9. Отсутствие делегирования для поддоменов

Регистраторы прилагают все усилия для скорейшей обработки запросов, но может пройти день или два, прежде чем информация о делегировании вашего поддомена появится на DNS-серверах родительской зоны. Если родительская зона – не один из родовых доменов высшего уровня, время может меняться. Некоторые родители действуют оперативно и ответственно, другие медленно и не всегда правильно. Но как и в реальной жизни – родителей не заменишь на других.

Пока делегирование вашей зоны не объявится на DNS-серверах родительской, ваши DNS-серверы смогут получать данные из пространства имен сети Интернет, но никто во всей сети Интернет (не считая локального домена) не будет знать, как искать данные в *вашем* сегменте пространства имен.

Это означает, что можно посылать почтовые сообщения за пределы домена, но получатели не смогут ответить на эти сообщения. Более того, никто не сможет получить *telnet*, *ftp* и даже *ping*-доступ к узлам вашего домена по их именам.

Помните, это справедливо и для любых зон *in-addr.arpa*, находящихся в вашем ведении. Пока родительские зоны не делегируют их вашим DNS-серверам, DNS-серверы сети Интернет не смогут производить обратное отображение для адресов вашей сети.

Чтобы выяснить, добралась ли информация о делегировании до DNS-серверов родительской зоны, следует запросить у одного из них NS-записи для локальной зоны. Если данные существуют на родительском DNS-сервере, значит они доступны всей сети Интернет:

```
% nslookup
Default Server:  terminator.movie.edu
Address:  192.249.249.3

> server a.root-servers.net.      – Запрос к корневому DNS-серверу
Default Server:  a.root-servers.net
Address:  198.41.0.4

> set norecurse                   – Предписать серверу искать ответ только
                                    в локальных данных
> set type=ns                       – и только NS-записи
> 249.249.192.in-addr.arpa.         – для 249.249.192.in-addr.arpa
Server:  a.root-servers.net
Address:  198.41.0.4

*** a.root-servers.net can't find 249.249.192.in-addr.arpa.: Non-existent
domain
```

В данном случае очевидно, что информации о делегировании еще нет. Можно терпеливо ждать либо, если с момента запроса делегирования прошло уже неприлично много времени, связаться с администратором родительской зоны и спросить, в чем, собственно, дело.

10. Некорректное делегирование поддомена

Еще одна распространенная в сети Интернет проблема – некорректное делегирование поддоменов. Обновление информации о делегировании требует вмешательства человека – необходимо сообщать администратору родительской зоны об изменениях в наборе авторитативных DNS-серверов. Как следствие, информация о делегировании часто становится неправильной, если администраторы вносят изменения, не уведомляя родительскую зону. Слишком многие администраторы полагают, что делегирование – единичное действие и что рассказав родителям единожды, при создании зоны, какие серверы являются авторитативными, они могут больше никогда с родителями не общаться и даже не звонить в День матери.

Администратор может создать новый DNS-сервер, демобилизовать один из прежних, изменить IP-адрес третьего, ничего не говоря администратору родительской зоны. Постепенно число DNS-серверов с корректным делегированием сходит на нет. В лучшем случае это приводит к увеличению времени, уходящего на разрешение, поскольку удаленные DNS-серверы с трудом пытаются найти хотя бы один авторитативный сервер для зоны. Если данные делегирования окончательно устаревают, а последний авторитативный DNS-сервер перестает работать по причине выполнения профилактических работ, информация из сегмента пространства имен, начинающегося с текущей зоны, становится абсолютно недоступной.

Если администратор подозревает некорректное делегирование со стороны родительской зоны либо со стороны текущей зоны по отношению к одному из потомков, либо со стороны удаленной зоны по отношению к одному из ее потомков, то может произвести диагностику с помощью *nslookup*:

```
% nslookup
Default Server:  terminator.movie.edu
Address:  192.249.249.3

> server a.root-servers.net.   – использовать DNS-сервер родительской
                               – зоны, на который падает подозрение
                               – в некорректном делегировании
Default Server:  a.root-servers.net
Address:  198.41.0.4

> set type=ns                  – искать NS-записи
> hp.com.                      – для зоны, о которой идет речь
Server:  a.root-servers.net
Address:  198.41.0.4

Non-authoritative answer:
hp.com          nameserver = RELAY.HP.COM
hp.com          nameserver = HPLABS.HPL.HP.COM
hp.com          nameserver = NNSC.NSF.NET
hp.com          nameserver = HPSDLO.SDD.HP.COM

Authoritative answers can be found from:
hp.com          nameserver = RELAY.HP.COM
hp.com          nameserver = HPLABS.HPL.HP.COM
hp.com          nameserver = NNSC.NSF.NET
hp.com          nameserver = HPSDLO.SDD.HP.COM
RELAY.HP.COM    internet address = 15.255.152.2
HPLABS.HPL.HP.COM       internet address = 15.255.176.47
NNSC.NSF.NET    internet address = 128.89.1.178
HPSDLO.SDD.HP.COM       internet address = 15.255.160.64
HPSDLO.SDD.HP.COM       internet address = 15.26.112.11
```

Предположим, мы заподозрили, что делегирование *hpsdlo.sdd.hp.com* некорректно. Запрашиваем у *hpsdlo.sdd.hp.com* данные из зоны *hp.com* (скажем, SOA-запись для *hp.com*) и смотрим на ответ:

```
> server hpsdlo.sdd.hp.com.
Default Server:  hpsdlo.sdd.hp.com
Addresses:  15.255.160.64, 15.26.112.11

> set norecurse
> set type=soa
> hp.com.
Server:  hpsdlo.sdd.hp.com
Addresses:  15.255.160.64, 15.26.112.11

Non-authoritative answer:
hp.com
        origin = relay.hp.com
        mail addr = hostmaster.hp.com
        serial = 1001462
        refresh = 21600 (6 hours)
        retry   = 3600 (1 hour)
        expire  = 604800 (7 days)
        minimum ttl = 86400 (1 day)

Authoritative answers can be found from:
hp.com          nameserver = RELAY.HP.COM
hp.com          nameserver = HPLABS.HPL.HP.COM
hp.com          nameserver = NNSC.NSF.NET
RELAY.HP.COM    internet address = 15.255.152.2
HPLABS.HPL.HP.COM       internet address = 15.255.176.47
NNSC.NSF.NET    internet address = 128.89.1.178
```

Если бы узел *hpsdlo.sdd.hp.com* был действительно авторитативным для зоны *hp.com*, то вернул бы авторитативный ответ. Администратор зоны *hp.com* может уточнить, должен ли сервер *hpsdlo.sdd.hp.com* быть авторитативным для *hp.com*, так что именно с ним следует связаться.

Еще один распространенный симптом некорректного делегирования: сообщение об ошибке «lame server».

```
Oct 1 04:43:38 terminator named[146]: Lame server on '40.234.23.210.in-
addr.arpa' (in '210.in-addr.arpa'?): [198.41.0.5].53 'RSO.INTERNIC.NET':
learnt(A=198.41.0.21,NS=128.63.2.53)
```

Понимать ошибку следует так: ваш DNS-сервер был направлен сервером с адресом 128.63.2.53 к DNS-серверу с адресом 198.41.0.5 на предмет получения адреса для имени из домена *210.in-addr.arpa*, а именно – *40.234.23.210.in-addr.arpa*. Ответ от DNS-сервера с адресом 198.41.0.5 показывает, что этот сервер в действительности не является авторитативным для зоны *210.in-addr.arpa*. Поэтому либо сервер по адресу 128.63.2.53 вернул некорректную информацию о делегировании, либо сервер с адресом 198.41.0.5 неправильно настроен.

11. Ошибки синтаксиса в файле resolv.conf

Несмотря на примитивный синтаксис содержимого файла *resolv.conf*, люди время от времени делают ошибки при его редактировании. И, к сожалению, строки с ошибками в *resolv.conf* молча игнорируются клиентом. В результате некоторые элементы настройки могут не работать: используется неправильное локальное доменное имя либо список поиска, клиент не посылает запросы одному из перечисленных DNS-серверов. Команды, выполнение которых связано со списком поиска, не работают, клиент не посылает запрос нужному DNS-серверу либо вовсе не посылает запросы.

Самый простой способ проверить, работают ли настройки в *resolv.conf* как ожидалось, – выполнить *nslookup*. *nslookup* любезно сообщит об используемом локальном доменном имени и списке поиска, полученном из *resolv.conf*, а также об используемом DNS-сервере – при выполнении команды *set all*, о которой мы уже рассказывали в главе 12 «nslookup и dig»:

```
% nslookup
Default Server:  terminator.movie.edu
Address:  192.249.249.3

> set all
Default Server:  terminator.movie.edu
Address:  192.249.249.3

Set options:
  nodebug         defname         search         recurse
  nod2            novc            noignoretc     port=53
  querytype=A     class=IN        timeout=5      retry=4
  root=ns.nic.ddn.mil.
  domain=movie.edu
  srchlist=movie.edu

>
```

Вывод команды *set all* должен соответствовать ожиданиям, связанным с настройками в файле *resolv.conf*. К примеру, если в файле *resolv.conf* присутствует строка *search fx.movie.edu movie.edu*, вывод должен содержать следующие строки:

```
domain=fx.movie.edu
srchlist=fx.movie.edu/movie.edu
```

В противном случае следует внимательно изучить файл *resolv.conf*. Если очевидных ошибок нет, следует произвести поиск неотображаемых символов (скажем, с помощью команды *set list* редактора *vi*). Следует обращать особое внимание на возможное присутствие пробелов в конце строки; в более старых клиентах пробелы после доменного имени включались в имя. Естественно, имена существующих доменов высшего уровня никогда не заканчиваются пробелами, и такое включение

приводит к невозможности производить поиск для всех имен, которые не заканчиваются точкой.

12. Не определено локальное доменное имя

Еще одна распространенная оплошность. Локальное доменное имя можно установить явным образом с помощью *hostname* (в абсолютное доменное имя узла) либо в файле *resolv.conf*. Симптомы в этом случае довольно простые – использование имен из одной метки либо сокращенных доменных имен в командах не особенно эффективно:

```
% telnet br
br: No address associated with name
% telnet br.fx
br.fx: No address associated with name
% telnet br.fx.movie.edu
Trying...
Connected to bladerunner.fx.movie.edu.
Escape character is '^]'.

HP-UX bladerunner.fx.movie.edu A.08.07 A 9000/730 (ttys1)
login:
```

Чтобы продиагностировать проблему, можно использовать *nslookup*, примерно как в предыдущем пункте с *resolv.conf*:

```
% nslookup
Default Server:  terminator.movie.edu
Address:  192.249.249.3

> set all
Default Server:  terminator.movie.edu
Address:  192.249.249.3

Set options:
  nodebug         defname         search          recurse
  nod2            novc            noignoretc      port=53
  querytype=A     class=IN        timeout=5       retry=4
  root=ns.nic.ddn.mil.
  domain=
  srchlist=
```

Обратите внимание, что список поиска также отсутствует. В качестве варианта можно отследить эту проблему, включив отладку в DNS-сервере. (Разумеется, это требует доступа к DNS-серверу, который может работать совсем не на узле, для которого диагностируется проблема.) Ниже приводится фрагмент отладочного вывода DNS-сервера BIND 9 после выполнения вышеупомянутых команд *telnet*:

```
Sep 26 16:17:58.824 client 192.249.249.3#1032: query: br A
Sep 26 16:17:58.825 createfetch: br. A
Sep 26 16:18:09.996 client 192.249.249.3#1032: query: br.fx A
```

```
Sep 26 16:18:09.996 createfetch: br.fx. A
Sep 26 16:18:18.677 client 192.249.249.3#1032: query: br.fx.movie.edu A
```

Для DNS-сервера BIND 8 фрагмент выглядит примерно следующим образом:

```
Debug turned ON, Level 1

datagram from [192.249.249.3].1057, fd 5, len 20
req: nlookup(br) id 27974 type=1 class=1
req: missed 'br' as '' (cname=0)
forw: forw -> [198.41.0.4].53 ds=7 nsid=61691 id=27974 0ms retry 4 sec

datagram from [198.41.0.4].53, fd 5, len 20
ncache: dname br, type 1, class 1
send_msg -> [192.249.249.3].1057 (UDP 5) id=27974

datagram from [192.249.249.3].1059, fd 5, len 23
req: nlookup(br.fx) id 27975 type=1 class=1
req: missed 'br.fx' as '' (cname=0)
forw: forw -> [128.9.0.107].53 ds=7 nsid=61692 id=27975 0ms retry 4 sec

datagram from [128.9.0.107].53, fd 5, len 23
ncache: dname br.fx, type 1, class 1
send_msg -> [192.249.249.3].1059 (UDP 5) id=27975

datagram from [192.249.249.3].1060, fd 5, len 33
req: nlookup(br.fx.movie.edu) id 27976 type=1 class=1
req: found 'br.fx.movie.edu' as 'br.fx.movie.edu' (cname=0)
req: nlookup(bladerunner.fx.movie.edu) id 27976 type=1 class=1
req: found 'bladerunner.fx.movie.edu' as 'bladerunner.fx.movie.edu'
     (cname=1)
ns_req: answer -> [192.249.249.3].1060 fd=5 id=27976 size=183 Local
Debug turned OFF
```

Теперь сравните это с отладочным выводом, полученным при использовании списка поиска в главе 13. В данном случае поиск производится только для тех имен, которые набрал пользователь, никакие доменные имена не добавляются автоматически. Очевидно, список поиска не используется.

13. Ответ получен от неизвестного источника

Одна из проблем, с которой все чаще обращаются в конференции по DNS, связана с сообщением «response from unexpected source». Некогда такие ответы были названы «пришельцами»: они поступают с IP-адреса, который не совпадает с адресом сервера, которому был послан запрос. Когда сервер BIND посылает запрос удаленному серверу, то добросовестно проверяет, что полученные ответы исходят от IP-адреса этого удаленного сервера. Это позволяет минимизировать возможность получения поддельных ответов. BIND столь же требователен и к себе: DNS-сервер BIND всеми силами старается ответить через тот же сетевой интерфейс, через который был получен запрос.

Вот сообщение об ошибке, которое вероятнее всего будет создано при получении (возможно) непрошенного ответа:

```
Mar  8 17:21:04 terminator named[235]: Response from unexpected source ([205.
199.4.131].53)
```

Возможных причин появления такого сообщения две: кто-то пытается произвести spoof-атаку на DNS-сервер, либо – и это более вероятно – запрос был отправлен DNS-серверу более старой версии или другой модели, и адресат не очень заботится о том, чтобы посылать ответы через тот же сетевой интерфейс, через который поступают запросы.

Проблемы перехода на новую версию

С выпуском BIND версий 8 и 9 во многих Unix-системах началось обновление DNS-клиентов и DNS-серверов. Однако некоторые особенности самых последних версий BIND могут показаться вам ошибками. Мы попытаемся дать читателям представление об изменениях в DNS-сервере и самой DNS, происходящих при таком переходе.

Поведение клиента

Изменения, связанные со списком поиска по умолчанию, описанные в главе 6, могут оказаться настоящей проблемой для пользователей. Вспомним, что при локальном доменном имени *fx.movie.edu* список поиска по умолчанию уже не будет содержать имени *movie.edu*. Следовательно, пользователи, привыкшие выполнять команды вроде *telnet db.personnel* и получать автоматическое дополнение частичного имени до *db.personnel.movie.edu*, обнаружат, что их команды перестали работать. Чтобы решить эту проблему, можно воспользоваться инструкцией *search* и определить список поиска явным образом, включив в него имя родителя для локального доменного имени. Как вариант можно сообщить пользователям об изменении в поведении клиента.

Поведение DNS-сервера

До версии 4.9 DNS-сервер BIND с радостью загружал данные любой зоны из произвольного файла данных зоны, для которой сервер был первичным мастером. Если сервер был настроен в качестве первичного мастера для *movie.edu* и ему было сказано, что данные для *movie.edu* находятся в файле *db.movie.edu*, то после этого можно добавить данные для *hp.com* в файл *db.movie.edu*, и DNS-сервер загрузит RR-записи *hp.com* в кэш. В некоторых книгах даже предлагалось собирать данные для всех зон *in-addr.arpa* в один файл. Ох.

Все DNS-серверы BIND версии 4.9 и более поздних игнорируют «внезональные» RR-записи в файле данных зоны. Поэтому если затолкать все PTR-записи для всех зон *in-addr.arpa* в один файл и загрузить этот

файл единственным оператором *zone* или инструкцией *primary*, DNS-сервер проигнорирует все записи, не принадлежащие указанной зоне. Результат ясен – отсутствие практически всех PTR-записей и вызовы *gethostbyaddr( )*, не возвращающие результатов.

BIND комментирует факт игнорирования подобных записей в log-файле *syslog*. В BIND 9 сообщения выглядят следующим образом:

```
Sep 26 13:48:19 terminator named[21960]: dns_master_load: db.movie.edu:16:
ignoring out-of-zone data
```

А вот так – в BIND 8:

```
Jan  7 13:58:01 terminator named[231]: db.movie.edu:16: data "hp.com" outside zone
     "movie.edu" (ignored)
Jan  7 13:58:01 terminator named[231]: db.movie.edu:17: data "hp.com" outside zone
     "movie.edu" (ignored)
```

Решение: для каждой зоны использовать отдельный файл данных и один оператор *zone* либо инструкцию *primary* на одну зону.

Проблемы сосуществования и версий

С переходом на BIND 9 и появлением сервера Microsoft DNS появляется все больше проблем, связанных с взаимодействием DNS-серверов. Существуют также проблемы, присущие той или иной версии BIND либо используемой операционной системе. Многие из этих проблем легко поддаются диагностированию и разрешению, и мы проявили бы небрежность, не рассказав о них.

Передача зоны не может быть произведена из-за фирменной WINS-записи

Если сервер Microsoft DNS настроен на использование WINS-сервера для поиска информации по именам, не найденным в указанной зоне, происходит добавление специальной записи в файл данных зоны. Запись выглядит следующим образом:

```
@   IN   WINS   &IP-адрес сервера WINS
```

К несчастью, WINS не является стандартным типом записи класса IN. Следовательно, когда вторичные серверы BIND пытаются получить зону, они запинаются на WINS-записи, отказываясь производить загрузку:

```
May 23 15:58:43 terminator named-xfer[386]: "fx.movie.edu IN 65281" - unknown
type (65281)
```

Можно настроить сервер Microsoft DNS на удаление фирменной записи перед передачей зоны. В левой части окна DNS Manager следует вы-

брать имя зоны, нажать правую кнопку мыши, и выбрать пункт *Properties*. В полученном окне *Zone Properties* следует перейти на закладку *WINS Lookup* (рис. 14.1).

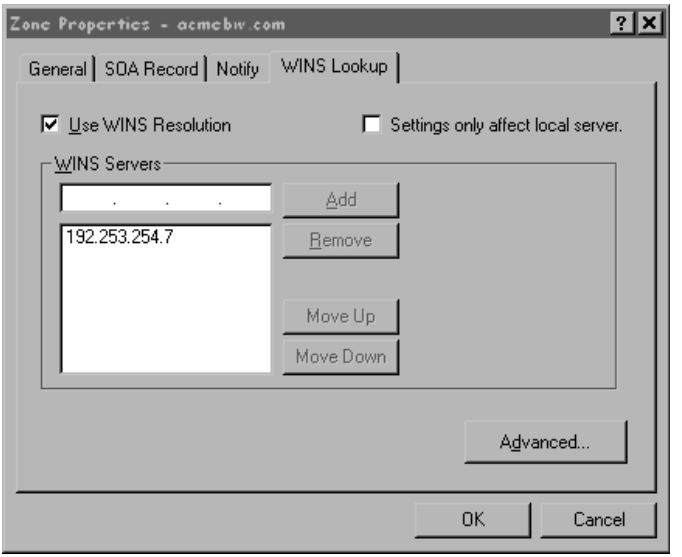

Рис. 14.1. *Окно свойств зоны*

Установка флажка *Settings only affect local server* приведет к фильтрации WINS-записи для текущей зоны. Но если присутствуют вторичные серверы Microsoft DNS, они также не получат эту запись, хотя и могли бы ее использовать.

DNS-сервер сообщает «No NS Record for SOA MNAME»

Эта ошибка существует только в серверах BIND 8.1:

```
May 8 03:44:38 terminator named[11680]: no NS RR for SOA MNAME "movie.edu" in
zone "movie.edu"
```

Сервер версии 8.1 был настоящим фанатом первого поля SOA-записи. Помните это поле? В главе 4 «Установка BIND» мы говорили, что по традиции это поле содержит доменное имя первичного DNS-мастер-сервера зоны. BIND 8.1, полагаясь на это, проверяет наличие соответствующей NS-записи, связывающей доменное имя зоны с сервером в поле MNAME. Если такой NS-записи не существует, BIND создает приведенное сообщение об ошибке. Также перестают корректно работать сообщения NOTIFY. Можно вписать в поле MNAME доменное имя DNS-сервера, для которого существует NS-запись, либо обновить BIND до более новой версии BIND 8. Рекомендуется обновление, по-

скольку BIND 8.1 слишком стар. Проверка была удалена из кода уже в версии 8.1.1.

DNS-сервер сообщает «Too Many Open Files»

На узлах с большим числом IP-адресов или строгим ограничением на количество открытых пользователем файлов, BIND может выдавать сообщение:

```
Dec 12 11:52:06 terminator named[7770]: socket(SOCK_RAW): Too many open files
```

и аварийно завершать работу.

Поскольку BIND пытается выполнить *bind( )* и производить прослушивание для каждого сетевого интерфейса узла, могут просто кончиться файловые дескрипторы. Это особенно часто встречается на узлах с многочисленными виртуальными интерфейсами, которые могут использоваться для поддержки предоставления услуг по размещению веб-информации. Возможны следующие действия:

- Использовать виртуальное размещение, основанное на именах, поскольку в этом случае не требуются дополнительные IP-адреса.
- Настроить DNS-сервер BIND 8 или 9 на прослушивание лишь нескольких сетевых интерфейсов узла – с помощью предписания *listen-on*. Если возникают проблемы с узлом *terminator.movie.edu*, следующий оператор:

  ```
  options {
      listen-on { 192.249.249.3; };
  };
  ```

  объяснит серверу *named* на узле *terminator.movie.edu*, что следует выполнять *bind( )* только для IP-адреса *192.249.249.3*.
- Перенастроить операционную систему, разрешив процессам одновременно открывать большее число файловых дескрипторов.

Клиент сообщает «Looked for PTR, Found CNAME»

Еще одна проблема, связанная со строгостью BIND. В результате некоторых поисковых операций клиент заносит в log-файл сообщения:

```
Sep 24 10:40:11 terminator syslog: gethostby*.getanswer: asked for
     "37.103.74.204.in-addr.arpa IN PTR", got type "CNAME"
Sep 24 10:40:11 terminator syslog: gethostby*.getanswer: asked for
     "37.103.74.204.in-addr.arpa", got "37.32/27.103.74.204.in-addr.arpa"
```

Происходит следующее: клиент запросил у DNS-сервера обратное отображение IP-адреса 204.74.103.37 в доменное имя. Сервер выполнил запрос, но в процессе обнаружил, что *37.103.74.204.in-addr.arpa* в действительности является псевдонимом для *37.32/27.103.74.204.in-addr.arpa*. Практически наверняка это произошло потому, что ребята,

заведующие зоной *103.74.204.in-addr.arpa* для делегирования сегмента пространства имен воспользовались методом, который мы описали в главе 9 «Материнство». Но клиент BIND 4.9.3-BETA таких вещей не понимает и сообщает об ошибке, считая, что доменное имя (запрошенный тип) не было получено. Хотите верьте, хотите – нет, некоторые операционные системы поставляются с DNS-клиентом BIND 4.9.3-BETA в качестве системного.

Единственное решение этой проблемы – обновить клиент до более поздней версии.

DNS-сервер отказывается стартовать: выключены контрольные суммы UDP

На некоторых узлах, работающих под управлением SunOS 4.1.x, можно увидеть такую ошибку:

```
Sep 24 10:40:11 terminator named[7770]: ns_udp checksums NOT turned on:
exiting
```

Сервер *named* намеревался убедиться, что проверка контрольных сумм для UDP включена в системе, и получил отрицательный ответ, после чего завершил работу. Для такого поведения есть веская причина – UDP активно используется *named*, и сервер хочет знать наверняка, что UDP-дейтаграммы приходят нетронутыми.

Проблема решается включением проверки контрольных сумм UDP в системе. Дистрибутив BIND содержит информацию на эту тему в файлах *shres/sunos/INSTALL* и *shres/sunos/ISSUES* (в дистрибутиве BIND 4) либо *src/port/sunos/shres/ISSUES* (в дистрибутиве BIND 8).

Клиент SunOS настроен, но узел не использует DNS

Эта проблема, связана с конкретной реализацией. Некоторые администраторы систем SunOS 4 настраивают клиенты с помощью файла *resolv.conf* и наивно полагают, что *ping*, *telnet* и их собратья сразу заработают. В главе 6 мы говорили, каким образом в SunOS 4 реализован DNS-клиент (в программе *ypserv*, как наверное помнят читатели). Если на узле не запущена служба NIS, простая настройка клиента ни к чему не приведет. Администратору придется создать пустую карту *hosts*, либо заменить функции DNS-клиента. Более подробно оба варианта описаны в разделе «SunOS 4.x от Sun» (глава 6).

Другие DNS-серверы не кэшируют отрицательные ответы

Чтобы заметить эту проблему, нужен очень острый глаз, а если используется BIND 8, придется отключить один из важных механизмов, чтобы столкнуться с ней. В BIND 9 этот механизм отключен по умол-

чанию. Предположим, в случае использования BIND 8 или 9 другие DNS-серверы и DNS-клиенты игнорируют кэшированные отрицательные ответы. Вероятно, предписание *auth-nxdomain* не работает.

auth-nxdomain – это предписание оператора *options*, которое говорит DNS-серверам BIND 8 и 9, что кэшируемые отрицательные ответы следует отмечать как авторитативные, даже если это не так. То есть если DNS-сервер кэшировал информацию о том, что *titanic.movie.edu* не существует, получив ответ от авторитативного DNS-сервера *movie.edu*, *auth-nxdomain* предписывает выдавать этот кэшированный ответ на запросы других DNS-клиентов и серверов так, как если бы наш сервер сам был бы авторитативным для *movie.edu*.

Причина, по которой такой механизм иногда находит применение, заключается в том, что некоторые DNS-серверы проверяют отрицательные ответы (код возврата NXDOMAIN или NOERROR с отсутствующими записями) на авторитативность. Когда отрицательное кэширование еще не существовало, любой отрицательный ответ *должен* был исходить от авторитативного DNS-сервера, и такая проверка представлялась вполне разумной. Но с приходом отрицательного кэширования ситуация изменилась – отрицательный ответ может извлекаться из кэша. Чтобы гарантировать, что более старые DNS-серверы не будут игнорировать такие ответы и тем более не будут считать их ошибками, в BIND 8 и 9 существует возможность неправомочно устанавливать флаг компетентности. Более того, для серверов BIND 8 это поведение по умолчанию, так что навряд ли вы столкнетесь с тем, что удаленные клиенты игнорируют отрицательные ответы, если не будет явно отключен механизм *auth-nxdomain*. В серверах BIND 9, с другой стороны, *auth-nxdomain* по умолчанию не работает, поэтому клиенты вполне способны игнорировать запросы, даже если администратор не прикасался к файлу настройки.

Не определено время жизни

Как было сказано в главе 4, документ RFC 2308 был опубликован буквально перед выходом BIND 8.2. Этот документ изменил семантику последнего поля SOA-записи (оно стало определять отрицательное TTL) и добавил новую директиву, $TTL, позволяющую определять значение TTL по умолчанию в файле данных зоны.

Если произведено обновление до сервера BIND 8 версии более поздней, чем 8.2, но не были добавлены директивы $TTL в файлы данных зон, в log-файле демона *syslog* появятся примерно такие сообщения от DNS-сервера:

```
Sep 26 19:34:39 terminator named[22116]: Zone "movie.edu" (file
db.movie.edu): No default TTL ($TTL <value>) set, using SOA minimum instead
```

BIND 8 благородно полагает, что администратор еще не читал RFC 2308, и позволяет использовать последнее поле SOA-записи в качестве

стандартного TTL для зоны, и также в качестве времени жизни для кэширования отрицательных ответов. Но BIND 9 не так снисходителен:

```
Sep 26 19:35:54 terminator named[22124]: dns_master_load: db.movie.edu:7: no
TTL specified
Sep 26 19:35:54 terminator named[22124]: dns_zone_load: zone movie.edu/IN:
database db.movie.edu: dns_db_load failed: no ttl
Sep 26 19:35:54 terminator named[22124]: loading zones: no ttl
Sep 26 19:35:54 terminator named[22124]: exiting (due to fatal error)
```

Перед обновлением до BIND 9 не забудьте добавить соответствующие директивы $TTL.

Ошибки TSIG

Как мы уже говорили в главе 11, для работы транзакционных подписей требуются синхронизация по времени и по ключу (обе стороны должны использовать одинаковый ключ и одинаковое имя для этого ключа). Вот некоторые ошибки, которые могут появиться, если утрачена синхронизация по времени либо используются разные ключи (имена ключей).

Вот ошибка, которую можно получить от DNS-сервера BIND 8, если произвести настройку TSIG, но забыть избавиться от разрыва во времени между первичным мастером и вторичным DNS-серверами:

```
Sep 27 10:47:49 wormhole named[22139]: Err/TO getting serial# for "movie.edu"
Sep 27 10:47:49 wormhole named-xfer[22584]: SOA TSIG verification from server
[192.249.249.3], zone movie.edu: message had BADTIME set (18)
```

DNS-сервер пытался проверить порядковый номер зоны *movie.edu*, обратившись к узлу *terminator.movie.edu* (192.249.249.3). Ответ от узла *terminator.movie.edu* не принят, поскольку время на часах *wormhole.movie.edu* более чем на десять минут отличается от времени, которым подписан ответ. Сообщение *Err/TO* – просто побочный продукт отрицательных результатов проверки ответа с TSIG-подписью.

Если стороны используют различные имена ключей, даже в случае идентичности самих ключей от DNS-сервера BIND 8 будет получена примерно следующая ошибка:

```
Sep 27 12:02:44 wormhole named-xfer[22651]: SOA TSIG verification from server
[209.8.5.250], zone movie.edu: BADKEY(-17)
```

На этот раз проверка ответного сообщения с TSIG-подписью привела к отрицательным результатам, поскольку проверяющая сторона не смогла найти ключ с указанным в TSIG-записи именем. Такая же ошибка генерируется, если совпадают имена ключей, но эти имена связаны с различающимися ключами.

Как обычно, вывод сервера BIND 9 намного более лаконичен. Для третьего уровня отладки и любого из предыдущих вариантов получается примерно следующее:

```
Sep 27 13:35:42.804 client 192.249.249.1#1115: query: movie.edu SOA
Sep 27 13:35:42.804 client 192.249.249.1#1115: error
```

Симптомы проблем

К сожалению, некоторые проблемы определяются не так легко, как уже описанные. Вы столкнетесь со странным поведением программ, но не сможете четко определить причину, поскольку к наблюдаемым симптомам могут приводить различные причины. Для таких случаев мы рассмотрим наиболее вероятные причины и способы их устранения.

Локальное имя не найдено

Когда программа вроде *telnet* или *ftp* утверждает, что не может найти адрес локального доменного имени, прежде всего следует применить *nslookup* или *dig* – и выполнить поиск для того же имени. Говоря «для того же имени» – мы имеем в виду *в точности* то же имя – без добавления меток или последней точки, если ее не было в исходном имени. Обращайтесь к тому же DNS-серверу, что использовался для неудавшихся запросов.

Довольно часто дело в опечатке, сделанной пользователем, либо в том, что он не понимает принципов работы списка поиска – в таком случае ему просто нужен совет. Но время от времени речь идет действительно об ошибках настройки узла:

- Синтаксические ошибки в файле *resolv.conf* (проблема 11 в разделе «Перечень возможных проблем», см. выше).
- Не определенное локальное доменное имя (проблема 12).

Обе возможности можно исследовать с помощью команды *set all* программы *nslookup*.

Если работа с *nslookup* выявляет проблему с DNS-сервером, а не с настройкой узла, следует искать проблему, связанную с типом DNS-сервера. Если DNS-сервер является первичным мастером для зоны, но не выдает ожидаемой информации:

- Убедитесь, что файл данных зоны содержит данные, о которых идет речь, и что этот файл загружен DNS-сервером (проблема 2). Чтобы определить, были ли загружены данные, можно воспользоваться дампом базы данных.
- Проверьте файл настройки и соответствующий файл данных зоны на наличие синтаксических ошибок (проблема 5). Соответствующие сообщения должны содержаться в log-файле демона *syslog*.

- Убедитесь, что в записях присутствуют последние точки в именах – там, где их присутствие необходимо (проблема 6).

Если DNS-сервер является вторичным для зоны, следует прежде всего удостовериться в корректности данных на DNS-мастер-сервере. Если данные основного сервера в порядке, но того же нельзя сказать о дополнительном:

- Убедитесь, что порядковый номер зоны на основном DNS-сервере был увеличен (проблема 1).
- Займитесь поиском проблемы, связанной с загрузкой зоны вторичным сервером (проблема 3).

Если данные на основном сервере не являются корректными, следует заняться диагностированием проблем основного сервера.

Если DNS-сервер, о котором идет речь, специализируется на кэшировании:

- Убедитесь, что для него существуют данные корневых указателей (проблема 7).
- Проверьте корректность делегирования в родительской зоны (проблемы 9 и 10). Следует помнить, что для выделенного под кэширование сервера зона выглядит точно так же, как и любая другая из внешних зон. Пусть даже сервер работает на узле в пределах зоны, специальный кэширующий DNS-сервер должен иметь возможность найти авторитативный сервер для этой зоны, обратившись к DNS-серверам родительской зоны.

Невозможно произвести поиск для удаленного имени

Если поиск для локальных имен проходит успешно, но для любых доменных имен вне локальной зоны заканчивается неудачей, следует задуматься о несколько другом наборе возможных проблем:

- DNS-сервер только что установлен? Возможно, были забыты данные корневых указателей (проблема 7).
- Возвращаются ли пакеты *ping*, отправленные DNS-серверам внешней зоны? Возможно, что DNS-серверы недоступны из-за сбоя в сети (проблема 8).
- Внешняя зона только что появилась? Возможно, информация о делегировании еще не распространена (проблема 9). Помимо этого, информация о делегировании для внешней зоны может быть некорректной либо устаревшей (проблема 10).
- Действительно ли существует доменное имя в данных DNS-серверах внешней зоны (проблема 2)? Вполне возможно, что это верно только для некоторых из них (проблемы 1 и 3).

Неверные или противоречивые ответы

Если поиск для локального доменного имени приводит к получению неверного или противоречивого ответа, зависящего от используемого DNS-сервера или времени выполнения запроса, прежде всего следует проверить наличие синхронизации DNS-серверов:

- Одинаков ли порядковый номер хранимых зон для этих DNS-серверов? Был ли увеличен порядковый номер на первичном мастер-сервере после внесения в зону изменений (проблема 1)? Если нет, порядковый номер хранимых зон на различных серверах будет одинаковым, но они будут давать различные ответы, когда речь пойдет о данных в пределах авторитативности.
- Не был ли сброшен порядковый номер зоны в единицу (снова проблема 1)? В этом случае порядковый номер на первичном DNS-мастер-сервере будет гораздо меньше соответствующих номеров на дополнительных.
- Первичный мастер-сервер не был перезагружен (проблема 2)? В этом случае он будет возвращать (скажем, программам *nslookup* и *dig*) порядковый номер, отличный от хранимого в файле данных зоны.
- Существуют ли у дополнительных DNS-серверов проблемы, связанные с обновлением информации с основных (проблема 3)? В таком случае, log-файл *syslog* должен содержать соответствующие сообщения обо ошибках.
- Производит ли механизм round robin перестановку адресов, получаемых для доменного имени?

Если подобные результаты получаются при поиске для доменного имени из внешней зоны, следует проверить, не сбилась ли синхронизация DNS-серверов этой зоны. Чтобы определить, скажем, не забыл ли администратор внешней зоны увеличить ее порядковый номер, можно воспользоваться программами *nslookup* и *dig*. Если DNS-серверы возвращают различные авторитативные ответы, но возвращают одинаковые порядковые номера для зоны, то скорее всего имела место такая ошибка администратора. Если порядковый номер первичного DNS-мастер-сервера намного меньше, чем у дополнительных DNS-серверов, это говорит о том, что произошел случайный сброс порядкового номера основного сервера. Обычно мы предполагаем, что первичный DNS-мастер-сервер работает на узле, указанном в поле MNAME (первое поле) SOA-записи.

Возможно, однако, вам не удается вывести заключение о том, что первичный мастер не был перезагружен. Также бывает сложно выявить точное место возникновения проблем, связанных с обновлениями зон между удаленными серверами. В подобных случаях, если вы определили, что удаленные DNS-серверы выдают некорректные ответы, свяжитесь с администратором зоны и (вежливо) сообщите ему то, что вы наблюдаете. Это поможет ему выявить проблему на том конце.

Если можно точно определить, что родительский DNS-сервер внешней зоны, либо локальной зоны, либо один из серверов локальной зоны возвращает неправильные ответы, проверьте, не связано ли это с устаревшей информацией о делегировании. Возможно, придется связаться как с администратором внешней зоны, так и с администратором родительской зоны, чтобы сверить информацию о делегировании и списки существующих авторитативных DNS-серверов.

Если невозможно повлиять на администратора, чтобы он урегулировал проблему, а также в случаях, когда администратора невозможно найти, можно воспользоваться предписанием *bogus* инструкции *bogusns*, чтобы объяснить своему DNS-серверу, что какому-то конкретному серверу нет смысла посылать запросы.

Поиск занимает много времени

Медленное разрешение имен обычно происходит по двум причинам:

- Проблемы с сетевым соединением (проблема 8), которые могут быть продиагностированы с помощью отладочного вывода DNS-сервера и инструментов вроде *ping*.
- Некорректная информация о делегировании (проблема 10), содержащая неправильные ссылки на DNS-серверы либо неправильные IP-адреса.

Чаще всего чтение отладочного вывода и посылка *ping* пакетов в нескольких направлениях приводит к одному из следующих выводов: маршруты до DNS-серверов отсутствуют, либо DNS-серверы не отвечают.

Однако существуют случаи, когда сделать однозначный вывод невозможно. К примеру, родительские DNS-серверы делегируют ответственность DNS-серверам, которые не отвечают на пакеты *ping* или другие запросы, но с маршрутом до удаленной сети все в порядке (то есть *traceroute* доводит пользователя до «крыльца» этой сети – последнего маршрутизатора на пути между сетями). Возможно, информация о делегировании чрезмерно устарела, и DNS-серверы давным-давно проживают по другим адресам. Возможно, узлы не работают. Возможно, существует проблема в удаленной сети. Обычно, чтобы узнать наверняка, требуется позвонить или написать письмо администратору удаленной зоны. (Помните, что *whois* предоставляет телефонные номера!)

rlogin и rsh – в доступе отказано

Эта проблема встречается непосредственно после установки DNS-серверов. Пользователи, которые не в курсе замены таблицы узлов службой доменных имен, не знают, что следует обновить файлы *.rhosts*. (Требуемые изменения мы рассмотрели в главе 6.) Как следствие, проверка прав доступа при использовании *rlogin* или *rsh* будет завершаться с отрицательным результатом.

Другие возможные причины – отсутствие, либо некорректное делегирование зоны *in-addr.arpa* (проблемы 9 и 10), либо отсутствие PTR-записи для узла клиента (проблема 4). Если недавно было произведено обновление до BIND версии 4.9 или более новой, и PTR-данные для более чем одной зоны *in-addr.arpa* содержатся в каком-то одном файле, DNS-сервер, возможно, игнорирует «внезональные» данные, что и приводит к осложнениям. В любом случае, поведение программ-клиентов будет однозначным:

```
% rlogin wormhole
Password:
```

Иначе говоря, пользователю предлагается ввести пароль, несмотря на то, что был настроен доступ без пароля – с помощью файлов *.rhosts* или *hosts.equiv*. Заглянув в log-файл демона *syslog* на целевом узле (в данном случае – *wormhole.movie.edu*), мы, вероятно, увидели бы следующее сообщение:

```
May  4 18:06:22 wormhole inetd[22514]: login/tcp: Connection
       from unknown (192.249.249.213)
```

Причину проблемы можно определить, пройдя через процесс разрешения вместе с любимым инструментом для создания запросов. Прежде всего следует запросить у одного из серверов родительской зоны *in-addr.arpa* NS-записи для хранимой зоны *in-addr.arpa*. Если записи корректны, послать перечисленным серверам запрос PTR-записей, соответствующих IP-адресу клиента *rlogin* или *rsh*. Убедитесь, что все серверы возвращают правильную PRT-запись, которая обеспечивает отображение в правильное доменное имя. В противном случае речь идет об отсутствии синхронизации между мастером и вторичным сервером (проблемы 1 и 3).

Отказано в доступе к службам

Иногда перестают работать не только *rlogin* и *rsh*. Установив BIND на сервере, администратор может обнаружить, что перестали загружаться бездисковые станции, а узлы не могут монтировать диски с сервера.

В таком случае следует убедиться, что регистр имен, возвращаемых сервером BIND, совпадает с регистром имен, который использовался в предыдущей службе имен. К примеру, если используется NIS и в картах NIS содержатся только имена в нижнем регистре, то DNS-серверы также должны возвращать имена в нижнем регистре. Некоторые программы чувствительны к регистру символов и по этой причине могут испытывать трудности с распознаванием имен в таких файлах, как */etc/bootparams* или */etc/exports*.

Невозможно избавиться от старых данных

После демобилизации DNS-сервера либо изменения IP-адреса одного из серверов можно обнаружить, что старые адресные записи подзадержались на этом свете. Старая запись может существовать в кэше DNS-сервера или в файле данных зоны недели и даже месяцы спустя. Казалось бы, запись должна была давно устареть и исчезнуть из всех кэшей. Так в чем же дело? Есть несколько вероятных причин. Мы начнем с самых простых случаев.

Старая информация о делегировании

Первый (и самый простой) вариант связан с тем, что родительская зона не поспевает за детьми, либо дети не уведомляют родителей об изменениях, связанных с авторитативными DNS-серверами зоны. Если администраторы *edu* хранят следующую (устаревшую) информацию о делегировании для *movie.edu*:

```
$ORIGIN movie.edu.
@     86400    IN    NS    terminator
      86400    IN    NS    wormhole
terminator     86400    IN    A    192.249.249.3
wormhole       86400    IN    A    192.249.249.254 ; wormhole's former
                                     ; IP address
```

DNS-серверы зоны *edu* будут возвращать фальшивый старый адрес *wormhole.movie.edu*.

Ситуацию легко исправить, поняв, что дело в DNS-серверах родительской зоны: достаточно просто связаться с администратором родительской зоны и попросить обновить информацию о делегировании. Если родительская зона является одной из родовых зон высшего уровня, проблему, скорее всего, можно разрешить, заполнив форму на веб-сайте регистратора и изменив таким образом информацию о DNS-сервере. Если данные кэшированы одним из DNS-серверов самой зоны, его можно остановить (в целях удаления кэшированных данных), удалить резервные копии файлов данных, содержащих устаревшую информацию, а затем запустить вновь.

Зарегистрированный сервер не является DNS-сервером

Эта проблема встречается только в gTLD-зонах *com*, *net* и *org*. Можно обнаружить, что DNS-серверы такой зоны выдают устаревшую информацию об узле в одной из наших зон, причем этот узел не является DNS-сервером! С какой целью DNS-серверы gTLD-зоны хранят информацию о произвольном узле одной из наших зон?

Ответ таков: в gTLD-зонах можно регистрировать узлы, которые не являются DNS-серверами, к примеру, узлы веб-серверов. Скажем, можно зарегистрировать адрес *www.foo.com*, воспользовавшись услугами

одного из *com*-регистраторов, и DNS-серверы *com* будут выдавать именно этот адрес. Но не следует этого делать, поскольку возможности работы с адресом сокращаются. Если адрес понадобится изменить, выполнение этой процедуры регистратором может занять в лучшем случае день. Если же мы управляем первичным DNS-мастер-сервером зоны *foo.com*, то можем внести изменение мгновенно.

Что со мной?

Как определить, какая из проблем действительно имеет место? Следует обращать внимание на то, какие DNS-серверы распространяют устаревшую информацию и к каким зонам относятся эти данные:

- DNS-сервер gTLD-зоны? Вероятно, он хранит устаревший, но зарегистрированный адрес.
- DNS-сервер родительской зоны, но не зоны gTLD? Вероятно, устарела информация о делегировании.

Вот и все, что мы собирались рассмотреть в этой главе. Разумеется, перечень не полон, но мы надеемся, что он поможет справиться с наиболее распространенными сложностями, которые встречаются при работе с DNS, и вполне дает представление о том, как подходить к решению проблем, которые не описаны. Ха, вот если бы *у нас* было руководство по разрешению проблем, когда *мы* начинали!

15

- *Написание сценариев командного интерпретатора с помощью программы nslookup*
- *Программирование на языке C при помощи функций библиотеки DNS-клиента*
- *Программирование на языке Perl при помощи модуля Net::DNS*

Программирование при помощи функций библиотеки DNS–клиента

– Я знаю, о чем ты думаешь, – сказал Труляля, – но это не так! Ни в коем разе!
– И задом наперед, совсем наоборот, – подхватил Траляля. – Если бы это было так, это бы еще ничего, а если бы ничего, оно бы так и было, но так как это не так, так оно и не этак! Такова логика вещей!

Готовы спорить, читателям кажется, что программирование с использованием клиента – вещь невероятно сложная. Напротив! В действительности ничего сложного нет. Формат сообщений DNS достаточно прост – нет необходимости иметь дело с ASN.1[1], как в случае с SNMP. К тому же, существуют отличные библиотечные функции, облегчающие разбор сообщений DNS. Мы приводим отдельные фрагменты документа RFC 1035 в приложении А «Формат сообщений DNS и RR-записей». Но не будет лишним иметь под рукой копию RFC 1035 при чтении этой главы или доступ к этому документу в процессе написания программ, работающих с DNS.

[1] ASN.1 (Abstract Syntax Notation) – это способ определения типов объектов, принятый в качестве международного стандарта организацией ISO (International Organization for Standardization, Международная организация стандартизации).

Написание сценариев командного интерпретатора с помощью программы nslookup

Прежде чем вы возьметесь за язык C и начнете писать программу, которая будет выполнять всю грязную работу, связанную с DNS, следует написать программу на языке командного интерпретатора, используя *nslookup* или *dig*. Тому существует несколько веских причин:

- Сценарий интерпретатора можно создать гораздо быстрее, чем программу на языке C.
- Если вы плохо знакомы с DNS, особенности логики программы можно понять, набросав прототип в виде сценария. В процессе написания C-программы, вы уже не будете отвлекаться на основную функциональность и связанные с ней вопросы, и сможете сконцентрировать внимание на дополнительных возможностях.
- Может оказаться, что версия программы на языке командного интерпретатора выполняет свою работу вполне удовлетворительно, поэтому нет никакого смысла писать программу на C. Сценарии интерпретатора не только быстрее создаются, они проще в сопровождении, если есть необходимость использовать их в течение долгого времени.

Те из читателей, кто предпочитает сценариям интерпретатора сценарии на языке Perl, могут писать первую версию программы на этом языке. В последнем разделе этой главы мы расскажем об использовании модуля Perl Net::DNS, за авторством Майкла Фера (Michael Fuhr).

Типичная проблема

Прежде чем мы начнем писать программу, необходимо определиться с проблемой, которую эта программа будет решать. Предположим, мы хотим, чтобы наша система управления сетью присматривала за первичным мастером и вторичными DNS-серверами. Программа должна выдавать уведомление о нескольких типах проблем: DNS-сервер не запущен (он вполне мог просто прекратить работать), DNS-сервер, который должен быть авторитативным для зоны, не является таковым (возможно, присутствуют ошибки в файле настройки или файлах данных зоны) либо DNS-сервер отказывается обновлять хранимую зону (порядковый номер зоны на первичном DNS-мастер-сервере случайно уменьшился).

Каждая из этих проблем может быть с легкостью зарегистрирована. Если DNS-сервер не работает на узле, узел возвращает ICMP-сообщение о недоступности порта (*port unreachable*). Эту информацию можно получить любой программой, позволяющей делать запросы, либо с помощью функций клиента. Проверить, является ли DNS-сервер автори-

тативным для зоны, тоже легко: получить SOA-запись этого сервера. Если ответ является неавторитативным или SOA-запись отсутствует, значит, у нас проблемы. Необходимо запрашивать SOA-запись в *нерекурсивном* запросе, чтобы DNS-сервер не пытался искать эту запись на других DNS-серверах. А получив SOA-запись, мы получаем и порядковый номер зоны.

Решение задачи с помощью сценария

Для решения этой задачи необходима программа, которая принимает доменное имя зоны в качестве аргумента, находит DNS-серверы этой зоны и запрашивает у всех DNS-серверов SOA-запись для зоны. Ответ в каждом случае позволяет определить, является ли DNS-сервер авторитативным, а также позволяет получить порядковый номер зоны. Если ответа нет, программа должна определить, работает ли на указанном узле DNS-сервер. После написания этой программы ее следует выполнить для каждой из зон, которым необходима диагностика. Поскольку программа занимается поиском DNS-серверов (используя NS-записи зоны), мы будем предполагать, что все DNS-серверы перечислены в зональных данных. В противном случае придется изменить программу таким образом, чтобы она получала список DNS-серверов в качестве аргументов командной строки.

Итак, приступаем к написанию сценария, который использует *nslookup*. Прежде всего необходимо определить, как выглядит вывод команды *nslookup*, чтобы иметь возможность разобрать результаты с помощью инструментария Unix-системы. Попробуем произвести поиск NS-записей с целью выяснения, какие из DNS-серверов должны являться авторитативными для зоны, как с использованием авторитативного сервера для данной зоны, так и неавторитативного:

```
% nslookup
Default Server:  relay.hp.com
Address:  15.255.152.2

> set type=ns
```

Определим, как выглядит результат, если DNS-сервер не является авторитативным в смысле поставки NS-записей:

```
> mit.edu.
Server:  relay.hp.com
Address:  15.255.152.2

Non-authoritative answer:
mit.edu nameserver = STRAWB.MIT.EDU
mit.edu nameserver = W20NS.MIT.EDU
mit.edu nameserver = BITSY.MIT.EDU

Authoritative answers can be found from:
MIT.EDU nameserver = STRAWB.MIT.EDU
MIT.EDU nameserver = W20NS.MIT.EDU
```

```
MIT.EDU nameserver = BITSY.MIT.EDU
STRAWB.MIT.EDU  internet address = 18.71.0.151
W20NS.MIT.EDU   internet address = 18.70.0.160
BITSY.MIT.EDU   internet address = 18.72.0.3
```

Теперь определим результат для авторитативного DNS-сервера:

```
> server strawb.mit.edu.
Default Server:  strawb.mit.edu
Address:  18.71.0.151

> mit.edu.
Server:  strawb.mit.edu
Address:  18.71.0.151

mit.edu nameserver = BITSY.MIT.EDU
mit.edu nameserver = STRAWB.MIT.EDU
mit.edu nameserver = W20NS.MIT.EDU
BITSY.MIT.EDU   internet address = 18.72.0.3
STRAWB.MIT.EDU  internet address = 18.71.0.151
W20NS.MIT.EDU   internet address = 18.70.0.160
```

Можно видеть, что доменные имена DNS-серверов можно получить путем сохранения всех последних полей строк, содержащих слово *nameserver*. В случае, когда сервер не является авторитативным в смысле поставки NS-записей, эти записи отображаются по два раза, поэтому необходимо будет избавиться от повторяющихся значений.

Теперь произведем поиск SOA-записи для зоны для случаев, когда сервер является авторитативным для зоны, содержащей SOA-запись, и, соответственно, для случаев, когда он таковым не является. Мы выключаем *рекурсию*, чтобы DNS-сервер не запрашивал SOA-запись у другого, авторитативного сервера:

```
% nslookup
Default Server:  relay.hp.com
Address:  15.255.152.2

> set type=soa
> set norecurse
```

Выясним, как выглядит ответ в случае, если DNS-сервер не является авторитативным и не владеет SOA-записью:

```
> mit.edu.
Server:  relay.hp.com
Address:  15.255.152.2

Authoritative answers can be found from:
MIT.EDU nameserver = STRAWB.MIT.EDU
MIT.EDU nameserver = W20NS.MIT.EDU
MIT.EDU nameserver = BITSY.MIT.EDU
STRAWB.MIT.EDU  internet address = 18.71.0.151
W20NS.MIT.EDU   internet address = 18.70.0.160
BITSY.MIT.EDU   internet address = 18.72.0.3
```

Теперь выясним, как выглядит ответ в случае, если DNS-сервер *является* авторитативным для зоны:

```
> server strawb.mit.edu.
Default Server:  strawb.mit.edu
Address:  18.71.0.151

> mit.edu.
Server:  strawb.mit.edu
Address:  18.71.0.151

mit.edu
        origin = BITSY.MIT.EDU
        mail addr = NETWORK-REQUEST.BITSY.MIT.EDU
        serial = 1995
        refresh = 3600 (1H)
        retry   = 900 (15M)
        expire  = 3600000 (5w6d16h)
        minimum ttl = 21600 (6H)
```

Если DNS-сервер не является авторитативным для зоны, он возвращает ссылки на другие серверы. Если DNS-сервер до этого производил поиск SOA-записи и кэшировал ее, то в неавторитативном ответе будет возвращена эта SOA-запись. Следует проверять оба варианта. Если DNS-сервер возвращает SOA-запись и является авторитативным, можно извлечь порядковый номер зоны из строки, содержащей слово *serial*.

Теперь мы должны выяснить, какой результат возвращает *nslookup* в случае, когда DNS-сервер отсутствует на указанном узле. Мы попробуем произвести поиск SOA-записи на узле, про который известно, что на нем не работает DNS-сервер:

```
% nslookup
Default Server:  relay.hp.com
Address:  15.255.152.2

> server galt.cs.purdue.edu.
Default Server:  galt.cs.purdue.edu
Address:  128.10.2.39

> set type=soa
> mit.edu.
Server:  galt.cs.purdue.edu
Address:  128.10.2.39

*** galt.cs.purdue.edu can't find mit.edu.: No response from server
```

Наконец, необходимо понять, что возвращает *nslookup* в случае отсутствия ответа от узла. Это можно сделать, сменив DNS-сервер на неиспользуемый адрес локальной сети:

```
% nslookup
Default Server:  relay.hp.com
```

```
Address:  15.255.152.2

> server 15.255.152.100
Default Server:  [15.255.152.100]
Address:  15.255.152.100

> set type=soa
> mit.edu.
Server:  [15.255.152.100]
Address:  15.255.152.100

*** Request to [15.255.152.100] timed-out
```

В двух последних случаях сообщение об ошибке было напечатано в стандартный поток ошибок, *stderr*[1]. Мы можем использовать это обстоятельство при написании сценария интерпретатора. Теперь мы готовы к собственно созданию сценария. Назовем его *check_soa*:

```
#!/bin/sh
if test "$1" = ""
then
    echo Применение: $0 зона
    exit 1
fi
ZONE=$1
#
# Используем nslookup для получения списка DNS-серверов для этой зоны ($1).
# Используем awk, чтобы извлечь доменные имена DNS-серверов из строк со словом
# nameserver.(Имена всегда присутствуют в качестве последнего поля.)
# Используем sort -u для удаления дублирующихся строк;
# количество для нас не главное.
#
SERVERS=`nslookup -type=ns $ZONE |\
                 awk '/nameserver/ {print $NF}' | sort -u`
if test "$SERVERS" = ""
then
    #
    # Не были найдены серверы. Обычное завершение работы;
    # nslookup зарегистрирует ошибку и выдаст соответствующее сообщение.
    # Этого вполне достаточно.
    #
    exit 1
fi
#
# Проверка порядковых номеров в SOA-записях серверов. Вывод nslookup
# сохраняется в двух временных файлах: nso.$$ (стандартный вывод)
# и nse.$$ (стандартный поток ошибок). Эти файлы перезаписываются
# на каждой итерации. defname и search выключаются, поскольку
```

[1] Не все версии программы *nslookup* отображают последнее сообщение об ошибке для случаев окончания интервала ожидания. Не забудьте уточнить, как обстоят дела в вашем случае.

```
# мы должны работать только с абсолютными доменными именами.
#
# ПРИМЕЧАНИЕ: выполнение цикла занимает довольно много времени;
# не стоит пугаться.
#
for i in $SERVERS
do
  nslookup >/tmp/nso.$$ 2>/tmp/nse.$$ <<-EOF
    server $i
    set nosearch
    set nodefname
    set norecurse
    set q=soa
    $ZONE
EOF
  #
  # Говорит ли полученный ответ о том, что текущий сервер ($i) является
  # авторитативным? Сервер НЕ является авторитативным, если (a) это прямо
  # сказано в ответе либо (b) в ответе содержится информация о том,
  # что авторитативные ответы стоит поискать в другом месте.
  #
  if egrep "Non-authoritative|Authoritative answers can be" \
                                        /tmp/nso.$$ >/dev/null
  then
    echo $i не является авторитативным для зоны $ZONE
    continue
  fi
  #
  # Нам известно, что сервер авторитативный; извлекаем порядковый номер.
  #
  SERIAL=`cat /tmp/nso.$$ | grep serial | sed -e "s/.*= //"`
  if test "$SERIAL" = ""
  then
    #
    # Эта ветвь выполняется, если порядковый номер представлен пустой
    # строкой. В этом случае должно поступить сообщение об ошибке от
    # nslookup; выполняем cat для файла "стандартного потока ошибок".
    #
    cat /tmp/nse.$$
  else
    #
    # Отображаем доменное имя сервера и порядковый номер.
    #
    echo $i имеет порядковый номер $SERIAL
  fi
done  # конец цикла "for"
#
# Удаляем временные файлы.
#
rm -f /tmp/nso.$$ /tmp/nse.$$
```

Вот так выглядит вывод:

```
% check_soa mit.edu
BIT-SY.MIT.EDU имеет порядковый номер 1995
STRAWB.MIT.EDU имеет порядковый номер 1995
W20NS.MIT.EDU имеет порядковый номер 1995
```

Если у вас мало времени, эта простая программа поможет его сэкономить, решив задачу. Если вы обнаружили, что речь идет о проверке слишком многих зон, имеет смысл преобразовать сценарий в С-программу. Помимо этого, если есть необходимость более тонко обращаться с сообщениями об ошибках, не полагаясь на вывод программы *nslookup*, придется писать программу на языке С. Именно это мы сделаем чуть позже.

Программирование на языке С при помощи функций библиотеки DNS-клиента

Но прежде чем начать писать конкретный код, следует познакомиться с форматом сообщений DNS и функциями библиотеки DNS-клиента. В только что написанном сценарии интерпретатора разбором сообщений DNS занималась программа *nslookup*. Но в программе на языке С разбором придется заниматься программисту. Этот раздел мы начнем с изучения формата сообщений DNS.

Формат сообщений DNS

Читатели уже встречались с форматом сообщений DNS ранее, в главе 12 «nslookup и dig». Сообщение состоит из следующих разделов:

- Раздел заголовка
- Раздел вопроса
- Раздел ответа
- Раздел авторитативности
- Дополнительный раздел

Формат раздела заголовка описан в документе RFC 1035, на страницах с 26 по 28, а также в приложении А. Раздел заголовка выглядит следующим образом:

```
номер идентификации запроса(2 октета)
ответ на запрос (1 бит)
код операции (4 бита)
авторитативный ответ (1 бит)
усечение (1 бит)
необходимость рекурсии (1 бит)
доступность рекурсии (1 бит)
зарезервировано (3 бита)
```

```
код ответа (4 бита )
счетчик ответа (2 октета)
счетчик записей ответа (2 октета)
счетчик записей DNS-сервера (2 октета)
счетчик дополнительных записей (2 octets)
```

Значения кода операции, кода ответа, типа и класса определены в файле *arpa/nameser.h*, наряду с функциями извлечения этой информации из сообщения. Скоро мы обсудим эти функции, входящие в библиотеку DNS-сервера.

Раздел вопроса описан на страницах 28 и 29 документа RFC 1035. Он выглядит следующим образом:

```
доменное имя (переменной длины)
тип запроса (2 октета)
класс запроса (2 октета)
```

Разделы ответа, авторитативности и дополнительный описаны на страницах 29 и 30 документа RFC 1035. Эти разделы состоят из переменного числа RR-записей, которые выглядят следующим образом:

```
доменное имя (переменной длины)
тип (2 октета)
класс (2 октета)
TTL (4 октета)
длина сегмента данных (2 октета)
сегмент данных (переменной длины)
```

Раздел заголовка содержит счетчики RR-записей для каждого раздела.

Хранение доменных имен

Как можно видеть, имена, передаваемые в сообщениях DNS, имеют переменную длину. DNS не хранит имена в виде строк, завершаемых нулем, как это принято в языке C. Доменные имена хранятся в виде наборов пар длина/значение, заканчивающихся нулевым октетом. Каждая метка в доменном имени представляется октетом длины и собственно меткой. Имя *venera.isi.edu* хранится в виде:

```
6 venera  3  isi  3 edu 0
```

Теперь представьте себе, сколько места в сообщении DNS могут занимать передаваемые имена. Разработчики DNS распознали эту проблему и придумали простой способ упаковки доменных имен.

Упаковка доменных имен

Довольно часто все доменное имя либо несколько последних меток доменного имени, соответствует имени, которое уже содержится в сообщении. Упаковка доменных имен исключает повторение доменных

имен путем замены ранее встречавшихся имен или их частей указателями. Механизм работает следующим образом. Допустим, сообщение с ответом уже содержит имя *venera.isi.edu*. Если к ответу добавляется имя *vaxa.isi.edu*, реально добавляется метка *vaxa*, а затем указатель на предыдущее вхождение *isi.edu*. Как реализованы эти указатели?

Первые два бита октета длины определяют, что именно следует за ними – пара длина-метка или указатель на такую пару. Если первые два бита нулевые, следует длина и метка. Читатели, вероятно, помнят, что в главе 2 «Как работает DNS» мы рассказывали, что длина метки ограничена 63 символами. Это происходит потому, что поле длины содержит лишь 6 свободных битов для хранения длины метки – и их хватает на хранение числа из диапазона от 0 до 63. Если первые два бита октета длины – единицы, за ними следует не длина, но указатель. Указатель состоит из последних шести битов октета длины *и* всего следующего октета, поэтому его длина составляет 14 битов. Указатель является смещением от начала сообщения DNS. Итак, когда имя *vaxa.isi.edu* упаковывается в буфере, который содержит только имя *venera.isi.edu*, получается следующий результат:

```
смещение в байтах: 0 123456 7 890 1 234 5 6 7890 1    2
                    -------------+--------------+--------
содержимое пакета: 6 venera 3 isi 3 edu 0 4 vaxa 0xC0 7
```

Значение *0xC0* представляет байт, старшие два бита которого установлены в единицу, а остальные сброшены в нуль. Поскольку два старших бита оба равны единице, речь идет об указателе, а не о значении длины. Значение указателя – семь, поскольку все шесть значащих битов первого октета сброшены в нуль, а второй октет имеет значение 7. По смещению семь в данном буфере расположена оставшаяся часть доменного имени, которое начинается с *vaxa*, то есть *isi.edu*.

В приведенном примере речь шла об упаковке лишь двух доменных имен в буфере, а не об упаковке целого сообщения DNS. В сообщении DNS, наряду с другими полями, присутствовал бы заголовок. Этот пример призван лишь дать читателям представление о том, как работает упаковка доменных имен. Есть хорошая новость: в действительности, нет большой необходимости думать о том, как упаковываются имена, если этим занимаются работающие библиотечные функции. Необходимо помнить, что разбор сообщения DNS может привести к получению абсолютной каши, если ошибиться хотя бы на один байт. Попробуйте распаковать имя, которое начинается со второго байта, а не с первого. И обнаружите, что буква «v» – очень хороший октет длины или указатель.

Функции библиотеки DNS-клиента

Библиотека клиента содержит функции, необходимые для создания приложений. Эти функции используются для создания запросов. Опи-

санные ниже функции *библиотеки DNS-сервера* используются для разбора ответов.

Может возникнуть вопрос: почему мы не пользуемся функциями клиента BIND 9 при создании собственного кода. Так вот, ответ заключается в том, что таковые еще не существуют. В BIND 9 присутствуют библиотечные функции, выполняющие многочисленные и важные операции DNS, но эти функции предназначены для использования DNS-сервером BIND 9 и очень сложны в использовании, как нам это известно. Разработчики сообщили нам, что более простая библиотека клиента уже на подходе, а пока что следует использовать библиотеку клиента BIND версии 8. Программа, собранная с использованием функций библиотеки клиента BIND 8, будет без проблем работать с DNS-сервером BIND 9.

Вот заголовочные файлы, которые необходимо включать в программу:

```
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/nameser.h>
#include <resolv.h>
```

А теперь перейдем к изучению функций библиотеки клиента.

---

res_search

```
int res_search(const char *dname,
               int class,
               int type,
               u_char *answer,
               int anslen)
```

res_search – это функция «самого высокого уровня», она используется функцией *gethostbyname*. *res_search* применяет алгоритм поиска к полученному доменному имени. Полученное доменное имя (*dname*) «дополняется» (в случае, когда оно не является абсолютным) различными доменными именами из списка поиска клиента, и для каждого имени вызывается функция *res_query*, пока не будет получен положительный результат. Положительным результатом считается существующее абсолютное доменное имя. Помимо реализации алгоритма поиска *res_search* производит чтение файла, имя которого определяется переменной среды HOSTALIASES. (Переменная HOSTALIASES описана в главе 6 «Конфигурирование узлов».) Таким образом, функция обращает внимание на все определенные «частным образом» псевдонимы узлов. *res_search* возвращает размер ответа, либо заполняет переменную *h_errno* и возвращает значение –1 – в случае наличия ошибки либо при количестве ответов, равном нулю. (Переменная *h_errno* похожа на переменную *errno*, но используется при поиске в DNS.)

Таким образом, единственным параметром, представляющим реальный интерес для функции *res_search*, является *dname*; все прочие передаются для использования функций *res_query* и другими функциями клиента. Вот эти аргументы:

class

Класс данных, являющихся предметом поиска. Этот аргумент практически всегда является константой *C_IN*, которая определяет класс Интернет. Константы классов определены в файле *arpa/nameser.h*.

type

Тип данных, которые являются предметом поиска. И снова это одна из констант, определенных в файле *arpa/nameser.h*. Часто встречается значение T_NS, позволяющее получить записи DNS-серверов, или T_MX, инициирующее поиск MX-записей.

answer

Буфер, в который функция *res_search* записывает ответное сообщение. Минимальный размер буфера – PACKETSZ байт (константа определена в файле *arpa/nameser.h*).

anslen

Размер буфера *answer* (к примеру, PACKETSZ).

res_search возвращает размер полученного ответа либо значение –1 в случае ошибки.

---

res_query

```
int res_query(const char *dname,
              int class,
              int type,
              u_char *answer,
              int anslen)
```

res_query – одна из функций «среднего уровня». Она выполняет всю поисковую работу для доменного имени: создает сообщение-запрос с помощью вызова функции *res_mkquery*, посылает запрос с помощью вызова функции *res_send* и изучает ответное сообщение настолько, чтобы можно было понять, получен ли ответ на заданный вопрос. Во многих случаях *res_query* вызывается из функции *res_search*, которая поставляет доменные имена для поиска. Как можно догадаться, аргументы этих двух функций идентичны. *res_query* возвращает размер ответного сообщения либо заполняет переменную *h_errno* и возвращает значение –1 в случае наличия ошибки либо количества ответов, равного нулю.

res_mkquery

```
int res_mkquery(int op,
                const char *dname,
                int class,
                int type,
                const u_char *data,
                int datalen,
                const u_char *newrr,
                u_char *buf,
                int buflen)
```

Функция *res_mkquery* создает сообщение-запрос. Она заполняет все поля заголовка, производит упаковку доменного имени для раздела вопроса, а также заполняет прочие поля раздела вопроса.

Аргументы *dname*, *class* и *type* имеют такой же смысл, как для функций *res_search* и *res_query*. Остальные аргументы:

op

«Операция», которая должна быть выполнена. Как правило, имеет значение QUERY, но может принимать значение IQUERY (инверсный запрос). Однако, как мы уже говорили, IQUERY используется крайне редко. BIND версии 4.9.4 и более поздних по умолчанию вообще не поддерживают IQUERY.

data

Буфер, содержащий данные для обратных запросов. Является NULL-указателем в случаях, когда *op* имеет значение QUERY.

datalen

Размер буфера *data*. Если буфер *data* является NULL-указателем, *datalen* принимает нулевое значение.

newrr

Буфер, который используется кодом динамического обновления (подробнее в главе 10 «Дополнительные возможности»). Если только вы не исследуете возможности этого механизма, буфер является NULL-указателем.

buf

Буфер, в котором *res_mkquery* создает сообщение-запрос. Буфер должен иметь размер PACKETSZ или больший, как и в случае с буфером ответа в функциях *res_search* и *res_query*.

buflen

Размер буфера *buf* (например, PACKETSZ).

res_mkquery возвращает размер сообщения-запроса либо значение –1, если произошла ошибка.

res_send

```
int res_send(const u_char *msg,
             int msglen,
             u_char *answer,
             int anslen)
```

Функция *res_send* реализует алгоритм повторения попытки. Она посылает сообщение-запрос, *msg*, в UDP-дейтаграмме либо через TCP-соединение. Ответное сообщение записывается в буфер *answer*. Это единственная из всех функций библиотеки клиента, использующая черную магию (если только вы не из тех, кто все знает о связанных сокетах дейтаграмм – *connected datagramm sockets*). Эти аргументы уже встречались читателям в описаниях других функций:

msg

Буфер, содержащий сообщение-запрос DNS.

msglen

Размер сообщения.

answer

Буфер, в который следует записывать ответное сообщение DNS.

anslen

Размер ответного сообщения.

res_send возвращает размер ответного сообщения либо значение –1, если произошла ошибка. Если функция вернула значение –1, а переменная *errno* установлена в значение ECONNREFUSED, это означает, что на целевом узле не работает DNS-сервер.

Проверять значение переменной *errno* на равенство ECONNREFUSED можно после вызова функции *res_search* или *res_query*. (*res_search* вызывает *res_query*, а *res_query* вызывает *res_send*.) Если необходимо проверить значение *errno* после вызова *res_query*, следует обнулить *errno до вызова*. Таким образом, можно будет наверняка сказать, что именно последний вызов *res_send* изменил значение *errno*. При вызове *res_search* нет необходимости обнулять *errno*, поскольку *res_search* делает это самостоятельно перед вызовом *res_query*.

res_init

```
int res_init(void)
```

res_init читает файл *resolv.conf* и инициализирует структуру *_res* (о которой чуть позже). Все упомянутые функции вызывают *res_init*, если определяют, что эта функция еще не выполнялась. Автор программы может сам вызвать эту функцию. Такая возможность полезна, если существует необходимость изменить стандартные значения перед вызовом первой прикладной функции библиотеки DNS-клиента. Если в

файле *resolv.conf* присутствуют строки, которых *res_init* не понимает, эти строки игнорируются. *res_init* всегда возвращает нулевое значение, даже если страницы руководства резервируют за этой функцией право возвращать значение –1.

---

herror и h_errno

```
extern int h_errno;
int herror(const char *s)
```

herror – это функция, аналогичная функции *perror*, с той разницей, что она печатает строки, основанные на значении внешней переменной *h_errno*, а не *errno*. Единственный аргумент:

s Строка, которая используется для идентификации сообщения об ошибке. Если присутствует аргумент *s*, он выдается на печать первым и отделяется от диагностического сообщения символом «:».

Существуют следующие значения переменной *h_errno*:

HOST_NOT_FOUND

Доменное имя не существует. Код возврата в ответе DNS-сервера – NXDOMAIN.

TRY_AGAIN

DNS-сервер не запущен либо DNS-сервер вернул SERVFAIL.

NO_RECOVERY

Доменное имя не может быть упаковано, поскольку является некорректным (например, имя, в котором не хватает метки *.movie.edu*), либо DNS-сервер вернул FORMERR, NOTIMP или REFUSED.

NO_DATA

Доменное имя существует, но нет данных указанного типа.

NETDB_INTERNAL

Ошибка библиотеки, не связанная с сетевыми службами и DNS. Описание проблемы содержится в переменной *errno*.

Структура _res

Каждая функция библиотеки клиента (имя которой содержит префикс *res_*) использует общую структуру данных, которая называется *_res*. Поведение функций DNS-клиента можно менять, изменяя значения в структуре *_res*. Если необходимо изменить число повторных попыток при посылке запроса функцией *res_send*, следует изменить значение поля *retry*. Если необходимо отключить алгоритм поиска, следует сбросить бит RES_DNSRCH в маске *options*. Крайне важная структура *_res* определена в файле *resolv.h*:

```
struct __res_state {
```

```
    int      retrans;   /* временной интервал для переключений */
    int      retry;     /* число повторных попыток */
    u_long   options;   /* флаги настроек - см. ниже. */
    int      nscount;   /* число DNS-серверов */
    struct sockaddr_in
          nsaddr_list[MAXNS];   /* адрес DNS-сервера */
#define nsaddr nsaddr_list[0]      /* в целях обратной совместимости */
    u_short id;                    /* идентификатор текущего пакета */
    char     *dnsrch[MAXDNSRCH+1]; /* составляющие домена для поиска */
    char     defdname[MAXDNAME];   /* домен по умолчанию */
    u_long   pfcode;               /* флаги RES_PRF_ - см. ниже. */
    unsigned ndots:4;              /* пороговое значение для начального
                                      запроса */
    unsigned nsort:4;              /* число элементов в sort_list[] */
    char     unused[3];
    struct {
           struct in_addr  addr;   /* адрес для сортировки */
           u_int32_t       mask;
    } sort_list[MAXRESOLVSORT];
};
```

Поле *options* является обычной битовой маской различных настроек. Чтобы включить определенный режим, следует установить соответствующий бит в поле *options*. Битовые маски для каждой настройки определены в *resolv.h*; а вот и сами настройки:

RES_INIT

Если бит установлен, произошел вызов *res_init*.

RES_DEBUG

При установленном бите происходит печать отладочных сообщений клиента, если клиент был собран со включенным режимом DEBUG. По умолчанию бит сброшен.

RES_AAONLY

Установленный бит предписывает получение ответа от авторитативного DNS-сервера, но не из кэша. К сожалению, для этого флага отсутствует реализация, поскольку возможность была бы востребована. Учитывая архитектуру DNS-клиента BIND, можно сделать вывод, что эта возможность должна быть реализована (но не реализована) в DNS-сервере.

RES_PRIMARY

Посылать запрос только первичному DNS-мастер-серверу (реализация опять же отсутствует).

RES_USEVC

Если установить данный бит, клиент будет делать запросы через виртуальное соединение (TCP), а не путем посылки UDP-дейтаграмм. Как можно догадаться, установление и уничтожение TCP-

соединения отрицательно влияют на производительность. Бит по умолчанию сброшен.

RES_STAYOPEN

При посылке запросов через TCP-соединение установка этого бита приводит к тому, что соединение не разрывается, и его можно использовать для посылки других запросов удаленному DNS-серверу. В противном случае соединение уничтожается после получения ответа на запрос. Бит по умолчанию сброшен.

RES_IGNTC

Если в ответном сообщении DNS-сервера установлен бит усечения, стандартным для клиента поведением становится повторение попыток с использованием TCP-соединений. При установленном бите *RES_IGNTC* бит усечения в ответном сообщении игнорируется и запрос не повторяется с использованием TCP. По умолчанию бит сброшен.

RES_RECURSE

По умолчанию клиент BIND посылает рекурсивные запросы. Сброс этого бита сбрасывает бит «запроса рекурсии» в сообщении-запросе. По умолчанию бит установлен.

RES_DEFNAMES

По умолчанию клиент BIND добавляет локальное доменное имя к любому доменному имени, в котором отсутствует хотя бы одна точка. Сброс этого бита выключает этот механизм. По умолчанию бит установлен.

RES_DNSRCH

По умолчанию клиент BIND добавляет элементы списка поиска к доменному имени, которое не заканчивается точкой. Сброс этого бита отключает функциональность списка поиска. По умолчанию бит установлен.

RES_INSECURE1

По умолчанию клиент BIND версии 4.9.3 и более поздних игнорирует ответы от DNS-серверов, которым не посылались запросы. Установка этого бита отключает данную проверку. По умолчанию бит сброшен (то есть проверка производится).

RES_INSECURE2

По умолчанию клиент BIND версии 4.9.3 и более поздних игнорирует ответные сообщения, в которых раздел вопроса не соответствует разделу вопроса исходного запроса. Установка этого бита отключает данную проверку. По умолчанию бит сброшен (то есть проверка производится).

RES_NOALIASES

По умолчанию клиент BIND использует псевдонимы, определенные в файле, на который указывает переменная среды HOSTALIASES. Установка этого бита отключает использование HOSTALIASES в клиентах BIND 4.9.3 и более поздних версий. Клиенты более старых версий не предоставляли возможности отключить использование псевдонимов. По умолчанию бит сброшен.

RES_USE_INET6

Предписание клиенту возвращать адрес в формате IPv6 (в дополнение к адресу в формате IPv4) функции *gethostbyname*.

RES_ROTATE

В обычной ситуации при посылке повторяющихся запросов клиент использует сначала первый из DNS-серверов из файла *resolv.conf*. Если бит RES_ROTATE установлен, клиент BIND версии 8.2 и более поздних посылает первый запрос первому DNS-серверу из *resolv.conf*, второй запрос – второму DNS-серверу и т. д. Более подробная информация содержится в главе 6, в описании инструкции *options rotate*. По умолчанию изменения порядка опроса DNS-серверов не происходит.

RES_NOCHECKNAME

Начиная с BIND версии 4.9.4, клиент проверяет доменное имя в ответе на предмет соответствия правилам именования, описанным в главе 4 «Установка BIND». Клиенты BIND 8.2 предоставляют возможность отключения механизма проверки. По умолчанию бит сброшен (то есть проверка производится).

RES_KEEPTSIG

Данный бит предписывает клиенту BIND версии 8.2 или более поздней не удалять TSIG-запись из подписанного сообщения DNS. В этом случае приложение, использующее клиент, имеет возможность изучить подпись.

RES_BLAST

«Бомбить» все рекурсивные серверы посылкой параллельных запросов. Реализация механизма отсутствует.

RES_DEFAULT

Это не самостоятельный флаг, а комбинация флагов RES_RECURSE, RES_DEFNAMES и RES_DNSRCH, которые по умолчанию установлены. В обычной ситуации нет необходимости явным образом устанавливать *RES_DEFAULT*; эта опция устанавливается автоматически при вызове *res_init*.

Функции библиотеки DNS–сервера

Библиотека DNS-сервера содержит функции, которые необходимы для разбора ответных сообщений. Необходимо включать в приложение следующие заголовочные файлы:

```
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/nameser.h>
#include <resolv.h>
```

Ниже приводятся описания функций библиотеки DNS-сервера.

---

ns_initparse

```
int ns_initparse(const u_char *msg,
                 int msglen,
                 ns_msg *handle)
```

ns_initparse – первая функция, которую необходимо вызвать, прежде чем можно будет воспользоваться другими функциями библиотеки DNS-сервера. *ns_initparse* инициализирует структуру данных, на которую ссылается *handle* и которая является параметром, передаваемым другим функциям. Аргументы функции:

msg

Указатель на начало буфера ответного сообщения.

msglen

Размер буфера сообщения.

handle

Указатель на структуру данных, инициализируемую *ns_initparse*.

ns_initparse возвращает нуль при успешном завершении и значение –1 в случае невозможности произвести разбор буфера сообщения.

---

ns_msg_base, ns_msg_end и ns_msg_size

```
const u_char *ns_msg_base(ns_msg handle)
const u_char *ns_msg_end(ns_msg handle)
int ns_msg_size(ns_msg handle)
```

Перечисленные функции возвращают указатель на начало сообщения, указатель на конец сообщения и размер сообщения. Они возвращают данные, которые были переданы функции *ns_initparse*. Единственный аргумент:

handle

Структура данных, инициализируемая функцией *ns_initparse*.

ns_msg_id

```
u_int16_t ns_msg_id(ns_msg handle)
```

ns_msg_id возвращает идентификатор из (описанного ранее) раздела заголовка ответного сообщения. Единственный аргумент:

handle

Структура данных, инициализируемая функцией *ns_initparse*.

ns_msg_get_flag

```
u_int16_t ns_msg_get_flag(ns_msg handle, ns_flag flag)
```

ns_msg_get_flag возвращает поля-флаги из раздела заголовка ответного сообщения. Аргументы функции:

handle

Структура данных, инициализируемая функцией *ns_initparse*.

flag

Перечислимый тип, содержащий следующий значения:

```
ns_f_qr      /* Вопрос/Ответ */
ns_f_opcode  /* Код операции */
ns_f_aa      /* Авторитативный ответ */
ns_f_tc      /* Произошло усечение */
ns_f_rd      /* Запрос рекурсии */
ns_f_ra      /* Рекурсия доступна */
ns_f_z       /* Нулевое значение */
ns_f_ad      /* Достоверные данные (DNSSEC) */
ns_f_cd      /* Проверка отключена (DNSSEC) */
ns_f_rcode   /* Код ответа */
ns_f_max
```

ns_msg_count

```
u_int16_t ns_msg_count(ns_msg handle, ns_sect section)
```

ns_msg_count возвращает счетчик из раздела заголовка ответного сообщения. Аргументы функции:

handle

Структура данных, инициализируемая функцией *ns_initparse*.

section

Перечислимый тип, содержащий следующие значения:

```
ns_s_qd   /* Запрос: раздел вопроса */
ns_s_zn   /* Обновление: раздел зоны */
ns_s_an   /* Запрос: раздел ответа */
```

```
ns_s_pr   /* Обновление: раздел предварительных требований */
ns_s_ns   /* Запрос: раздел DNS-сервера */
ns_s_ud   /* Обновление: раздел обновления */
ns_s_ar   /* Запрос|Обновление: раздел дополнительных записей */
```

---

ns_parserr

```
int ns_parserr(ns_msg *handle,
               ns_sect section,
               int rrnum,
               ns_rr *rr)
```

ns_parserr извлекает информацию о записи ответа и сохраняет ее в структуре *rr*, которая передается в качестве параметра другим функциям библиотеки DNS-сервера. Аргументы функции:

handle

Указатель на структуру данных, инициализируемую функцией *ns_initparse*.

section

Описание *section* приводится в описании функции *ns_msg_count*.

rrnum

Число RR-записей в данном разделе. Нумерация начинается с нуля. Функция *ns_msg_count* возвращает число RR-записей в данном разделе.

rr Указатель на структуру данных, которую следует инициализировать.

ns_parserr возвращает нуль при успешном завершении и значение –1, если буфер ответного сообщения не может быть разобран.

функции ns_rr

```
char *ns_rr_name(ns_rr rr)
u_int16_t ns_rr_type(ns_rr rr)
u_int16_t ns_rr_class(ns_rr rr)
u_int32_t ns_rr_ttl(ns_rr rr)
u_int16_t ns_rr_rdlen(ns_rr rr)
const u_char *ns_rr_rdata(ns_rr rr)
```

Перечисленные функции возвращают значения отдельных полей ответной записи. Единственный аргумент:

rr Структура данных, инициализируемая функцией *ns_parserr*.

ns_name_compress

```
int ns_name_compress(const char *exp_dn,
                     u_char *comp_dn,
                     size_t length,
                     const u_char **dnptrs,
                     const u_char **lastdnptr)
```

ns_name_compress производит упаковку доменного имени. В обычной ситуации приложение не вызывает эту функцию явно – это делает функция *res_mkquery*. Но если существует необходимость произвести упаковку имени, то эта функция подойдет наилучшим образом. Аргументы функции:

exp_dn

Передаваемое «обычное» доменное имя; то есть нормальная строка, завершаемая нулевым байтом, которая содержит абсолютное доменное имя.

comp_dn

Буфер, в который будет записано упакованное доменное имя.

length

Размер буфера *comp_dn*.

dnptrs

Массив указателей на уже упакованные доменные имена. *dnptrs[0]* указывает на начало сообщения; список заканчивается NULL-указателем. После того как элемент *dnptrs[0]* инициализирован указателем на начало сообщения, а *dnptrs[1]* – NULL-указателем, *dn_comp* обновляет список при каждом вызове.

lastdnptr

Указатель на конец массива *dnptrs*. Эта информация необходима функции *ns_name_compress*, чтобы не выйти за пределы массива.

Если вы намереваетесь воспользоваться этой функцией, изучите ее использование в исходных текстах пакета BIND: в файле *src/lib/resolv/res_mkquery.c* (BIND 8) или в файле *res/res_mkquery.c* (BIND 4). Часто гораздо проще изучить применение функции на конкретном примере, чем понять, как ее следует применять из объяснения. *ns_name_compress* возвращает размер сжатого имени или значение –1 в случае, если произошла ошибка.

ns_name_uncompress

```
int ns_name_uncompress(const u_char *msg,
                       const u_char *eomorig,
                       const u_char *comp_dn,
                       char *exp_dn,
                       size_t length)
```

ns_name_uncompress производит распаковку «упакованного» доменного имени. Эту функцию можно использовать при разборе ответных сообщений DNS-сервера, как это сделано в *check_soa*, программе на языке C, которую мы приводим ниже. Аргументы функции:

msg

Указатель на начало ответного сообщения.

eomorig

Указатель на первый байт, не входящий в сообщение. Используется, чтобы предотвратить выход *ns_name_uncompress* за пределы конца сообщения.

comp_dn

Указатель на упакованное доменное имя в пределах сообщения.

exp_dn

Буфер, в который функция *ns_name_uncompress* записывает распакованное имя. Должен иметь размер в MAXDNAME символов.

length

Размер буфера *exp_dn*.

ns_name_uncompress возвращает размер упакованного имени либо значение –1, если произошла ошибка. Возникает резонный вопрос, почему *ns_name_uncompress* возвращает размер *упакованного* имени, а не размер *распакованного*? Дело в том, что при вызове *ns_name_uncompress* происходит разбор сообщения DNS, и необходимо знать, сколько места в сообщении занимало упакованное имя, чтобы иметь возможность пропустить его.

---

ns_name_skip

```
int ns_name_skip(const u_char **ptrptr, const u_char *eom)
```

Функция *ns_name_skip* схожа с *ns_name_uncompress*, но вместо распаковки имени она просто пропускает его. Аргументы функции:

ptrptr

Указатель на указатель на имя, которое следует пропустить. Исходный указатель увеличивается на длину имени.

eom

Указатель на первый байт после конца сообщения. Используется, чтобы предотвратить выход *ns_name_skip* за пределы конца сообщения.

ns_name_skip возвращает нуль при успешном завершении либо значение –1, если имя не может быть распаковано.

ns_get16 и ns_put16

```
u_int ns_get16(const u_char *cp)
void  ns_put16(u_int s, u_char *cp)
```

Сообщения DNS содержат поля, имеющие тип беззнакового короткого целого числа (тип, класс, длина данных, к примеру). *ns_get16* возвращает 16-битное целое, расположенное по адресу *cp*, а *ns_put16* присваивает 16-битное значение *s* ячейке памяти по адресу *cp*.

ns_get32 и ns_put32

```
u_long ns_get32(const u_char *cp)
void   ns_put32(u_long l, u_char *cp)
```

Эти функции работают так же, как их 16-битные аналоги, только с 32-битными целыми числами. Поле TTL (время жизни) ресурса является 32-битным целым числом.

Разбор ответов DNS

Самый простой способ научиться разбирать сообщения DNS – прочесть код, который этим занимается. Предполагая, что у читателей есть исходные тексты пакета BIND, мы отсылаем их к лучшему источнику информации, файлу *src/lib/resolv/res_debug.c* (BIND 8) или *res/res_debug.c* (BIND 4). (Если очень хочется использовать BIND 9, придется прочесть почти 3000 строк файла *lib/dns/message.c*.) Файл *res_debug.c* содержит функцию *fp_query* (*res_pquery* в BIND 8.2 и более поздних версиях), которая печатает сообщения DNS в отладочном выводе DNS-сервера. Наша программа-пример берет начало в коде из этого файла.

Не всегда существует необходимость в ручном разборе ответов DNS. В качестве «обходного» способа разбора ответов можно предложить вызов функции *p_query*, которая вызывает функцию *fp_query* для печати сообщения DNS. Затем можно использовать Perl или *awk* для извлечения нужных данных. Известно, что Крикет ходил по этому пути наименьшего сопротивления.

Пример программы: check_soa

Теперь рассмотрим С-программу, решающую ту самую задачу, которую мы ранее решили путем написания сценария командного интерпретатора.

Итак, вот необходимые заголовочные файлы, объявления внешних переменных и объявления функций. Обратите внимание, что мы использовали как переменную *h_errno* (для функций клиента), так и *errno*. Мы ограничиваем возможности программы 20 DNS-серверами. Редко

когда можно встретить зону, для которой существует более десяти DNS-серверов, поэтому ограничения в 20 серверов должно хватить на все случаи жизни:

```
/*********************************************************
* check_soa - Получение SOA-записи с каждого DNS-сервера  *
*      определенной зоны и печать порядкового номера.     *
*                                                         *
* применение: check_soa зона                              *
*                                                         *
* Обрабатываются и отображаются следующие ошибки:         *
*     o Отсутствует адрес для сервера.                    *
*     o Отсутствует сервер на узле.                       *
*     o Сервер не ответил.                                *
*     o Сервер не является авторитативным для зоны.       *
*     o Ответ содержал код ошибки.                        *
*     o Ответ содержал более одного результата            *
*     o Ответ не содержал SOA-записи.                     *
*     o Ошибка при распаковке доменного имени.            *
*********************************************************/

/* Различные файлы заголовков */
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <errno.h>
#include <arpa/nameser.h>
#include <resolv.h>

/* Переменные-контейнеры для кодов ошибок */
extern int h_errno;  /* ошибки клиента */
extern int errno;    /* общесистемные ошибки */

/* Наши функции; код приводится далее в главе */
void nsError();           /* получение ошибок клиента*/
void findNameServers();  /* поиск DNS-серверов для зоны */
void addNameServers();   /* добавление DNS-серверов к списку */
void queryNameServers(); /* получение SOA-записей */
void returnCodeError();  /* вывод ошибок, содержащихся в ответных сообщениях */

/* Максимальное число проверяемых DNS-серверов */
#define MAX_NS 20
```

Основная функция получается небольшой. Массив строковых указателей, *nsList*, будет хранить имена DNS-серверов для зоны. Мы вызываем функцию клиента *res_init* для инициализации структуры *_res*. Нет необходимости вызывать *res_init* явно, поскольку эта функция будет выполнена первой же функцией клиента, использующей структуру *_res*. Тем не менее, если нам понадобится изменить значение каких-либо полей *_res* перед вызовом первой из таких функций, эти изменения следует вносить после вызова *res_init*. Далее программа вызывает

функцию *findNameServers*, которая находит все DNS-серверы для зоны, указанной в аргументе *argv[1]*; имена серверов записываются в массив *nsList*. И наконец, программа вызывает функцию *queryNameServers*, которая опрашивает все DNS-серверы из списка *nsList* на предмет получения SOA-записи для зоны:

```
main(argc, argv)
int argc;
char *argv[];
{
    char *nsList[MAX_NS];  /* список DNS-серверов */
    int  nsNum = 0;        /* число DNS-серверов в списке */

    /* проверка аргументов: допустим один и только один */
    if(argc != 2){
        (void) fprintf(stderr, "применение: %s зона\n", argv[0]);
        exit(1);
    }

    (void) res_init();

    /*
     * Поиск всех DNS-серверов для зоны.
     * Имена серверов заносятся в список nsList.
     */
    findNameServers(argv[1], nsList, &nsNum);

    /*
     * Запрос SOA-записи у каждого из DNS-серверов зоны.
     * Используются имена серверов из списка nsList.
     */
    queryNameServers(argv[1], nsList, nsNum);

    exit(0);
}
```

Ниже приводится реализация функции *findNameServers*. Эта функция запрашивает у локального DNS-сервера NS-записи для указанной зоны. Затем происходит вызов функции *addNameServers* с целью разбора ответных сообщений и сохранения в списке найденных DNS-серверов. Заголовочные файлы *arpa/nameser.h* и *resolv.h* содержат объявления, которые мы активно используем:

```
/*******************************************************
 *    findNameServers - поиск всех DNS-серверов для     *
 *    указанной зоны и сохранение имен в списке nsList. *
 *    nsNum содержит число серверов в массиве nsList.   *
 *******************************************************/
void
findNameServers(domain, nsList, nsNum)
char *domain;
char *nsList[];
int  *nsNum;
```

```
{
    union {
        HEADER hdr;              /* определяется в resolv.h */
        u_char buf[NS_PACKETSZ]; /* определяется в arpa/nameser.h */
    } response;                  /* буфер ответов */
    int responseLen;             /* длина буфера */

    ns_msg handle;  /* дескриптор для ответных сообщений */

    /*
     * Поиск NS-записей для указанного доменного имени.
     * Мы считаем, что представлено полное доменное имя, поэтому используем
     * res_query(). Если бы мы хотели воспользоваться алгоритмом поиска
     * клиента, то вызывали бы res_search().
     */
    if((responseLen =
           res_query(domain,      /* зона, которая нам интересна */
                     ns_c_in,     /* записи класса Интернет      */
                     ns_t_ns,     /* поиск записей DNS-серверов  */
                     (u_char *)&response,      /*буфер ответа    */
                     sizeof(response)))        /*размер буфера   */
                                        < 0){  /*В случае получения      */
        nsError(h_errno, domain);           /* отрицательного значения   */
                                            /* сообщить об ошибке        */
        exit(1);                            /* и завершить работу        */
    }

    /*
     * Инициализировать дескриптор данным ответом. Дескриптор будет
     * использован позже для извлечения информации, полученной в ответе.
     */
    if (ns_initparse(response.buf, responseLen, &handle) < 0) {
        fprintf(stderr, "ns_initparse: %s\n", strerror(errno));
        return;
    }

    /*
     * Создать список DNS-серверов, перечисленных в ответе. NS-записи могут
     * содержаться в разделе ответа и/или в разделе авторитативности,
     * в зависимости от используемой реализации  DNS. Исследовать оба раздела.
     * Адреса DNS-серверов могут содержаться в разделе дополнительных
     * записей, но мы проигнорируем их, поскольку намного проще позже вызвать
     * функцию gethostbyname(), чем разбирать и хранить адреса сейчас.
     */

    /*
     * Добавить DNS-серверы из раздела ответа.
     */
    addNameServers(nsList, nsNum, handle, ns_s_an);

    /*
     * Добавить DNS-серверы из раздела авторитативности.
     */
    addNameServers(nsList, nsNum, handle, ns_s_ns);
}
```

```
/*****************************************************
* addNameServers - Исследовать RR-записи в разделе  *
* Сохранить имена всех DNS-серверов.                *
 *****************************************************/

void
addNameServers(nsList, nsNum, handle, section)
char *nsList[];
int  *nsNum;
ns_msg handle;
ns_sect section;
{
    int rrnum;  /* число RR-записей */
    ns_rr rr;   /* распакованная RR-запись */

    int i, dup; /* прочие переменные */

    /*
     * Для всех RR-записей в данном разделе
     */
    for(rrnum = 0; rrnum < ns_msg_count(handle, section); rrnum++)
    {
        /*
         * Расширить номер записи rrnum в rr.
         */
        if (ns_parserr(&handle, section, rrnum, &rr)) {
            fprintf(stderr, "ns_parserr: %s\n", strerror(errno));
        }

        /*
         * Если тип записи - NS, сохранить имя DNS-сервера.
         */
        if (ns_rr_type(rr) == ns_t_ns) {

            /*
             * Выделить память для хранения имени.  Как и полагается хорошим
            * программистам, мы проверяем результат выполнения функции malloc
             * и прекращаем работу, если память не может быть выделена.
             */
            nsList[*nsNum] = (char *) malloc (MAXDNAME);
            if(nsList[*nsNum] == NULL){
                (void) fprintf(stderr, "невозможно выполнить malloc\n");
                exit(1);
            }

            /* Распаковка доменного имени DNS-сервера */
            if (ns_name_uncompress(
                        ns_msg_base(handle),/* начало сообщения */
                        ns_msg_end(handle), /* конец сообщения */
                       ns_rr_rdata(rr),    /* Позиция в пределах сообщения */
                        nsList[*nsNum],     /* Результат                    */
                        MAXDNAME)           /* Размер буфера  nsList        */
```

```
                                < 0) {   /* Отрицательное значение: ошибка  */
                    (void) fprintf(stderr, "ошибка при выполнении
                                        ns_name_uncompress\n");
                     exit(1);
                }

                /*
                 * Добавить только что полученное доменное имя
                 * в список DNS-серверов, если это имя в списке отсутствует.
                 * В противном случае имя следует проигнорировать.
                 */
                for(i = 0, dup=0; (i < *nsNum) && !dup; i++)
                    dup = !strcasecmp(nsList[i], nsList[*nsNum]);
                if(dup)
                    free(nsList[*nsNum]);
                else
                    (*nsNum)++;
            }
        }
    }
```

Обратите внимание, что мы явным образом не проверяем получение нулевого числа записей DNS-серверов. Дело в том, что функция *res_query* считает такой результат ошибкой; она возвращает значение –1 и устанавливает значение *herrno* в *NO_DATA*. Если функция *res_query* возвращает –1, мы вызываем собственную функцию *nsError*, которая печатает сообщение об ошибке из переменной *h_errno*, не используя *herror*. Функция *herror* не подходит для нашей программы, поскольку ее сообщения предполагают, что производился поиск адресных данных (то есть если *h_errno* принимает значение *NO_DATA*, сообщение об ошибке выглядит так: «No address associated with name» – «Нет адреса, связанного с именем»).

Следующая функция посылает каждому из найденных DNS-серверов запрос SOA-записи. В этой функции мы изменяем значения нескольких полей структуры *_res*. Изменяя поле *nsaddr_list*, мы указываем, какому DNS-серверу следует посылать (*res_send*) запросы. Мы отключаем список поиска сбрасывая соответствующие биты в поле *options* – все доменные имена, с которыми работает эта программа, являются абсолютными:

```
/*************************************************************************
* queryNameServers - Послать запрос SOA-записи каждому из DNS-серверов, *
*     перечисленных в списке nsList.  Сообщать обо всех полученных      *
*     ошибках (например, об отсутствии DNS-сервера либо об отсутствии  *
*     должной авторитативности у автора ответа). Если ошибок нет,       *
*     напечатать порядковый номер для зоны.                             *
*************************************************************************/
void
queryNameServers(domain, nsList, nsNum)
char *domain;
```

```
char *nsList[];
int nsNum;
{
    union {
        HEADER hdr;                /* определяется в resolv.h */
        u_char buf[NS_PACKETSZ];   /* определяется в arpa/nameser.h */
    } query, response;             /* буферы запроса и ответа */
    int responseLen, queryLen;     /* длина буфера */

    u_char    *cp;    /* указатель на символьную строку */
                      /* разобранного DNS-сообщения */

    struct in_addr saveNsAddr[MAXNS];  /* адреса из _res */
    int nsCount;          /* счетчик адресов из res */
    struct hostent *host; /* структура для поиска по ns addr */
    int i;                /* переменная-счетчик */

    ns_msg handle;  /* дескриптор для ответного сообщения */
    ns_rr rr;       /* распакованная RR-запись */

    /*
     * Сохраняем список DNS-серверов из _res, он еще понадобится позже.
     */
    nsCount = _res.nscount;
    for(i = 0; i < nsCount; i++)
      saveNsAddr[i] = _res.nsaddr_list[i].sin_addr;

    /*
     * Выключаем поисковый алгоритм и добавление локального доменного
     * имени перед вызовом gethostbyname(); доменные имена DNS-серверов
     * могут быть только абсолютными.
     */
    _res.options &= ~(RES_DNSRCH | RES_DEFNAMES);

    /*
     * Запросить SOA-запись у каждого DNS-сервера зоны.
     */
    for(nsNum-- ; nsNum >= 0; nsNum--){

        /*
         * Сначала необходимо получить IP-адрес каждого DNS-сервера. Пока что
         * у нас есть только доменные имена. Мы используем gethostbyname()
         * для получения адресов и не будем изощряться. Но сначала
         * необходимо восстановить определенные значения в _res, поскольку
         * содержимое res влияет на работу gethostbyname(). (Мы изменили
         * _res в последнем проходе цикла.)
         *
         * Невозможно просто вызвать res_init() и восстановить эти значения,
         * поскольку некоторые из полей _res инициализируются при объявлении
         * переменной, а не при вызове res_init().
         */
        _res.options |= RES_RECURSE;  /* рекурсия включена (по умолчанию) */
        _res.retry = 4;               /* 4 повтора (по умолчанию)    */
        _res.nscount = nsCount;       /* исходные DNS-серверы */
         for(i = 0; i < nsCount; i++)
```

```
            _res.nsaddr_list[i].sin_addr = saveNsAddr[i];

        /* Поиск адреса DNS-сервера */
        host = gethostbyname(nsList[nsNum]);
        if (host == NULL) {
            (void) fprintf(stderr,"Не существует адреса для %s\n",
                                            nsList[nsNum]);
            continue; /* nsNum, цикл for */
        }

/*
         * А теперь настало время веселиться. host содержит IP-адреса
         * DNS-сервера, который обрабатывается. Сохраним первый адрес узла
         * в структуре _res. Скоро мы будем искать SOA-запись...
         */
        (void) memcpy((void *)&_res.nsaddr_list[0].sin_addr,
           (void *)host->h_addr_list[0], (size_t)host->h_length);
        _res.nscount = 1;

        /*
         * Выключаем рекурсию. Нам не нужно, чтобы DNS-сервер обращался
         * к другим серверам, пытаясь найти SOA-запись; DNS-сервер должен быть
         * компетентен в искомом типе данных.
         */
        _res.options &= ~RES_RECURSE;

        /*
         * Сокращаем число повторных попыток. Мы можем перебрать несколько
         * серверов и потому не желаем ждать слишком долго ответа каждого
         * сервера в отдельности. Две попытки и один адрес для запроса
         * сократят время ожидания до 15 секунд в худшем случае.
         */
        _res.retry = 2;

        /*
         * Мы хотим получить код следующего ответа, поэтому должны создать
         * сообщение-запрос и самостоятельно послать его, не пользуясь
         * функцией res_query(). Если res_query() возвращает -1, то может
         * не существовать ответа, на который можно было бы взглянуть.
         *
         * Нет необходимости проверять, возвращает ли функция res_mkquery()
         * значение -1. Если должны возникнуть ошибки на этапе упаковки,
         * они возникли бы уже при вызове функции res_query() для данного
         * доменного имени, а этот вызов уже был сделан ранее.
         */
        queryLen = res_mkquery(
                    ns_o_query,      /* обычный запрос  */
                    domain,          /* зона для поиска данных*/
                    ns_c_in,         /* Тип Интернет */
                    ns_t_soa,        /* поиск SOA-записи */
                    (u_char *)NULL,  /* всегда имеет значение NULL       */
                    0,               /* длина NULL        */
                    (u_char *)NULL,  /* всегда NULL       */
                    (u_char *)&query,/* буфер для запроса */
```

```
                sizeof(query));  /* размер буфера   */
/*
* Посылаем сообщение-запрос. Если на целевом узле не работает DNS-сервер,
 * функция res_send() возвращает значение -1 и устанавливает
 * значение errno в ECONNREFUSED. Во-первых, обнулим errno.
 */
errno = 0;
if((responseLen = res_send((u_char *)&query,/* запрос */
                          queryLen,        /* действительная длина */
                           (u_char *)&response,/* буфер  */
                           sizeof(response)))  /*размер буфера */
                                < 0){          /* ошибка  */
    if(errno == ECONNREFUSED) { /* нет DNS-сервера на узле */
        (void) fprintf(stderr,
            "Не найден DNS-сервер на узле %s\n",
            nsList[nsNum]);
    } else {                    /* все остальное: нет ответа */
        (void) fprintf(stderr,
            "%s не ответил\n",
            nsList[nsNum]);
    }
    continue; /* nsNum, цикл for */
}
/*
 * Инициализируем дескриптор этим ответом.  Дескриптор будет
 * позже использован для извлечения информации из ответа.
 */
if (ns_initparse(response.buf, responseLen, &handle) < 0) {
    fprintf(stderr, "ns_initparse: %s\n", strerror(errno));
    return;
}
/*
 * Если в ответе содержится ошибка, выдать соответствующее сообщение
 * и перейти к следующему серверу в списке.
 */
if(ns_msg_getflag(handle, ns_f_rcode) != ns_r_noerror){
    returnCodeError(ns_msg_getflag(handle, ns_f_rcode),
                                        nsList[nsNum]);
    continue; /* nsNum, цикл for */
}
/*
 * Получен ли авторитативный ответ? Проверяем бит компетентности
 * ответа.  Если DNS-сервер не является авторитативным,
 * сообщаем об этом и переходим к следующему серверу.
 */
if(!ns_msg_getflag(handle, ns_f_aa)){
    (void) fprintf(stderr,
        "%s не является авторитативным для зоны %s\n",
        nsList[nsNum], domain);
```

```
                continue; /* nsNum, цикл for */
            }

            /*
             * Ответ в ответном сообщении должен быть только один, в противном
             * случае следует сообщить об ошибке и перейти к следующему серверу.
             */
            if(ns_msg_count(handle, ns_s_an) != 1){
                (void) fprintf(stderr,
                    "%s: ожидался 1 ответ, получено %d\n",
                    nsList[nsNum], ns_msg_count(handle, ns_s_an));
                continue; /* nsNum, цикл for */
            }

            /*
             * Распаковать нулевую запись раздела ответа в rr.
             */
            if (ns_parserr(&handle, ns_s_an, 0, &rr)) {
                    if (errno != ENODEV){
                            fprintf(stderr, "ns_parserr: %s\n",
                                    strerror(errno));
                    }
            }

            /*
             * Мы запрашивали SOA-запись, если получены другие данные,
             * сообщить об ошибке и перейти к следующему серверу.
             */
            if (ns_rr_type(rr) != ns_t_soa) {
                 (void) fprintf(stderr,
                    "%s: ожидался ответ типа %d, получен ответ типа %d\n",
                    nsList[nsNum], ns_t_soa, ns_rr_type(rr));
                continue; /* nsNum for-loop */
            }

            /*
             * Сделать cp указателем на SOA-запись.
             */
            cp = (u_char *)ns_rr_rdata(rr);

            /*
             * Пропустить суффикс по умолчанию SOA и почтовый адрес, это
             * информация, которая нам не нужна. Оба поля являются
             * стандартными "упакованными именами."
             */
            ns_name_skip(&cp, ns_msg_end(handle));
            ns_name_skip(&cp, ns_msg_end(handle));

            /* cp является указателем на порядковый номер, который */
            /* и следует отобразить. */
            (void) printf("%s имеет порядковый номер %d\n",
                nsList[nsNum], ns_get32(cp));

        } /* конец for-цикла nsNum */
}
```

Обратите внимание, что мы делали рекурсивные запросы, когда вызывали функцию *gethostbyname*, и нерекурсивные при поиске SOA-записей. Функции *gethostbyname* может понадобиться опросить другие DNS-серверы при поиске адреса узла. Но мы не хотим, чтобы DNS-сервер посылал запросы другому серверу, если нам нужна SOA-запись – *предполагается*, в конце концов, что сервер является авторитативным для зоны. Если разрешить DNS-серверу искать SOA-запись на других серверах, это не позволит грамотно обработать возможные ошибки.

Следующие две функции занимаются печатью сообщений об ошибках:

```
/*****************************************************************
* nsError - Напечатать сообщение об ошибке из h_errno в случае  *
*     возникновения ошибки при поиске NS-записей. res_query()   *
*     отображает коды ответных сообщений DNS в более краткий    *
*     набор ошибок и помещает значение ошибки в h_errno.        *
*     Существует функция herror(), предназначенная для печати   *
*     строк из h_errno аналогично тому, как perror() печатает   *
*     значения errno. К сожалению, сообщения herror()           *
*     предполагают, что производился поиск адресных записей     *
*     для узла. В данной программе мы производим поиск         *
*     NS-записей для различных зон, поэтому нам необходим       *
*     собственный перечень сообщений об ошибках.                *
*****************************************************************/
void
nsError(error, domain)
int error;
char *domain;
{
    switch(error){
        case HOST_NOT_FOUND:
          (void) fprintf(stderr, "Неизвестная зона: %s\n", domain);
          break;
        case NO_DATA:
          (void) fprintf(stderr, "Отсутствуют NS-записи для %s\n", domain);
          break;
        case TRY_AGAIN:
          (void) fprintf(stderr, "Нет ответа на запрос NS-записей\n");
          break;
        default:
          (void) fprintf(stderr, "Непредвиденная ошибка \n");
          break;
    }
}

/*************************************************************************
 * returnCodeError - напечатать сообщение об ошибке,                     *
 * исходя из кода возврата,содержащегося в ответном сообщении.           *
*************************************************************************/
void
returnCodeError(rcode, nameserver)
ns_rcode rcode;
```

```
char *nameserver;
{
    (void) fprintf(stderr, "%s: ", nameserver);
    switch(rcode){
        case ns_r_formerr:
          (void) fprintf(stderr, "Ответ FORMERR\n");
          break;
        case ns_r_servfail:
          (void) fprintf(stderr, "Ответ SERVFAIL\n");
          break;
        case ns_r_nxdomain:
          (void) fprintf(stderr, "Ответ NXDOMAIN\n");
          break;
        case ns_r_notimpl:
          (void) fprintf(stderr, "Ответ NOTIMP\n");
          break;
        case ns_r_refused:
          (void) fprintf(stderr, "Ответ REFUSED\n");
          break;
        default:
          (void) fprintf(stderr, "непредусмотренный код возврата\n");
          break;
    }
}
```

Чтобы скомпилировать программу, используя клиент и функции DNS-сервера из библиотеки *libc*, следует выполнить:

```
% cc -o check_soa check_soa.c
```

Либо если BIND был собран из исходных текстов (процедура описана в приложении C «Сборка и установка BIND на Linux-системах»), можно использовать самые свежие версии заголовочных файлов и библиотеки клиента:

```
% cc -o check_soa -I/usr/local/src/bind/src/include \
check_soa.c /usr/local/src/bind/src/lib/libbind.a
```

Вот так выглядит вывод программы:

```
% check_soa mit.edu
BITSY.MIT.EDU has serial number 1995
W20NS.MIT.EDU has serial number 1995
STRAWB.MIT.EDU has serial number 1995
```

Если сравнить этот результат с результатом работы сценария командного интерпретатора, то окажется, что они одинаковы, с тем исключением, что вывод сценария отсортирован по именам серверов. Помимо этого, можно отметить, что программа на языке C работает гораздо быстрее.

Программирование на языке Perl при помощи модуля Net::DNS

Если использование интерпретатора для разбора вывода *nslookup* кажется слишком неудобным, а программирование на языке C – слишком сложным, попробуйте написать программу на языке Perl, используя разработанный Майклом Фером модуль Net::DNS. Пакет доступен по адресу *http://www.perl.com/CPAN-local/modules/by-module/Net/Net-DNS-0.12.tar.gz*.

Net::DNS работает с клиентами, сообщениями DNS и отдельными RR-записями как с объектами, и предоставляет методы для изменения или получения значений отдельных атрибутов объекта. Сначала мы рассмотрим типы существующих объектов, а затем представим Perl-вариант программы *check_soa*.

Объекты клиента

Прежде чем делать какие-либо запросы, необходимо создать объект клиента:

```
$res = new Net::DNS::Resolver;
```

Объекты клиента инициализируются на основе содержимого файла *resolv.conf*, но стандартные настройки можно изменить путем вызова соответствующих методов объекта. Многие из методов, описанных на страницах руководства по Net::DNS::Resolver, соответствуют полям и параметрам структуры *_res*, описанной ранее в этой главе. К примеру, если необходимо установить число повторений для каждого запроса, следует воспользоваться методом *$res->retry*:

```
$res->retry(2);
```

Чтобы сделать запрос, вызовите один из следующих методов:

```
$res->search
$res->query
$res->send
```

Эти методы работают аналогично библиотечным функциям *res_search*, *res_query* и *res_send*, описанным в разделе программирования на языке C, хотя принимают меньшее число аргументов. Обязательно указывать доменное имя, и по необходимости можно указывать тип записи и класс (по умолчанию запрашиваются A-записи класса IN). Эти методы возвращают объект типа Net::DNS::Packet, который описан в следующем разделе. Вот несколько примеров вызовов:

```
$packet = $res->search("terminator");
$packet = $res->query("movie.edu", "MX");
$packet = $res->send("version.bind", "TXT", "CH");
```

Объекты пакетов

Запросы клиента возвращают объекты Net::DNS::Packet, методы которых предоставляют доступ к разделам заголовка, вопроса, ответа, авторитативности, а также вторичным разделам сообщений DNS:

```
$header     = $packet->header;
@question   = $packet->question;
@answer     = $packet->answer;
@authority  = $packet->authority;
@additional = $packet->additional;
```

Объекты заголовков

Заголовки сообщений DNS возвращаются в виде объектов типа Net::DNS::Header. Методы, описанные на страницах руководства по Net::DNS::Header, соответствуют полям заголовка, описанным в документе RFC 1035 и структуре *HEADER*, используемой в C-программах. К примеру, если необходимо узнать, исходит ли полученный ответ от авторитативного DNS-сервера, следует вызвать метод *$header->aa*:

```
if ($header->aa) {
    print "ответ получен от авторитативного сервера\n";
} else {
    print "ответ получен не от авторитативного сервера\n";
}
```

Объекты вопросов

Раздел вопроса сообщения DNS возвращается в виде объекта типа Net::DNS::Question objects. Получить имя, тип и класс объекта вопроса можно с помощью следующих методов:

```
$question->qname
$question->qtype
$question->qclass
```

Объекты RR-записей

Разделы ответа, авторитативности, а также дополнительные возвращаются в виде списков объектов Net::DNS::RR. Имя, тип, класс и значение TTL для RR-объекта можно получить с помощью следующих методов:

```
$rr->name
$rr->type
$rr->class
$rr->ttl
```

Каждый тип записей является подклассом класса Net::DNS::RR и имеет специфичные для этого типа методы. Следующий пример пока-

зывает, как можно получить значения предпочтения и имя почтового ретранслятора из MX-записи:

```
$preference = $rr->preference;
$exchanger  = $rr->exchange;
```

Perl-вариант программы check_soa

Теперь, описав объекты, существующие в Net::DNS, посмотрим, как использовать эти объекты для создания полной программы. Мы переписали *check_soa* на языке Perl:

```
#!/usr/local/bin/perl -w

use Net::DNS;

#-----------------------------------------------------------------------
# Получить имя зоны из командной строки.
#-----------------------------------------------------------------------

die "Применение: check_soa зона\n" unless @ARGV == 1;
$domain = $ARGV[0];

#-----------------------------------------------------------------------
# Поиск всех DNS-серверов для зоны.
#-----------------------------------------------------------------------

$res = new Net::DNS::Resolver;

$res->defnames(0);
$res->retry(2);

$ns_req = $res->query($domain, "NS");
die "Не найдены DNS-серверы для $domain: ", $res->errorstring, "\n"
    unless defined($ns_req) and ($ns_req->header->ancount > 0);

@nameservers = grep { $_->type eq "NS" } $ns_req->answer;

#-----------------------------------------------------------------------
# Проверка SOA-записи каждого DNS-сервера.
#-----------------------------------------------------------------------

$| = 1;
$res->recurse(0);

foreach $nsrr (@nameservers) {

#-------------------------------------------------------------------
# Настроить клиент на использование данного DNS-сервера.
#-------------------------------------------------------------------

  $ns = $nsrr->nsdname;
  print "$ns ";

  unless ($res->nameservers($ns)) {
      warn ": невозможно найти адрес: ", $res->errorstring, "\n";
      next;
```

```
    }

    #-----------------------------------------------------------------
    # Получить SOA-запись.
    #-----------------------------------------------------------------

    $soa_req = $res->send($domain, "SOA");
    unless (defined($soa_req)) {
        warn ": ", $res->errorstring, "\n";
        next;
    }

    #-----------------------------------------------------------------
    # Является ли DNS-сервер авторитативным для зоны?
    #-----------------------------------------------------------------

    unless ($soa_req->header->aa) {
        warn "не является авторитативным для $domain\n";
        next;
    }

    #-----------------------------------------------------------------
    # Мы должны были получить ровно один ответ.
    #-----------------------------------------------------------------

    unless ($soa_req->header->ancount == 1) {
        warn ": ожидался 1 ответ, получено ",
              $soa_req->header->ancount, "\n";
        next;
    }

    #-----------------------------------------------------------------
    # Получена ли SOA-запись?
    #-----------------------------------------------------------------

     unless (($soa_req->answer)[0]->type eq "SOA") {
         warn ": ожидалась SOA-запись, получена ",
              ($soa_req->answer)[0]->type, "\n";
         next;
    }

    #-----------------------------------------------------------------
    # Печать порядкового номера.
    #-----------------------------------------------------------------

    print "имеет порядковый номер ", ($soa_req->answer)[0]->serial, "\n";
  }
```

Итак, изучив создание DNS-программ на языке командного интерпретатора, а также на языках Perl и C, читатели смогут написать свои собственные программы, используя язык, наиболее соответствующий ситуации.

- *Использование CNAME-записей*
- *Маски*
- *Ограничение MX-записей*
- *Коммутируемые соединения*
- *Имена и номера сетей*
- *Дополнительные RR-записи*
- *DNS и WINS*
- *DNS и Windows 2000*

16

Обо всем понемногу

И молвил Морж: «Пришла пора подумать о делах:
О башмаках и сургуче,
Капусте, королях,
И почему, как суп в котле,
Кипит вода в морях».

Настало время подводить итоги. Мы рассмотрели главные темы DNS и BIND, но существуют еще интересные моменты, которые мы не исследовали. Некоторые из них могут оказаться для читателей полезными на практике (например, объяснения, как подружить Windows 2000 и BIND), прочие – просто интересными. Мы не сможем с чистой совестью отпустить читателей в мир, не завершив их обучение!

Использование CNAME–записей

Мы рассказывали о CNAME-записях в главе 4 «Установка BIND». Но мы рассказали о CNAME-записях не все, оставив кое-что для этой главы. При установке первых DNS-серверов нет смысла задумываться о тонкостях, связанных с волшебной записью типа CNAME. Возможно, вы не осознавали, что тема шире, чем можно подумать, возможно, вам просто было все равно. Факты, приводимые далее, интересны или загадочны – зависит от того, как вы будете их воспринимать.

CNAME–записи для внутренних узлов

Если вам случалось сталкиваться с необходимостью переименования зоны в связи с реорганизациями в компании, то вы, возможно, задумывались о создании единственной CNAME-записи, которая служила

бы мостом между старым доменным именем зоны и новым. К примеру, если бы зона *fx.movie.edu* была переименована в *magic.movie.edu*, очень соблазнительно было бы создать CNAME-запись с целью отображения всех старых доменных имен в новые:

```
fx.movie.edu.  IN  CNAME  magic.movie.edu.
```

После этого мы ожидали бы, что поиск для имени *empire.fx.movie.edu* приводил бы к поиску для *empire.magic.movie.edu*. К сожалению, этот способ не работает: *невозможно* связать CNAME-запись с именем, вроде *fx.movie.edu*, если это имя входит также в записи других типов. Вспомним, что для *fx.movie.edu* существует SOA-запись и NS-записи, поэтому добавление CNAME-записи нарушает правило, которое гласит, что доменное имя может быть псевдонимом либо каноническим именем, но не тем и другим одновременно.

Если используется BIND 9, можно воспользоваться великолепной новой штуковиной – записью DNAME (рассмотренной в главе 10 «Дополнительные возможности») для создания отображения старых доменных имен в новые:

```
fx.movie.edu.  IN  DNAME  magic.movie.edu.
```

Запись DNAME может сосуществовать с записями другого типа для *fx.movie.edu* – SOA и NS, которые гарантированно существуют, но *невозможно* создать другие доменные имена, заканчивающиеся на *fx.movie.edu*. Эта запись будет «синтезировать» CNAME-записи для доменных имен *fx.movie.edu*, и создавать отображение в *magic.movie.edu* при поиске для имен из *fx.movie.edu*.

Если BIND версии 9 не используется, придется создавать псевдонимы по старинке – по одной CNAME-записи на каждое доменное имя зоны:

```
empire.fx.movie.edu.       IN  CNAME  empire.magic.movie.edu.
bladerunner.fx.movie.edu.  IN  CNAME  bladerunner.magic.movie.edu.
```

Если поддомен не делегирован, то есть не имеет SOA- и NS-записей, так же можно создать псевдоним для *fx.movie.edu*. Он будет работать только для доменного имени *fx.movie.edu*, но не для других доменных имен зоны *fx.movie.edu*.

В идеале инструмент, используемый для управления файлами данных зоны, поможет в создании CNAME-записей. (Утилита *h2n*, описанная в главе 4, вполне на это способна.)

CNAME-указатели на CNAME-записи

Возможно ли создать псевдоним (CNAME-запись), указывающий на другой псевдоним? Такая возможность полезна в ситуации, когда во внешней зоне существует псевдоним, ссылающийся на каноническое имя в пределах локальной зоны. Допустим, администратор локальной зоны не имеет возможности влиять на «внешний» псевдоним. Но что

если появляется необходимость изменить имя узла локальной зоны, на которое ссылается внешний псевдоним? Можно ли просто создать еще одну CNAME-запись?

Ответ на этот вопрос: да, из CNAME-записей можно создавать цепочки. Цепочки CNAME-записей работают в реализациях пакета BIND, а также явно не запрещены соответствующим RFC-документом. Несмотря на существование такой возможности, ее применение может оказаться не самой лучше мыслью. Документы RFC по DNS не рекомендуют использовать такой механизм из-за потенциальной возможности создания CNAME-петли, а также по причине замедления процесса разрешения имен. Короче говоря, сделать это можно, но если что-нибудь сломается, мало кто в Сети вам посочувствует. Помимо этого, нет никакой гарантии, что увязывание CNAME-записей в цепочку будет работать в новой (не основанной на BIND) реализации DNS-сервера.[1]

Псевдонимы в данных RR-записей

Все записи, кроме имеющих CNAME, обязаны использовать канонические доменные имена в разделе данных. В противном случае приложения и DNS-серверы не будут корректно работать. К примеру, как мы говорили в главе 5 «DNS и электронная почта», *sendmail* распознает в правой части MX-записей только каноническое имя узла, на котором работает. Если этого не происходит, *sendmail* неправильно удаляет MX-записи при сокращении их списка, после чего может произойти доставка узлом почты самому себе либо менее предпочтительным узлам, что приведет к появлению петли маршрутизации.

Встречая псевдоним в правой части записи, DNS-сервер BIND 8 записывает в log-файл примерно такие сообщения:

```
Sep 27 07:43:48 terminator named[22139]: "digidesign.com IN NS" points to a
CNAME (ns1.digidesign.com)
Sep 27 07:43:49 terminator named[22139]: "moreland.k12.ca.us IN MX" points to
a CNAME (mail.moreland.k12.ca.us)
```

Множественные CNAME-записи

Один клинический случай настройки, мысль о возможности существования которого, откровенно говоря, не приходила нам в голову – а мы повидали много клинических случаев – это множественные CNAME-записи для одного доменного имени. Некоторые администраторы используют такую настройку в паре с механизмом «round robin» для смены наборов RR-записей. К примеру, следующие записи:

---

[1] Например, такой реализацией является сервер Microsoft DNS, который поставляется в составе Windows NT и Windows 2000. При этом в нем допустимо увязывание CNAME-записей в цепочку.

```
fullmonty  IN  CNAME  fullmonty1
fullmonty  IN  CNAME  fullmonty2
fullmonty  IN  CNAME  fullmonty3
```

могли бы использоваться с целью получения всех адресов для *fullmonty1*, затем всех адресов для *fullmonty2*, затем всех адресов для *fullmonty3* на DNS-сервере, который не распознает в этой настройке мерзость, каковой она и является (хотя бы потому, что нарушает правило «О CNAME и прочих данных»).

BIND 4 не считает это ошибкой, в отличие от BIND 8 и 9.1.0, а также более поздних версий. BIND 8 предоставляет возможность разрешить такую настройку:

```
options {
                multiple-cnames yes;
};
```

В BIND 9 подобной настройки не существует. По умолчанию, естественно, множественные CNAME-записи запрещены.

Поиск CNAME-записей

Иногда появляется необходимость произвести поиск собственно CNAME-записи, а не данных по каноническому имени, которые она содержит. Это легко сделать с помощью *nslookup* или *dig*. Можно установить тип запроса *cname*, либо *any*, а затем произвести поиск по имени:

```
% nslookup
Default Server:  wormhole
Address:  0.0.0.0

> set query=cname
> bigt
Server:  wormhole
Address:  0.0.0.0

bigt.movie.edu  canonical name = terminator.movie.edu
> set query=any
> bigt
Server:  wormhole
Address:  0.0.0.0

bigt.movie.edu  canonical name = terminator.movie.edu
> exit

% dig bigt.movie.edu cname
; <<>> DiG 8.3 <<>> bigt.movie.edu cname
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 4
;; QUERY SECTION:
```

```
;;      bigt.movie.edu, type = CNAME, class = IN

;; ANSWER SECTION:
bigt.movie.edu.         1D IN CNAME     terminator.movie.edu.
```

Нахождение псевдонимов узла

Есть одна вещь, которую нельзя просто сделать в DNS: узнать псевдонимы для узла. В случае использования таблицы узлов легко найти и каноническое имя узла, и все его псевдонимы: нет разницы, какое из имен искать, поскольку они все перечислены в одной строке:

```
% grep terminator /etc/hosts
192.249.249.3  terminator.movie.edu terminator bigt
```

Однако в DNS при поиске по каноническому имени мы получаем каноническое имя, не более того. Не существует простого способа с помощью DNS-сервера или приложения узнать, существуют ли для этого канонического имени псевдонимы:

```
% nslookup
Default Server:  wormhole
Address:  0.0.0.0

> terminator
Server:  wormhole
Address:  0.0.0.0

Name:    terminator.movie.edu
Address:  192.249.249.3
```

Если воспользоваться *nslookup* или *dig* для поиска по псевдониму, то результаты поиска будут содержать псевдоним и каноническое имя. *nslookup* и *dig* отображают в сообщении псевдоним и каноническое имя. Но в этом случае мы ничего не узнаем о прочих существующих для этого канонического имени псевдонимах:

```
% nslookup
Default Server:  wormhole
Address:  0.0.0.0

> bigt
Server:  wormhole
Address:  0.0.0.0

Name:    terminator.movie.edu
Address:  192.249.249.3
Aliases:  bigt.movie.edu

> exit
% dig bigt.movie.edu
; <<>> DiG 8.3 <<>> bigt.movie.edu
;; res options: init recurs defnam dnsrch
;; got answer:
```

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 3, ADDITIONAL: 4
;; QUERY SECTION:
;;      bigt.movie.edu, type = A, class = IN

;; ANSWER SECTION:
bigt.movie.edu.         1D IN CNAME     terminator.movie.edu.
terminator.movie.edu.   1D IN A         192.249.249.3
```

Едва ли ни единственным способом получить все CNAME-записи для узла – это получить зону целиком, а затем выбрать CNAME-записи для интересующего имени:

```
% nslookup
Default Server:  wormhole
Address:  0.0.0.0

> ls -t cname movie.edu
 [wormhole.movie.edu]
$ORIGIN movie.edu.
bigt                    1D IN CNAME     terminator
wh                      1D IN CNAME     wormhole
dh                      1D IN CNAME     diehard
>
```

И даже этот метод позволяет получить только псевдонимы, определенные в пределах зоны; а могут существовать псевдонимы, определенные для того же канонического имени в других зонах.

Маски

Есть еще одна вещь, которую мы еще не изучили подробно, – *маски* DNS. Иногда хочется, чтобы одна RR-запись использовалась для многих доменных имен, и не было необходимости создавать огромное количество записей, которые одинаковы во всем, кроме доменного имени. DNS резервирует специальный символ, звездочку (*), для использования в файлах данных зоны в качестве части маски. Звездочка сопоставляется с любым числом меток в имени за исключением тех случаев, когда запись для имени уже существует в базе данных DNS-сервера.

Чаще всего маски используются для передачи почты в сети без подключения к Интернету. Предположим, что наши машины не имеют внешнего подключения, но у нас есть один узел, который производит передачу почты между локальной сетью и Интернетом. Мы можем создать в зоне *movie.edu* MX-запись с маской, чтобы программы, отправляли всю нашу почту узлу-ретранслятору. Пример:

```
*.movie.edu.  IN  MX  10 movie-relay.nea.gov.
```

Поскольку символ * сопоставляется с одной или несколькими метками, эта RR-запись будет использована для имен вроде *terminator.mo-*

vie.edu, *empire.fx.movie.edu* и *casablanca.bogart.classics.movie.edu*. Опасность использования масок заключается в том, что они могут создавать конфликты со списком поиска. Приведенной маске соответствует имя *cujo.movie.edu.movie.edu*, так что использование масок во внутренних данных зоны просто опасно. Вспомним, что некоторые из версий *sendmail* используют список поиска при поиске MX-записей:

```
% nslookup
Default Server:  wormhole
Address:  0.0.0.0

> set type=mx                    – искать MX-записи
> cujo.movie.edu                 – для cujo
Server:  wormhole
Address:  0.0.0.0

cujo.movie.edu.movie.edu    – Узел с таким именем не существует!
        preference = 10, mail exchanger = movie-relay.nea.gov
```

Каковы ограничения масок? Маски не сопоставляются с доменными именами, для которых уже определены данные. Допустим, мы использовали маски в данных нашей зоны, что отражено в приводимом фрагменте файла *db.movie.edu*:

```
*     IN  MX  10 mail-hub.movie.edu.
et    IN  MX  10 et.movie.edu.
jaws  IN  A   192.253.253.113
fx    IN  NS  bladerunner.fx.movie.edu.
fx    IN  NS  outland.fx.movie.edu.
```

Почта для узла *terminator.movie.edu* посылается узлу *mail-hub.movie.edu*, но почта для узла *et.movie.edu* посылается напрямую этому узлу. Поиск MX-записей для *jaws.movie.edu* приведет к получению ответа, что MX-записи для данного доменного имени не существуют. Маска не может быть использована, поскольку для этого имени существует адресная запись. Маска также не будет использована для доменных имен в пределах зоны *fx.movie.edu*, поскольку маски не распространяются за границы делегирования. Указанная маска также будет использована для доменного имени *movie.edu*, поскольку маска должна сопоставляться с нулевым или большим числом меток, *завершаемых точкой*, за которой следует имя *movie.edu*.

Ограничение MX-записей

Раз уж мы заговорили про записи типа MX, рассмотрим ситуацию, когда почтовые сообщения отправляются более длинным, чем следует, маршрутом. MX-записи представляют собой список информативных данных, который возвращается при поиске доменного имени конечного пункта назначения. Этот список не упорядочен в соответствие с тем,

какой из почтовых ретрансляторов находится ближе к получателю. Приведем пример связанной с этим обстоятельством проблемы. В нашей сети, не подключенной к Интернету, существует два узла, умеющих передавать в нашу сеть интернет-почту. Один узел расположен в США, а другой – во Франции. Сеть находится в Греции. Большая часть почты приходит из США, поэтому приходится заниматься сопровождением зоны и создать две MX-записи с масками – отдав наибольшее предпочтение передающему узлу в США, а наименьшее – узлу во Франции. Поскольку узел в США имеет более высокий приоритет, *вся* почта будет отправляться через этот узел (разумеется, если он доступен). Если человек из Франции пошлет нам письмо, оно пропутешествует через Атлантический океан в США и обратно, поскольку ничто в списке MX-записей не говорит о том, что французский ретранслятор находится ближе к отправителю.

Коммутируемые соединения

Еще одна относительно новая вещь, представляющая некоторые сложности для DNS, – это коммутируемые соединения с сетью Интернет. Во времена молодости сети Интернет, когда появилась DNS, коммутируемые соединения не существовали. После невероятного взрыва популярности Интернета и возникновения многочисленных провайдеров интернет-услуг, предлагающих коммутируемое подключение к сети широким массам, возникло целое семейство проблем, связанных с использованием службы имен.

Основная цель при установке и настройке DNS для работы при коммутируемом соединении – дать каждому узлу внутренней сети возможность находить адреса всех узлов, к которым этот узел должен иметь доступ. (Разумеется, если соединение с сетью Интернет не установлено, узлам нет необходимости производить разрешение доменных имен в Интернете). Если в сети используется установка соединения по необходимости (dial-on-demand), существует дополнительная цель – сокращение числа устанавливаемых соединений: если происходит поиск для доменного имени узла, расположенного в пределах локальной сети, это не должно приводить к установлению соединения с сетью Интернет.

Мы разделим коммутируемые соединения на две категории: производимые вручную (подразумевается, что соединение с сетью устанавливается пользователем) и производимые по необходимости (подразумевается использование устройства – возможно, маршрутизатора, но чаще всего обычного узла сети, работающего под управлением Linux или другой серверной операционной системы – для автоматического подключения к сети Интернет при создании узлами трафика, который относится к ресурсам Интернета). Для каждой из категорий мы рассмотрим два сценария. Первый – для случая, когда существует лишь один

узел, устанавливающий соединение с сетью Интернет, а второй – для случая небольшой сети узлов, устанавливающей соединение. Но прежде, мы выясним, что именно служит причиной установки коммутируемого соединения и как минимизировать коммутацию.

Причины соединений

Многие пользователи – особенно в Европе, где широко применяются ISDN-подключения – используют для подключения к сети соединения типа dial-on-demand (коммутация по необходимости). Практически все эти пользователи желают минимизировать, если не исключить абсолютно, ненужные подключения к сети Интернет. Установка соединения часто стоит дороже, чем оплата использования сетевых ресурсов, и всегда занимает какое-то время.

К сожалению, нельзя сказать, что DNS-серверы BIND замечательно приспособлены для работы с соединениями по необходимости. Они периодически посылают системные запросы с целью получения текущего списка корневых DNS-серверов, даже если не занимаются непосредственно разрешением доменных имен. Помимо этого, работа со списком поиска может приводить к отправке запросов удаленным DNS-серверам. Допустим, локальное доменное имя узла – *tinyoffice.megacorp.com*, и на этом узле работает DNS-сервер, авторитативный для данной зоны. Для некоторых клиентов список поиска по умолчанию будет, скорее всего, выглядеть следующим образом:

```
tinyoffice.megacorp.com
megacorp.com
```

Предположим, мы пытаемся установить FTP-соединение с одной из систем локальной сети, *deadbeef.tinyoffice.megacorp.com*, но сделали ошибку в имени и вместо *deadbeef* набрали *deadbeer*:

```
% ftp deadbeer
```

Используя список поиска, клиент начнет работу с имени *deadbeer.tinyoffice.megacorp.com*. Локальный DNS-сервер, авторитативный для зоны *tinyoffice.megacorp.com*, сразу ответит, что это доменное имя не существует. Тогда клиент добавляет второе доменное имя из списка поиска и ищет имя *deadbeer.megacorp.com*. Чтобы понять, существует ли это доменное имя, DNS-сервер должен послать запрос DNS-серверу зоны *megacorp.com*, а это потребует установления связи с сетью Интернет.

Минимизация числа подключений

Существует несколько основных приемов минимизации подключений к сети Интернет. Первый, и вероятно самый простой, – использовать версию BIND, в которой поддерживается отрицательное кэширование (то есть любой пакет с версией более поздней, чем 4.9.5, но мы, безус-

ловно, предпочитаем BIND 8 или 9). В этом случае, если имя *deadbeer* ошибочно попадает в файл настройки, DNS-сервер производит поиск для имени *deadbeer.megacorp.com* единожды, а затем кэширует тот факт, что доменное имя не существует, на время, определяемое значением времени жизни отрицательных ответов для зоны *megacorp.com*.

Второе: следует использовать список поиска минимальных размеров. Если локальное доменное имя – *tinyoffice.megacorp.com*, можно обойтись списком поиска, который содержит только это имя. В этом случае опечатка не вызовет подключения к сети.

Также важно использовать современный клиент. В клиентах BIND версий более поздних, чем 4.9, список поиска по умолчанию содержит только локальное доменное имя (такой список в нашей книге мы называем «минимальным»). К тому же, современный клиент умеет производить буквальный поиск по введенным именам, которые содержат точки, даже в случаях, когда имя не заканчивается точкой.

И наконец, можно использовать другую службу имен, скажем файл */etc/hosts*, для разрешения локальных имен и настроить клиенты на использование DNS только в случаях, когда имя не найдено в */etc/hosts*. Если имена всех локальных узлов присутствуют в файле */etc/hosts*, можно не беспокоиться о ненужных подключениях.

Теперь используем эти приемы для двух упомянутых сценариев.

Один узел, подключение вручную

Самый легкий способ решить проблемы для случая одного узла и подключения вручную – настроить клиент узла на использование DNS-сервера, который управляется провайдером интернет-услуг. Большинство провайдеров поддерживают DNS-серверы для нужд своих пользователей. Если вы не уверены, справедливо ли это и для вашего провайдера либо если неизвестны адреса DNS-серверов, попробуйте поискать информацию на веб-сайте провайдера, послать письмо с вопросом или позвонить по телефону в службу поддержки.

Некоторые операционные системы, в частности Windows 95, 98 и NT, позволяют для каждого провайдера коммутируемых интернет-соединений задавать набор используемых DNS-серверов. Можно создать один список используемых DNS-серверов для подключений через UUNet и другой – для подключений к вашему офису. Если вы пользуетесь услугами нескольких провайдеров, такая возможность весьма полезна.

Подобная настройка обычно вполне удовлетворяет потребностям среднего пользователя коммутируемых соединений. Разрешение имен не будет работать, если подключение не установлено, но навряд ли это является проблемой, поскольку без подключения к сети Интернет нет особого смысла использовать службу имен.

Но некоторым из читателей, возможно, захочется иметь работающий DNS-сервер при активном коммутируемом подключении. DNS-сервер может повысить производительность, кэшируя информацию для доменных имен, которые часто используются. На любой Unix-подобной системе (например, Linux) этого добиться легко: обычно применяется сценарий вроде *ifup* для установки подключения и *ifdown* для разрыва соединения. В таком случае, скорее всего, существуют сценарии *ifup-post* и *ifdown-post*, которые вызываются сценариями *ifup* и *ifdown* после выполнения основных действий. Можно запустить сервер *named* по имени либо с помощью команды *ndc start* из сценария *ifup-post* и завершить его работу с помощью *ndc stop* или *rndc stop* в сценарии *ifdown-post*. Едва ли не единственное, что останется сделать, – вписать локальное доменное имя в файл *resolv.conf*. Стандартное поведение клиента, при котором запросы посылаются DNS-серверу, работающему на том же узле, отлично подходит как для работы с запущенным DNS-сервером, так и для работы без него.

Несколько узлов, подключение вручную

Самое простое решение, пригодное к использованию в случае нескольких узлов и подключения, производимого вручную, весьма похоже на решение для предыдущего случая. Можно настроить клиенты на использование DNS-серверов провайдера интернет-услуг, и также настроить клиенты на просмотр файла */etc/hosts* (или NIS, если вы используете подобные вещи) *до* отправки запроса DNS-серверу. Следует убедиться, что файл */etc/hosts* содержит имена всех узлов локальной сети.

Если вы намереваетесь запускать локальный DNS-сервер, конфигурацию придется лишь немного изменить: настроить клиенты на использование локального сервера, а не DNS-серверов провайдеров интернет-услуг. Это позволит воспользоваться преимуществами локального кэширования, но разрешение локальных имен будет работать (с помощью файла */etc/hosts*) даже в случае, когда подключение к сети Интернет отсутствует. Можно запускать DNS-сервер и прекращать его работу описанным выше способом – из сценариев *ifup-post* и *ifdown-post*.

Тем, кто непременно хочет использовать DNS для разрешения *всех* имен, можно посоветовать избавиться от файла */etc/hosts* и создать зоны прямого и обратного отображения для узлов локальной сети на локальном же DNS-сервере. Следует сократить до минимума списки поиска клиентов, чтобы до минимума сократить возможность поиска удаленного доменного имени DNS-сервером.

Один узел, подключение по необходимости

Если речь идет об одном узле с коммутируемым подключением к сети Интернет по необходимости, самое простое решение все то же – настроить клиент на использование DNS-серверов провайдера интернет-

услуг, так что, когда клиенту понадобится произвести поиск для доменного имени, он пошлет запрос одному из этих DNS-серверов (и инициирует подключение). Если существуют доменные имена, поиск для которых этот узел производит регулярно в качестве рутинного действия, скажем, *localhost* или *1.0.0.127.in-addr.arpa*, можно добавить эти имена в */etc/hosts* и настроить клиент на поиск в этом файле перед отправкой запроса DNS-серверу.

Если существует необходимость в локальном DNS-сервере, убедитесь, что он умеет отображать имена *localhost* и *1.0.0.127.in-addr.arpa* в адрес *127.0.0.1* и имя *localhost* соответственно, а также сократите до минимума список поиска.

Если DNS-сервер устанавливает подключение чаще, чем следует по вашему мнению, попробуйте включить регистрацию запросов (с помощью *options query-log* в DNS-сервере BIND 4.9, *ndc querylog* в DNS-сервере BIND 4.9 или 8 либо *rndc querylog* в DNS-сервере BIND 9.1.0) и выясните, для каких доменных имен поиск приводит к подключению. Если многие из этих доменных имен находятся в одной зоне, можно настроить локальный DNS-сервер в качестве вторичного для этой зоны. В этом случае подключение будет устанавливаться не чаще, чем один раз за интервал обновления, в целях разрешения доменных имен зоны.

Несколько узлов, подключение по необходимости

Самое простое решение для данного сценария в точности совпадает с первым описанным решением из раздела «Несколько узлов, подключение вручную»: настраиваются только клиенты – на поиск в файле */etc/hosts* перед отправкой запросов DNS-серверу. Как и для всех прочих случаев с подключением по необходимости, следует сократить до минимума список поиска.

Как вариант, можно попробовать одну из следующих конфигураций: использование локального DNS-сервера в качестве резерва для */etc/hosts* либо создание зон прямого и обратного отображения для локальных узлов на локальном DNS-сервере.

Работа авторитативного DNS-сервера через подключение по необходимости

Некоторым из читателей это покажется глупостью – кому может придти в голову держать авторитативный DNS-сервер при коммутируемом подключении по необходимости? Но в некоторых частях света, где большая пропускная способность каналов и подключение к сети Интернет встречаются вовсе не на каждом шагу, это бывает неизбежно. Верьте или нет, но в BIND существует механизм создания таких DNS-серверов.

Если авторитативный DNS-сервер работает при соединении по необходимости, имеет смысл стараться сократить активность по управлению

зоной до как можно более маленького временного окна. Если DNS-сервер является авторитативным для сотни зон, навряд ли будет здорово видеть, как каждые несколько минут срабатывают таймеры обновления и запросы SOA-записей приводят к установлению соединения раз за разом.

В BIND 8.2 и более поздних версиях DNS-серверов существует возможность изменять *интервал сердцебиений* (*heartbeat interval*). *Интервал сердцебиений* определяет, насколько часто (в минутах) серверу следует производить подключение по необходимости:

```
options {
    heartbeat-interval 180;      // 3 часа
};
```

По умолчанию интервал длится 60 минут; выполнение служебных операций можно отключить, установив нулевой интервал.

Если отметить одну или несколько зон как обслуживаемые по коммутируемому каналу, DNS-сервер будет пытаться объединить выполнение служебных операций для этих зон в одном, коротком интервале времени, и выполнять служебные операции не чаще чем один раз в течение интервала сердцебиения. Для вторичного DNS-сервера это означает замедление работы таймера обновления зоны (вплоть до нарушения установленного интервала, в случаях, когда он меньше интервала сердцебиения) и запрос SOA-записей у мастер-сервера только по сердцебиению. Для мастер-сервера это означает отправку NOTIFY-сообщений, которые, предположительно, приводят к подключению к сети Интернет и инициируют обновление зоны на дополнительных DNS-серверах.

Чтобы отметить все зоны DNS-сервера в качестве обслуживаемых через коммутируемый канал, следует воспользоваться предписанием *dialup* оператора *options*:

```
options {
    heartbeat-interval 60;
    dialup yes;
};
```

Чтобы отметить одну зону в качестве обслуживаемой по коммутируемому каналу, следует использовать предписание *dialup* оператора *zone*:

```
zone "movie.edu" {
    type master;
    file "db.movie.edu";
    dialup yes;
};
```

Зоны, обслуживаемые через коммутируемые каналы, полезны еще в одном качестве, которое изначально не предусматривалось: на DNS-серверах, которые являются вторичными для тысяч зон. Некоторые

провайдеры интернет-услуг предоставляют услуги по сопровождению дополнительных DNS-серверов для пользовательских зон и подвергаются нападкам злодеев, устанавливающих слишком маленький интервал обновления для зон. Эти вторичные серверы оказываются перегружены посылкой SOA-запросов для таких зон. Сделав все зоны обслуживаемыми через коммутируемые каналы и установив разумный интервал сердцебиения, провайдер имеет возможность бороться с таким эффектом.

Имена и номера сетей

Исходная спецификация DNS не содержала возможности производить поиск имен сетей на основе их номеров, хотя такая возможность существовала в системе, основанной на файле *HOSTS.TXT*. После этого в документе RFC 1101 была определена система хранения имен сетей; эта система также действует для подсетей и масок подсетей, поэтому значительно превосходит механизм, который использовался в *HOSTS.TXT*. Более того, эта система не требует изменений в программном обеспечении службы имен; она полностью основана на умелом использовании PTR- и A-записей.

Вспомним, что для преобразования IP-адреса в имя в DNS следует записать IP-адрес в обратном порядке, добавить *in-addr.arpa*, а затем произвести поиск PTR-записи для полученного имени. Этот же метод используется для отображения номеров сетей в имена сетей, к примеру, номера сети 15/8 в имя «HP Internet». Чтобы произвести поиск по номеру сети, следует дополнить биты номера сети нулями до четырех байтов, а затем произвести поиск PTR-данных, как для IP-адреса узла. Так, чтобы найти имя сети для старой сети ARPAnet, которая имеет номер 10/8, следует произвести поиск PTR-данных для имени *0.0.0.10.in-addr.arpa*. Должен быть получен ответ вроде *ARPAnet.ARPA*.

Если бы ARPAnet являлась подсетью, для имени *0.0.0.10.in-addr.arpa* можно было бы дополнительно найти адресную запись. Адрес представлял бы собой маску сети, скажем 255.255.0.0. Если бы нас интересовало имя подсети, а не сети, следовало бы наложить маску на IP-адрес и произвести поиск по номеру подсети.

Эта техника позволяет производить преобразование номеров сетей в имена сетей. Чтобы получить законченное решение, необходимо придумать способ отображения имени сети в номер сети. Как и раньше, способ основан на использовании PTR-записей. Имя сети имеет связанные PTR-данные, которые определяют номер сети (в обратном порядке с добавленным именем *in-addr.arpa*).

Посмотрим теперь, как такие данные могут выглядеть в файлах данных зоны HP (HP Internet имеет номер сети 15/8), и произведем поиск имени сети по ее номеру.

Фрагмент файла *db.hp.com*:

```
;
; Отображение имени сети HP в адрес 15.0.0.0.

;
hp-net.hp.com.            IN  PTR 0.0.0.15.in-addr.arpa.
```

Фрагмент файла *db.corp.hp.com*:

```
;
; Отображение имени подсети corp в адрес 15.1.0.0.
;
corp-subnet.corp.hp.com.  IN  PTR 0.0.1.15.in-addr.arpa.
```

Фрагмент файла *db.15*:

```
;
; Отображение адреса 15.0.0.0 в hp-net.hp.com.
; Маска подсети HP - 255.255.248.0.
;
0.0.0.15.in-addr.arpa.    IN  PTR hp-net.hp.com.
                          IN  A   255.255.248.0
```

Фрагмент файла *db.15.1*:

```
;
; Отображение адреса 15.1.0.0 в имя подсети.
;
0.0.1.15.in-addr.arpa.    IN  PTR corp-subnet.corp.hp.com.
```

Ниже приводится процедура поиска имени подсети по IP-адресу 15.1.0.1:

1. Применить стандартную для класса адреса маску сети. Адрес 15.1.0.1 принадлежит классу A, его маска 255.0.0.0. Наложение маски на IP-адрес позволяет получить номер сети – 15.
2. Сделать запрос (*type=A* или *type=ANY*) для имени *0.0.0.15.in-addr.arpa*.
3. Ответное сообщение содержит адресные данные. Поскольку существуют адресные данные для *0.0.0.15.in-addr.arpa* (маска подсети, 255.255.248.0), следует применить маску подсети к IP-адресу. Получаем 15.1.0.0.
4. Сделать запрос (*type=A* или *type=ANY*) для имени *0.0.1.15.in-addr.arpa*.
5. Ответное сообщение не содержит адресных данных, делаем вывод, что 15.1.0.0 не делится на более мелкие сети.
6. Сделать запрос PTR-данных для *0.0.1.15.in-addr.arpa*.
7. Ответное сообщение содержит имя сети 15.1.0.1: *corp-subnet.corp.hp.com*.

Помимо двусторонних преобразований между именами и номерами сетей, можно также перечислить все сети зоны в PTR-записях:

```
movie.edu.  IN  PTR  0.249.249.192.in-addr.arpa.
            IN  PTR  0.253.253.192.in-addr.arpa.
```

А теперь плохая новость: несмотря на тот факт, что документ RFC 1101 содержит все, что нужно знать, чтобы начать работать с этим механизмом, очень немногие известные нам программы *используют* эту схему кодирования имен сетей, и очень немногие администраторы тратят время на добавление информации такого рода. До тех пор, пока программы не начнут использовать имена сетей в кодировке DNS, практически единственной причиной для настройки этого механизма будет оставаться желание похвастаться. Но для многих из нас этой причины вполне достаточно.

Дополнительные RR-записи

Существует некоторое количество RR-записей, которые мы еще не рассматривали. Первая из таких записей, HINFO, существует с самого начала работы DNS, но никогда широко не применялась. Прочие записи были определены в документе RFC 1183 и некоторых последующих RFC-документов. Большинство записей являются экспериментальными, но некоторые стали практически стандартом и используются все чаще. Мы опишем эти записи, чтобы предоставить читателям небольшую фору в их использовании.

Информация об узле

HINFO – это сокращение Host INFOrmation (информация об узле). Запись содержит пару строк, которые идентифицируют тип аппаратного обеспечения узла. Предположительно, строки должны принадлежать наборам MACHINE NAMES и OPERATING SYSTEM NAMES, описанным в документе RFC «Assigned Numbers» (в настоящее время это RFC 1700), но это необязательное требование, вы можете использовать свои собственные сокращения. Упомянутый документ RFC не является исчерпывающим, и вполне возможно, что ваша система попросту отсутствует в списке. Изначально записи информации об узле имели вспомогательную функцию – они должны были позволять службам вроде FTP определять, как следует взаимодействовать с удаленной системой. Подобный механизм позволил бы, к примеру, автоматизировать согласование преобразований типов передаваемых данных. К сожалению, этого не произошло – очень немногие сети предоставляют точную HINFO-информацию для своих систем. Отдельные системные администраторы используют записи HINFO, чтобы отслеживать типы машин, не пользуясь базой данных или записной книжкой. Вот два примера записей HINFO; обратите внимание, что тип аппаратного обеспечения и

имя операционной системы должны заключаться в кавычки, если содержат пробелы:

```
;
; Эти имена отсутствуют в RFC 1700
;
wormhole  IN  HINFO  ACME-HW  ACME-GW
cujo      IN  HINFO  "Watch Dog Hardware"  "Rabid OS"
```

Прежде чем добавлять записи HINFO к данным зоны, в особенности – к данным зоны, видимой из сети Интернет, следует осознать, что записи HINFO представляют угрозу безопасности системы. Предоставляя легко доступную информацию о системе, вы облегчаете взломщику задачу.

AFSDB

Синтаксис записей AFSDB схож с синтаксисом MX–записей, а семантика немного похожа на семантику NS-записей. Запись AFSDB отражает расположение сервера базы данных AFS для ячейки либо DNS-сервера с DCE-идентификацией. Тип сервера, на который указывает запись, а также имя узла, на котором работает сервер, содержатся в данных для записи.

Что же такое сервер базы данных AFS? Или просто AFS, если уж на то пошло. AFS (Andrew File System) – изначально это файловая система Эндрю, разработанная отличными ребятами в университете Карнеги-Меллона в качестве части Проекта Эндрю – Andrew Project. (Этот проект сегодня существует в качестве продукта от IBM.) AFS – это сетевая файловая система, как и NFS, но она гораздо лучше, чем NFS, справляется с увеличением задержек в больших сетях и обеспечивает локальное кэширование файлов в целях повышения производительности. На сервере базы данных AFS существует процесс, отвечающий за отслеживание файловых наборов (групп файлов) на различных серверах файлов AFS в пределах одной ячейки (логической группы узлов). Таким образом, поиск любого файла в ячейке связан с возможностью найти сервер базы данных AFS для этой ячейки.

Что такое удостоверенный (authenticated) DNS-сервер? DNS-сервер, обладающий информацией о расположении различных служб, доступных в пределах DCE-ячейки. DCE-ячейка? Логическая группа узлов, которые разделяют службы, предоставляемые Распределенной вычислительной средой (Distributed Computing Environment, DCE) от Open Group.

Вернемся к нашим баранам. Чтобы получить доступ к AFS- или DCE-службе другой ячейки по сети, следует сначала определить, где находятся серверы баз данных этой ячейки или удостоверенные DNS-серверы. Отсюда и необходимость существования нового типа записи. Доменное имя, с которым связана запись, является именем ячейки, кото-

рая известна серверу. Ячейки часто носят те же имена, что и домены DNS, так что записи выглядят вполне привычно.

Как уже было сказано, синтаксис записей AFSDB схож с синтаксисом MX-записей. Вместо приоритета указывается число 1 для сервера базы данных AFS либо число 2 для DNS-сервера с DCE-идентификацией.

Вместо имени узла-ретранслятора указывается имя узла, на котором работает сервер. И все!

Предположим, системный администратор *fx.movie.edu* создает DCE-ячейку (которая включает AFS-службы), поскольку хочет поэкспериментировать с распределенными вычислениями в целях ускорения обработки графики. Сервер базы данных AFS для ячейки, и DNS-сервер DCE работают на узле *bladerunner.fx.movie.edu*, плюс еще один сервер базе данных для клетки работает на узле *empire.fx.movie.edu*, а второй DNS-сервер DCT на узле *aliens.fx.movie.edu*. Следует создать следующие AFSDB-записи:

```
; Наша DCE-ячейка называется fx.movie.edu, то есть носит то же имя, что и
зона
fx.movie.edu.  IN  AFSDB  1 bladerunner.fx.movie.edu.
               IN  AFSDB  2 bladerunner.fx.movie.edu.
               IN  AFSDB  1 empire.fx.movie.edu.
               IN  AFSDB  2 aliens.fx.movie.edu.
```

X25, ISDN и RT

Эти три типа записей были созданы специально для поддержки исследований интернет-сетей следующего поколения. Две записи, X25 и ISDN, являются обычными адресными записями, специфичными для X.25- и ISDN-сетей. В каждом случае тип данных записи соответствует потребностям сети. В типе записей X25 используется адрес X.121 (X.121 – это рекомендация ITU-T, которая определяет формат адресов, используемых в сетях X.25). В записях ISDN используются ISDN-адреса.

ISDN расшифровывается как Integrated Services Digital Network (Цифровая сеть с предоставлением комплексных услуг). Телефонные компании всего мира используют протоколы ISDN для передачи голоса и данных по своим сетям и создания таким образом сети комплексных услуг. ISDN достаточно спорадически встречается в США, но весьма широко используется в других странах. Поскольку в ISDN используются сети телефонных компаний, адрес ISDN – это просто телефонный номер, который состоит, по сути дела, из кода страны, кода области или города, а также местного телефонного номера. Иногда в конце адреса присутствуют дополнительные цифры, которых нет в телефонном номере. Эти цифры являются вторичным адресом. В DNS-записях дополнительный адрес записывается в качестве отдельного поля.

Вот примеры X25- и ISDN-записей:

```
relay.pink.com.  IN  X25  31105060845

delay.hp.com.    IN  ISDN  141555514539488
hep.hp.com.      IN  ISDN  141555514539488 004
```

Эти записи должны использоваться совместно с записями сквозной маршрутизации – Route Through (или RT-записи). Синтаксически и семантически RT-записи схожи с MX-записями: они определяют промежуточные узлы, которые маршрутизируют *пакеты* (вместо почтовых сообщений) в целях доставки их узлу-адресату. Использование этих записей позволяет не только доставить почту узлу, не подключенному напрямую к сети Интернет, но доставить любой вид IP-пакетов этому узлу, используя другой узел в качестве ретранслятора. Пакет может быть частью сеанса Telnet или FTP и даже запросом DNS!

В RT-записях также существует значение предпочтения, показывающее, насколько желательна доставка через определенный узел. Так, следующие записи:

```
housesitter.movie.edu.  IN  RT  10 relay.pink.com.
                        IN  RT  20 delay.hp.com.
```

предписывают доставлять пакеты, адресованные узлу *housesitter.movie.edu* через *relay.pink.com* (более предпочтительно) либо через *delay.hp.com* (менее предпочтительно).

С X25-, ISDN- и даже A-записями RT работает следующим образом:

1. Интернет-узел A желает отправить пакет узлу B, который не подключен к сети Интернет.
2. Узел A производит поиск RT-записей для узла B. Этот поиск также возвращает все адресные записи (A, X25 и ISDN) для каждого из промежуточных узлов.
3. Узел A сортирует список промежуточных узлов и ищет собственное доменное имя. Если имя найдено, узел удаляет это имя и все промежуточные узлы с более высокими приоритетами. Аналогичным образом *sendmail* сокращает список почтовых ретрансляторов.
4. Узел A изучает запись (или записи) для наиболее предпочтительного из оставшихся узлов. Если узел A имеет подключение к сети, для которой существует адресная запись соответствующего типа, то эта сеть используется для отправки пакета промежуточному узлу. К примеру, если бы узел A намеревался послать пакет через узел *relay.pink.com*, ему понадобилось бы подключение к сети X.25.
5. Если соответствующее подключение узла A отсутствует, происходит переход к следующему из промежуточных узлов, указанных RT-записями. К примеру, если у узла A отсутствует подключение к сети X.25, он может использовать отправку через ISDN – узлу *delay.hp.com*.

Этот процесс повторяется до тех пор, пока пакет не будет доставлен наиболее предпочтительному из промежуточных узлов. Затем узел, получивший этот пакет, может доставить его прямо по адресу (A, X25 или ISDN) узла-получателя.

Координаты

Документ RFC 1876 определяет экспериментальный тип записей LOC, который позволяет администраторам записывать координаты своих компьютеров, подсетей и сетей. В данном случае координаты подразумевают широту, долготу и высоту. В будущем приложения могли бы использовать эту информацию для создания сетевых карт, оценки эффективности маршрутизации и прочих подобных целей.

В основном варианте LOC-запись содержит широту, долготу и высоту (в порядке перечисления). Широта и долгота имеют следующий формат записи:

```
<градусы> [минуты [секунды.<доли секунды>]] (N|S|E|W)
```

Высота выражается в метрах.

Если вы спрашиваете себя, где взять подобную информацию, прочитайте документ «RFC 1876 Resources» («Ресурсы RFC 1876»), расположенный по адресу *http://www.ckdhr.com/dns-loc*. Этот сайт, созданный Кристофером Дэвисом (Christopher Davis), одним из авторов документа RFC 1876, является незаменимым источником информации, полезных ссылок и инструментов для создания LOC-записей.

Если у вас нет собственного приемника системы глобального ориентирования (Global Positioning System, GPS), который можно было бы носить с собой, – нам *известно*, что многие так делают, – советуем посетить два очень полезных сайта: Etak's Eagle Geocoder по адресу *http://www.geocode.com/eagle.html-ssi*, который можно использовать для поиска широты и долготы большинства адресов в пределах США, и AirNav's Airport Information по адресу *http://www.airnav.com/airports*, который позволяет определить высоту для ближайшего аэропорта. Не переживайте, если неподалеку нет большого аэропорта, потому что база данных содержит информацию даже по посадочной площадке для вертолетов, расположенной у больницы, что недалеко от моего дома!

Вот LOC-запись для одного из наших узлов:

```
huskymo.boulder.acmebw.com.  IN  LOC  40 2 0.373 N 105 17 23.528 W 1638m
```

Необязательные поля записи позволяют указать, насколько велика описываемая сущность, – в метрах (в конце концов, записи LOC могут использоваться для описания довольно крупных сетей), а также горизонтальную и вертикальную точность. Размер по умолчанию принимается равным одному метру (идеально для единственного узла), горизонтальная точность – десяти тысячам метров, а вертикальная – деся-

ти метрам. Стандартные значения представляют размеры типичной ZIP-зоны или зоны почтового кода. Идея заключается в том, что можно относительно легко найти широту и долготу, исходя из ZIP-кода.

Также можно связывать LOC-записи с именами подсетей и сетей. Если вы потратили информацию на ввод информации об именах и адресах своих сетей в формате, который описывается в документе RFC 1101 (упоминался ранее в этой главе), то можете связать LOC-записи с именами сетей:

```
;
; Отображение имени сети HP в адрес 15.0.0.0.
;
hp-net.hp.com.   IN   PTR 0.0.0.15.in-addr.arpa.
                 IN   LOC 37 24 55.393 N 122 8 37 W 26m
```

SRV

Поиск службы или конкретного типа сервера в пределах зоны – нелегкая задача, если неизвестно, на каком узле работает служба или сервер. Отдельные администраторы пытались решить эту задачу, используя специальные псевдонимы служб в своих зонах. К примеру, для Университета кинематографии мы создали псевдоним *ftp.movie.edu*, который указывает на доменное имя узла, на котором расположен FTP-архив:

```
ftp.movie.edu.      IN    CNAME          plan9.fx.movie.edu.
```

Пользователям будет нетрудно догадаться, какое доменное имя приведет их к нашему FTP-архиву, к тому же, доменное имя, используемое для доступа к архиву, отделяется от доменного имени узла, на котором работает служба FTP. При переносе архива на другой узел можно просто изменить CNAME-запись.

Экспериментальная SRV-запись, представленная в документе RFC 2052, обеспечивает основу общего механизма для поиска служб. SRV также предоставляет мощные возможности, которые позволяют администраторам зон производить распределение нагрузки и создавать резервные службы; аналогичная функциональность предоставляется MX-записями.

Уникальность SRV-записей заключается в формате доменных имен, с которыми эти записи связываются. Как и специальные псевдонимы для служб, доменное имя, с которым связывается SRV-запись, позволяет получить имя для искомой службы, а также протокол, по которому эта служба работает, причем имя и протокол конкатенируются. Метки, представляющие имя службы и протокол, начинаются с подчеркивания, чтобы их можно было отличать от меток доменного имени или имени узла. Так, строка

```
_ftp._tcp.movie.edu
```

представляет SRV-запись, которую следует получить в целях поиска FTP-серверов *movie.edu*, а строка:

```
_http._tcp.www.movie.edu
```

представляет SRV-запись, которая должна быть получена при доступе к URL-адресу *http://www.movie.edu* – в целях поиска веб-серверов *www.movie.edu*.

Имена служб и протоколов должны присутствовать в самом свежем документе RFC «Assigned Numbers» (на момент написания книги это RFC 1700) либо представлять собой уникальные имена, которые используются исключительно локально. Не используйте *номера* портов и протоколов, только имена.

В SRV-записи четыре поля: *приоритет*, *вес*, *порт* и *цель*. Приоритет, вес и порт – положительные 16-битные числа (от 0 до 65535). Цель – доменное имя.

Приоритет работает аналогично предпочтению для MX-записи: чем меньше число в поле приоритета, тем более желательно использование связанной цели. При поиске узлов, предоставляющих искомую службу, клиент должен опросить все цели с заданным приоритетом, прежде чем переходить к более высокому значению.

Веса позволяют администраторам зоны распределять нагрузку между целями. Клиент должен опрашивать цели одного приоритета в пропорции к их весам. К примеру, если одна цель имеет нулевой приоритет и единичный вес, а вторая – нулевой приоритет и вес 2, вторая цель должна быть загружена в два раза больше (в цифрах запросов, соединений, чего угодно), чем первая. Распределение нагрузки ложится на клиента службы: как правило, используется системный вызов для получения случайного числа. Если число в первой трети диапазона, используется первая цель, если число в последних двух третях диапазона, используется вторая цель.

Порт определяет порт, с которым работает искомая служба. Это позволяет администраторам зон запускать серверы на нестандартных портах. Скажем, администратор мог бы использовать SRV-записи, чтобы направлять броузеры на веб-сервер, работающий через порт 8000, а не стандартный HTTP-порт (80).

И наконец, цель определяет доменное имя узла, на котором работает служба (через указанный *порт*). Цель должна определяться каноническим именем узла (не псевдонимом), с которым связаны адресные записи.

Итак, для FTP-сервера *movie.edu* мы добавили следующие записи в файл *db.movie.edu*:

```
ftp._tcp.movie.edu.  IN  SRV  1  0  21  plan9.fx.movie.edu.
                     IN  SRV  2  0  21  thing.fx.movie.edu.
```

FTP-клиенты, понимающие в SRV-записях, должны сначала пробовать связаться с FTP-сервером *plan9.fx.movie.edu* через 21 порт, а затем с FTP-сервером *thing.fx.movie.edu* через 21 порт, если FTP-сервер *plan9.fx.movie.edu* недоступен.

Эти записи:

```
_http._tcp.www.movie.edu.  IN  SRV  0  2  80   www.movie.edu.
                           IN  SRV  0  1  80   www2.movie.edu.
                           IN  SRV  1  1  8000 postmanrings2x.movie.edu.
```

направляют веб-запросы сайта *www.movie.edu* на 80 порт узлов *www.movie.edu* и *www2.movie.edu*, причем узлу *www.movie.edu* достается в два раза больше запросов, чем узлу *www2.movie.edu*. Если ни один из серверов не доступен, запросы направляются серверу *postmanrings2x.movie.edu* через порт 8000.

Чтобы показать, что определенная служба недоступна, можно использовать точку в качестве цели:

```
gopher._tcp.movie.edu.    IN  SRV  0  0  0  .
```

К сожалению, поддержка SRV-записей клиентами, мягко говоря, не часто встречается; выдающимся исключением является система Windows 2000. (Подробнее – чуть позже в этой главе.) И это очень печально, учитывая, какую пользу могли бы принести SRV-записи. Поскольку SRV-записи не получили широкой поддержки, не используйте их вместо адресных записей. Будет разумно включать по меньшей мере одну адресную запись для «основного» доменного имени, с которым связаны SRV-записи, или несколько, если существует необходимость распределять нагрузку между адресами. Если узел в SRV-записях присутствует в качестве резервного, не указывайте его IP-адрес. Кроме того, если служба на определенном узле использует нестандартный порт, не включайте адресную запись для этого узла, поскольку не существует способа направлять клиенты на нестандартный порт с помощью A-записи.

Итак, для *www.movie.edu* мы использовали следующие записи:

```
_http._tcp.www.movie.edu.  IN  SRV  0  2  80   www.movie.edu.
                           IN  SRV  0  1  80   www2.movie.edu.
                           IN  SRV  1  1  8000 postmanrings2x.movie.edu.
www.movie.edu.             IN  A    200.1.4.3 ; адрес www.movie.edu и
                           IN  A    200.1.4.4 ; адрес www2.movie.edu
                                              ; для клиентов, которые не умеют
                                               ; работать с SRV-записями
```

Броузеры, умеющие работать с SRV-записями, будут посылать узлу *www.movie.edu* в два раза больше запросов, чем узлу *www2.movie.edu*, и будут использовать узел *postmanrings2x.movie.edu* только в случае, когда оба основных веб-сервера недоступны. Для всех остальных броузеров будет производиться перестановка (round robin) адресов *www.movie.edu* и *www2.movie.edu*.

DNS и WINS

В первом издании книги – ах, как все было просто в те времена – мы упоминали близкую связь между именами NetBIOS и доменными именами, но отмечали, что, увы, не существует способа заставить DNS работать в качестве DNS-сервера NetBIOS. По существу, чтобы работать в качестве DNS-сервера NetBIOS, DNS-сервер должен поддерживать динамические обновления.

Разумеется, BIND 8 и 9 поддерживают динамические обновления. К сожалению, DHCP-сервер в Windows NT 4.0 не посылает DNS-серверам динамические обновления. Он общается только с WINS-серверами от Microsoft. WINS-серверы реализуют свои собственные, особенные динамические обновления скрытого формата, причем только для NetBIOS-клиентов. Другими словами, серверы WINS не говорят на языке DNS.

Однако Microsoft включает в состав системы Windows NT 4.0 DNS-сервер, который способен общаться с WINS-серверами. В сервере Microsoft DNS есть приятный графический интерфейс администратора, чего и следовало ожидать от корпорации Microsoft, а также удобная привязка к WINS: DNS-сервер можно настроить на посылку запросов адресных данных WINS-серверу в случаях, когда данные не найдены в зоне DNS.

Это делается добавлением новой записи WINS к зоне. WINS-запись, как и SOA-запись, связана с доменным именем зоны. Она работает в качестве флага, предписывающего серверу Microsoft DNS посылать запрос WINS-серверу, если не найден адрес для какого-либо имени. Эта запись:

```
@        0       IN     WINS            192.249.249.39 192.253.253.39
```

предписывает серверу Microsoft DNS посылать запрос по имени WINS-серверу по адресам 192.249.249.39 и 192.253.253.39 (в порядке перечисления). Нулевое значение TTL (времени жизни) – простая предосторожность, предотвращающая кэширование этой записи.

Существует также дополнительная запись WINS-R, которая позволяет серверу Microsoft DNS производить обратное преобразование IP-адресов путем использования NetBIOS-запроса NBSTAT. Если зона *in-addr.arpa* содержит WINS-R-запись вроде следующей:

```
@        0      IN      WINS-R          movie.edu
```

и искомый IP-адрес не содержится в зоне, DNS-сервер попытается послать запрос NBSTAT с целью обратного преобразования IP-адреса. Это примерно то же самое, что позвонить человеку по телефону и спросить «Как тебя зовут?» К результату добавляется точка и доменное имя, указанное в данных записи, в данном случае «.movie.edu».

Эти записи являются ценной связью между двумя пространствами имен. К сожалению, интеграция далека от идеала. Как говорится, самое сложное – в деталях.

Как нам видится, главная проблема заключается в том, что только сервер Microsoft DNS поддерживает записи WINS и WINS-R.[1] Таким образом, если мы хотим, чтобы поиск в зоне *fx.movie.edu* ретранслировался на WINS-сервер факультета спецэффектов, все DNS-серверы *fx.movie.edu* должны быть серверами Microsoft DNS. Почему? Представьте, что DNS-серверы для *fx.movie.edu* представляют набор из серверов Microsoft DNS и серверов BIND. Предположим, удаленный DNS-сервер делает запрос по NetBIOS-имени в зоне *fx.movie.edu*, выбирая при этом сервер на основе метрики времени передачи сигнала. Если был выбран сервер Microsoft DNS, он сможет произвести разрешение имени в динамически назначаемый адрес. Но если запрос получил сервер BIND, он не сможет произвести разрешение.

Наилучший вариант совместной работы DNS и WINS из известных нам выглядит следующим образом. Все данные WINS-отображений помещаются в отдельную зону, например *wins.movie.edu*. Все DNS-серверы зоны *wins.movie.edu* являются серверами Microsoft DNS, а сама зона *wins.movie.edu* содержит только SOA-запись, NS-записи и WINS-запись, указывающую на серверы WINS для *wins.movie.edu*. В этом случае не существует проблемы неодинаковых ответов различных DNS-серверов, авторитативных для зоны.

Данные обратного отображения, само собой, не могут быть разнесены в различные зоны, находящиеся под управлением серверов BIND и Microsoft DNS. Поэтому, если требуется иметь обычное обратное отображение, основанное на использовании PTR-записей, а также обратное отображение на основе записей WINS-R, придется размещать зоны обратного отображения только на серверах Microsoft DNS.

Другая проблема состоит в том, что формат записей WINS и WINS-R не является стандартным. DNS-серверы BIND не понимают их и, более того, вторичный DNS-сервер, получающий WINS-записи от первичного DNS-мастер-сервера Microsoft DNS, не сможет загрузить эту зону, поскольку WINS является неизвестным типом. (Этот вопрос и обходной путь мы обсуждали в главе 14 «Разрешение проблем DNS и BIND».)

Решением этих проблем является функциональность BIND, связанная со стандартными динамическими обновлениями DNS. Впервые они появились в BIND 8 и описаны нами в главе 10; динамические обновления поддерживаются операционной системой Windows 2000. Динамические обновления позволяют производить авторизованное удаление и добавление записей на DNS-сервере BIND, а это предоставляет ребятам из Microsoft функциональность, необходимую для использования DNS в качестве службы имен для NetBIOS. Поэтому без лишних слов мы переходим к...

---

[1] Помимо этого, несколько коммерческих продуктов, таких как Meta IP/DNS от MetaInfo (порт BIND 8 со добавленными механизмами WINS). В большинстве случаев BIND неспособен общаться с WINS-серверами.

DNS и Windows 2000

Система Windows 2000 способна использовать стандартные динамические обновления для регистрации узлов в DNS. Для клиента Windows 2000 *регистрация* означает добавление отображения имени в адрес и отображения адреса в имя для этого клиента, то есть информации, которую раньше Windows-клиенты регистрировали на WINS-серверах. Для сервера Windows 2000 регистрация включает добавление в зону записей, которые показывают клиенту, какие службы и на каких портах доступны. К примеру, контроллер домена Windows 2000 использует динамические обновления для добавления SRV-записей, сообщающих клиентам Windows 2000, как получить доступ к службе Kerberos домена Windows 2000.

Использование динамических обновлений системой Windows 2000

Так что же добавляется при регистрации клиента? Перезагрузим клиент на системе Windows 2000 в лаборатории Special Effects – и посмотрим.

Наш клиент называется *mummy.fx.movie.edu*. У него фиксированный IP-адрес 192.253.254.13 (он не получает адрес от нашего DHCP-сервера). При загрузке системы подпрограммы динамического обновления на клиенте выполняют следующие шаги:

1. Поиск SOA-записи для *mummy.fx.movie.edu* на локальном DNS-сервере. SOA-запись для этого доменного имени не существует, но раздел авторитативности ответного сообщения содержит SOA-запись зоны, в которую входит *mummy.fx.movie.edu*, а именно – *fx.movie.edu*.
2. Поиск адреса DNS-сервера, указанного в поле MNAME SOA-записи, – *bladerunner.fx.movie.edu*.
3. Отправка динамического обновления на *bladerunner.fx.movie.edu* на предмет проверки выполнения двух условий: *mummy.fx.movie.edu* не является псевдонимом (то есть не имеет связанной CNAME-записи) и не имеет адресной записи, которая указывает на адрес 192.253.254.13. Динамическое обновление не содержит раздела обновления, это просто зондирование.
4. Если имя *mummy.fx.movie.edu* уже связано со своим адресом, процедура прекращается. В противном случае отправляется еще одно динамическое обновление на *bladerunner.fx.movie.edu* на предмет проверки выполнения двух условий: *mummy.fx.movie.edu* не является псевдонимом и не имеет связанной адресной записи. Если условия удовлетворяются, в обновление добавляется адресная запись, связывающая имя *mummy.fx.movie.edu* с адресом 192.253.254.13. Если у *mummy.fx.movie.edu* уже есть адресная запись, клиент посылает обновление, удаляющее ее, а затем добавляет свою.

5. Поиск SOA-записи для *254.253.192.in-addr.arpa*.
6. Поиск адреса DNS-сервера, указанного в поле MNAME SOA-записи (поле MNAME содержит *bladerunner.fx.movie.edu*, поиск записи производился только что, а в Windows 2000 используется кэширующий клиент, поэтому вторичный запрос производиться не должен).
7. Отправка динамического обновления узлу *bladerunner.fx.movie.edu* на предмет проверки выполнения условия, что *13.254.253.192.in-addr.arpa* не является псевдонимом. Если условие выполняется, с помощью обновления создается PTR-запись для обратного отображения адреса 192.253.254.13 в имя *mummy.fx.movie.edu*. Если *13.254.253.192.in-addr.arpa* является псевдонимом, процедура завершается.

Если бы мы использовали Microsoft DHCP Server из состава Windows 2000, то DHCP-сервер добавил бы PTR-запись по умолчанию. В MMC-интерфейсе управления DHCP-сервером существует также возможность предписать DHCP-серверу добавление как PTR-записи, так и A-записи. Но если бы DHCP добавил A-запись, то не проверял бы предварительные условия.

Серверы, в особенности контроллер домена Windows 2000, регистрируют большое количество информации в DNS, используя динамические обновления, как в процессе начальной установки, так и регулярно в процессе работы. (Например, служба *netlogon* регистрирует свои SRV-записи *каждый час*!) Это позволяет клиентам обнаруживать службы, не зная точно, на каких узлах и портах они работают. Мы только что создали домен Windows 2000 с именем *fx.movie.edu*, давайте взглянем на записи, которые добавил наш контроллер домена, *matrix.fx.movie.edu*:

```
$ORIGIN fx.movie.edu.
@                          600       A        192.253.254.14
_kerberos._tcp.dc._msdcs   600       SRV      0 100 88   matrix.fx.movie.edu.
_ldap._tcp.dc._msdcs       600       SRV      0 100 389  matrix.fx.movie.edu.
_ldap._tcp.e437709a-1862-11d3-8eda-00400536c213.domains._msdcs 600  SRV  0
100 389 matrix.fx.movie.edu.
e4377099-1862-11d3-8eda-00400536c213._msdcs  600  CNAME
matrix.fx.movie.edu.
gc._msdcs                  600       A        192.253.253.14
_ldap._tcp.gc._msdcs       600       SRV      0 100 3268 matrix.fx.movie.edu.
_ldap._tcp.pdc._msdcs      600       SRV      0 100 389  matrix.fx.movie.edu.
_gc._tcp                   600       SRV      0 100 3268 matrix.fx.movie.edu.
_kerberos._tcp             600       SRV      0 100 88   matrix.fx.movie.edu.
_kpasswd._tcp              600       SRV      0 100 464  matrix.fx.movie.edu.
_ldap._tcp                 600       SRV      0 100 389  matrix.fx.movie.edu.
_kerberos._udp             600       SRV      0 100 88   matrix.fx.movie.edu.
_kpasswd._udp              600       SRV      0 100 464  matrix.fx.movie.edu.
```

Ого! Сколько их!

Эти записи сообщают клиентам Windows 2000 координаты служб, предлагаемых контроллером домена, включая Kerberos и LDAP.[1] Из SRV-записей можно видеть, что все службы работают на узле *matrix.fx.movie.edu*, нашем единственном контроллере домена. Если бы у нас было два контроллера, число SRV-записей также удвоилось бы.

Имена владельцев всех SRV-записей заканчиваются именем домена Windows 2000, *fx.movie.edu*. Если бы мы назвали домен Windows 2000 *effects.movie.edu*, подпрограммы динамического обновления обновили бы зону, содержащую доменное имя *effects.movie.edu*, а именно *movie.edu*. Само собой, это привело бы к появлению беспорядка в зоне *movie.edu*, поскольку она включает другие делегированные поддомены, работающие на Windows 2000. Поэтому мы дали домену Windows 2000 имя, соответствующее имени зоны.

Проблемы совместного использования Windows 2000 и BIND

Решение Microsoft сменить WINS на DNS было очень благородным, но реализация представляет некоторые проблемы для тех, кто использует DNS-серверы BIND. Во-первых, клиенты Windows 2000 и DHCP-серверы имеют отвратительную привычку удалять адресные записи, связанные с тем же доменным именем, что у клиентов или серверов. К примеру, если мы разрешим пользователям в лаборатории спецэффектов настраивать свои компьютеры и выбирать для этих компьютеров имена, а после этого одному из пользователей придет в голову назвать свою машину уже существующим именем, скажем, именем одного из четырех серверов обсчета графики, его компьютер постарается удалить конфликтующую адресную запись (адресную запись сервера обсчета) и добавить собственную. Это, прямо скажем, не очень спортивно.

К счастью, такое поведение можно исправить на стороне клиента. В действительности клиент проверяет, существует ли адрес для доменного имени, устанавливая условие в шаге 4. (Просто по умолчанию эта адресная запись удаляется, если она уже существует.) Можно последовать инструкциям, которые приводятся в статье Q246804 базы знаний Microsoft (Microsoft Knowledge Base), и объяснить клиенту, что удалять конфликтующие записи не стоит. Результат? Клиент не может отличить адрес, используемый другим узлом с таким же доменным именем, и адрес, который ранее принадлежал этому узлу, поэтому при смене адреса на узле, клиент не производит автоматического обновления зоны.

Если вы решите позволить DHCP-серверу заботиться о регистрации, то с конфликтующими адресами придется повозиться. DHCP-сервер

[1] Объяснение функциональности каждой из этих записей можно найти в статье Q178169 базы знаний Microsoft.

не использует проверку условий для выявления конфликтов, а просто безжалостно удаляет конфликтующие адресные записи.

Учитывая ограничения, которые связаны с проведением регистрации DHCP-сервером, с какой стати вообще использовать его в таком качестве? Потому что если разрешить любому клиенту регистрироваться самостоятельно, имея для авторизации динамических обновлений только примитивные списки управления доступом, основанные на фильтрации IP-адресов, это будет равносильно разрешению динамического обновления зоны *любым клиентским адресом*. Смекалистые пользователи таких клиентов могут с легкостью выполнить несколько продуманных динамических обновлений с целью изменения MX-записей или адреса вашего DNS-сервера.

Безопасные динамические обновления

Неужели Microsoft мирится с подобными проблемами? Разумеется нет, только не с сервером Microsoft DNS. Сервер Microsoft DNS поддерживает GSS-TSIG, диалект механизма TSIG (который мы исследовали в главе 11 «Безопасность»). Клиент, использующий GSS-TSIG, получает TSIG-ключ от сервера Kerberos, а затем использует его для подписи динамических обновлений. Использование GSS (Generic Security Service, Обобщенной службы безопасности) для получения ключа означает, что администратор избавлен от необходимости вручную создавать ключ на каждом из клиентов.

Поскольку имя TSIG-ключа, используемого клиентом для подписи обновлений, совпадает с доменным именем этого клиента, DNS-сервер может проверить, что удаление адреса производится клиентом, ранее добавившим его, – просто отслеживая доменное имя TSIG-ключа, использованного для добавления записи. Удалить эту запись будет разрешено только клиенту, использующему тот же TSIG-ключ для обновления.

Клиенты Windows 2000 пытаются произвести обновление, подписанное GSS-TSIG-ключом, в том случае, если в выполнении обычного динамического обновления было отказано. Их также можно настроить таким образом, чтобы сразу выполнялись подписанные обновления. Для этого следует выполнить инструкции, содержащиеся в статье Q246804 базы знаний Microsoft, которая упоминалась ранее.

BIND и GSS–TSIG

К сожалению, DNS-серверы BIND пока еще не поддерживают механизм GSS-TSIG, поэтому невозможно использовать безопасные динамические обновления Windows 2000 совместно с BIND. В одной из следующих версий планируется поддержка GSS-TSIG. Когда она появится, можно использовать все правила регулировки обновлений, описанные в главе 10, для определения, какими должны быть ключи для различных типов записей. Простого набора правил:

```
zone "fx.movie.edu" {
    type master;
    file "db.fx.movie.edu";
    update-policy {
        grant *.fx.movie.edu. self *.fx.movie.edu. A;
        grant matrix.fx.movie.edu. self matrix.fx.movie.edu. ANY;
        grant matrix.fx.movie.edu. subdomain fx.movie.edu. SRV;
    };
};
```

может быть вполне достаточно, чтобы разрешить клиентам и серверам Windows 2000 регистрировать нужные данные в зоне *fx.movie.edu*.

Как быть?

Но как сейчас справиться с умножением числа машин сети, работающих под управлением Windows 2000? Microsoft посоветует вам «модернизировать» все DNS-серверы до сервера Microsoft DNS в версии для Windows 2000. Но если вам нравится BIND – так же, как и нам – то, скорее всего, вы предпочтете иные варианты.

Работа с клиентами Windows 2000

Первый (и, видимо, самый распространенный) способ мирно работать с клиентами Windows 2000 – создать делегированный поддомен, в котором все они будут жить. Мы могли бы назвать его *win.fx.movie.edu*. В пределах *win.fx.movie.edu* допустимо все: клиенты могут не обращать внимания на адреса других клиентов, посылать созданные вручную динамические обновления и добавлять фальшивые записи к данным зоны. Идея заключатся в том, чтобы создать рабочее пространство (или тюремную камеру, кому что нравится), за пределы которого клиенты не могут выбраться и в пределах которого вольны делать все что угодно. Те из читателей, у кого есть дети, поймут идею интуитивно.

По умолчанию клиент Windows 2000 пытается зарегистрироваться в зоне прямого отображения, имеющее такое же имя, как домен Windows 2000, в который входит клиент. Поэтому придется пройти через дополнительные настройки, чтобы объяснить клиентам, что следует регистрироваться в зоне *win.fx.movie.edu*, а не *fx.movie.edu*. В частности, необходимо открыть окно, расположенное по адресу *My Computer->Properties->Network Identification->Properties->More*, отключить режим *Change primary DNS suffix when domain membership changes*, а затем набрать *win.fx.movie.edu* в поле, озаглавленном *Primary DNS suffix of this computer*. Операцию повторить для *всех* клиентов.

Второй вариант – оставить всех клиентов в основной рабочей зоне (для нашей лаборатории это *fx.movie.edu*), но разрешить динамические обновления только с адреса DHCP-сервера. Затем необходимо настроить сервер DHCP на сопровождение A-записей и PTR-записей. (Для узлов, не использующих DHCP, A- и PTR-записи можно добавить вручную.)

В последнем случае задача посылки собственных динамических обновлений становится для маленьких чертенят не такой простой, поскольку для ее решения необходима подделка IP-пакетов. Однако ситуация, когда создается клиент с доменным именем, которое конфликтует с уже существующим в зоне, по-прежнему возможна.

Работа с серверами Windows 2000

Главный сервер, с которым нужно справиться, – контроллер домена (или контроллеры, если их более одного). Контроллер домена, как мы уже рассказывали, желает добавить целую пачку SRV-записей. Если это невозможно на момент установки, записи в формате мастер-файла записываются в файл с именем *System32\Config\netlogon.dns* в корневом каталоге системы.

Во-первых, следует определить, какую зону необходимо обновить. Это вопрос выбора зоны, которая будет владеть доменным именем Windows 2000. Если домен Windows 2000 имеет имя, совпадающее с именем существующей зоны, то, разумеется, эту зону и следует обновлять. В противном случае следует удалять метки из начала домена Windows 2000 по одной, пока не получится доменное имя одной из зон.

Узнав, какую зону необходимо обновить, следует решить, как это делать. Если вы достаточно доверяете контроллеру домена, чтобы разрешить динамическое обновление зоны, просто добавьте соответствующее предписание *allow-update* в оператор *zone*, и дело сделано. В противном случае можно не разрешать динамическое обновление и воспользоваться тем, что контроллер создает файл *netlogon.dns*. Используем директива $INCLUDE для включения содержимого этого файла в файл данных зоны:

```
$INCLUDE netlogon.dns
```

Предположим, не подходит ни один из этих вариантов, поскольку нет желания разрешать контроллеру перекапывать зону, но существует необходимость в том, чтобы он мог изменять свои SRV-записи. На этот случай у нас есть туз в рукаве. Можно воспользоваться преимуществом забавных имен владельцев SRV-записей и создать делегированные поддомены с именами (для нашего случая) *_udp.fx.movie.edu*, *_tcp.fx.movie.edu*, *_sites.fx.movie.edu* и *_msdcs.fx.movie.edu*. Придется отключить проверку имен для *_msdcs.fx.movie.edu*, поскольку контроллер домена захочет добавить адресную запись к этой зоне, помимо уймы SRV-записей. Теперь разрешим контроллеру динамическое обновление этих зон, но не основной зоны:

```
acl dc { 192.253.254.13; };

zone "_udp.fx.movie.edu" {
    type master;
    file "db._udp.fx.movie.edu";
```

```
    allow-update { dc; };
};

zone "_tcp.fx.movie.edu" {
    type master;
    file "db._tcp.fx.movie.edu";
    allow-update { dc; };
};

zone "_sites.fx.movie.edu" {
    type master;
    file "db._udp.fx.movie.edu";
    allow-update { dc; };
};

zone "_msdcs.fx.movie.edu" {
    type master;
    file "db._udp.fx.movie.edu";
    allow-update { dc; };
    check-names ignore;
};
```

Итак, мы получили лучшее двух миров: динамическую регистрацию служб и безопасную рабочую зону.

A

Формат сообщений DNS и RR-записей

В этом приложении описан формат сообщений DNS, а также каждой в отдельности RR-записи. RR-записи приводятся в текстовом формате, то есть в том виде, в каком они приводятся в файле данных зоны, и в двоичном формате, который используется в сообщениях DNS. Здесь присутствуют описания нескольких записей, которые не упоминаются в книге, поскольку являются экспериментальными либо вышедшими из употребления.

Мы включили части документа RFC 1035 за авторством Пола Мокапетриса, которые относятся к текстовому формату мастер-файлов (файлов, которые мы называем *файлами данных зоны*) или к формату сообщений DNS (это для тех, кому понадобится разбирать DNS-пакеты).

Формат мастер-файла

(Из документа RFC 1035, стр. 33–35)

Формат этих файлов представляет собой последовательность записей. Записи являются преимущественно строко-ориентированными, хотя для создания многострочных записей можно использовать круглые скобки, а текстовые литералы могут содержать символы CRLF. Любое сочетание символов табуляции и пробелов является разделителем отдельных компонентов, составляющих запись. Любая строка в главном файле может заканчиваться комментарием. Комментарий начинается с символа точки с запятой (;).

Определены следующие записи:

```
blank[comment]

$ORIGIN domain-name [comment]

$INCLUDE file-name domain-name [comment]

domain-namerr [comment]

blankrr [comment]
```

Пустые строки, с комментариями или без, допускаются в любом месте файла.

Определены две директивы: $ORIGIN и $INCLUDE. Директива $ORIGIN в качестве аргумента воспринимает доменное имя и переустанавливает текущий суффикс по умолчанию для относительных доменных имен в указанное значение (*domain-name*). Директива $INCLUDE вставляет указанный файл в текущий и может также содержать указание доменного имени, которое будет являться относительным доменным суффиксом по умолчанию для включаемого файла. Директива $INCLUDE также может дополняться комментарием. Обратите внимание, что запись $INCLUDE не изменяет относительного суффикса по умолчанию для записей содержащего (родительского) файла, вне зависимости от того, как изменяется относительный суффикс по умолчанию в пределах включаемого файла.

Последние два варианта относятся к записям ресурсов (RR-записям, RRs). Если файловая запись для RR-записи начинается с пробела, эта запись считается относящейся к последнему упомянутому имени. Если RR-запись начинается с доменного имени (*domain-name*), то считается относящейся к этому имени.

Содержимое *RR*-записи может принимать одну из форм:

```
[TTL] [class] type RDATA
[class] [TTL] type RDATA
```

RR-запись начинается с необязательных полей TTL и класса, за которыми следуют поля типа и RDATA, соответствующие типу и классу. Для записи класса и типа используется стандартная мнемоника, TTL представляется десятеричным целым числом. Пропущенные значения класса и TTL автоматически устанавливаются в последние упомянутые значения класса и TTL. Поскольку мнемоники типов и классов не пересекаются, интерпретация всегда однозначна.

Доменные имена составляют основу данных мастер-файла. Метки в доменных именах представляются символьными строками и разделяются символом точки. Правила маскировки позволяют использовать любые символы в доменных именах. Доменные имена, заканчивающиеся точкой, называются абсолютными и интерпретируются как полные. Доменные имена, которые не заканчиваются точкой, называются относительными; реальное доменное имя является результатом сращения относительной части с суффиксом по умолчанию, указан-

ным в директивах $ORIGIN или $INCLUDE либо в качестве аргумента для подпрограммы загрузки мастер-файла. Использование относительного имени является ошибкой в том случае, когда суффикс по умолчанию не определен.

Символьная строка (*character-string*) может быть представлена одним из двух способов: в виде непрерывной последовательности символов, не включающей пробелы, либо в виде последовательности символов, начинающейся с символа " и заканчивающейся символом ". В пределах строки, заключенной в кавычки, допустимы любые символы, кроме собственно символа ", который для включения в строку должен маскироваться символом обратного слэша (\).

Поскольку описываемые файлы являются текстовыми, необходимо определить несколько специальных способов кодирования, позволяющих загружать произвольные данные. В частности:

. От корня.

@ Отдельно стоящий символ @ используется для обозначения текущего суффикса по умолчанию.

\X

Где X – произвольный символ (но не цифра от 0 до 9), а символ \ используется для маскировки символа и отменяет его специальный смысл. К примеру, последовательность \. может использоваться для включения символа точки в метку.[1]

\DDD

Где каждая буква D является одной из трех цифр октета, соответствующего восьмеричному числу, представленному в виде DDD. Полученный октет считается текстовым и не подвергается дальнейшей интерпретации.[2]

( ) Скобки используются для группировки данных, которые записываются в несколько строк. По сути дела, в пределах круглых скобок не происходит распознавание конца строки.[3]

; Точка с запятой является началом комментария, комментарий ограничен концом строки и никогда не интерпретируется.

Регистр символов

(Из документа RFC 1035, стр. 9)

Для всех составляющих DNS, которые входят в официальный протокол, любые сравнения для текстовых строк (например, меток, домен-

---

[1] В BIND версии 4.8.3 возможность не реализована.

[2] В BIND версии 4.8.3 возможность не реализована.

[3] В BIND версии 4.8.3 разрешается использование скобок только в SOA- и WKS-записях.

ных имен и т. д.) производятся без учета регистра символов. В настоящее время это правило имеет силу для всей DNS, без каких-либо исключений. Однако развитие системы за пределы существующих возможностей может привести к необходимости использования полных двоичных октетов в именах, поэтому следует избегать хранения доменных имен в 7-битном ASCII-формате, использования специальных символов для завершения меток и тому подобных вещей.

Типы

Вот полный перечень типов RR-записей. Текстовое представление используется в мастер-файлах. Двоичное представление используется в DNS-запросах и DNS-ответах. Эти RR-записи описаны на стр. 13–21 документа RFC 1035.

A address

(Из документа RFC 1035, стр. 20)

Текстовое представление:

```
owner ttl class A address
```

Пример:

```
localhost.movie.edu.   IN A 127.0.0.1
```

Двоичное представление:

```
Код типа адреса: 1
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
     |                    ADDRESS                    |
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
где:
ADDRESS           - 32-битный адрес сети Интернет.
```

CNAME canonical name

(Из документа RFC 1035, стр. 14)

Текстовое представление:

```
owner ttl class CNAME canonical-dname
```

Пример:

```
wh.movie.edu.  IN  CNAME  wormhole.movie.edu.
```

Двоичное представление:

```
Код типа CNAME: 5
```

```
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
     /                     CNAME                     /
     /                                               /
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
где:
CNAME           - доменное имя (domain-name), которое определяет
                  каноническое или основное имя владельца.
                  Имя владельца (owner) является псевдонимом.
```

HINFO host information

(Из документа RFC 1035, стр. 14)

Текстовое представление:

```
owner ttl class HINFO cpu os
```

Пример:

```
grizzly.movie.edu.  IN  HINFO  VAX-11/780 UNIX
```

Двоичное представление:

```
Код типа HINFO: 13
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
     /                      CPU                      /
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
     /                       OS                      /
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
где:
CPU             - символьная строка (character-string),
                  определяющая тип процессора.
OS              - символьная строка (character-string),
                  определяющая тип операционной системы.
```

MB mailbox domain name (экспериментальная)

(Из документа RFC 1035, стр. 14)

Текстовое представление:

```
owner ttl class MB mbox-dname
```

Пример:

```
al.movie.edu.  IN  MB  robocop.movie.edu.
```

Двоичное представление:

```
Код типа MB: 7
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
     /                   MADNAME                     /
     /                                               /
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

```
где:
MADNAME         - доменное имя (domain-name), определяющее узел,
                  с которым ассоциируется указанный почтовый ящик.
```

MD mail destination (не используется)

Вместо MD-записей используются MX-записи.

MF mail forwarder (не используется)

Вместо MF-записей используются MX-записи.

MG mail group member (экспериментальная)

(Из документа RFC 1035, стр. 16)

Текстовое представление:

```
owner ttl class MG mgroup-dname
```

Пример:

```
admin.movie.edu.  IN  MG  al.movie.edu.
                  IN  MG  ed.movie.edu.
                  IN  MG  jc.movie.edu.
```

Двоичное представление:

```
Код типа MG: 8
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
     /                   MGMNAME                     /
     /                                               /
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
где:
MGMNAME         - доменное имя (domain-name), определяющее почтовый ящик,
                  входящий в почтовую группу, определяемую доменным именем.
```

MINFO mailbox or mail list information (экспериментальная)

(Из документа RFC 1035, стр. 16)

Текстовое представление:

```
owner ttl class MINFO resp-mbox error-mbox
```

Пример:

```
admin.movie.edu.  IN  MINFO  al.movie.edu. al.movie.edu.
```

Двоичное представление:

```
Код типа MINFO: 14
```

```
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
     /                    RMAILBX                    /
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
     /                    EMAILBX                    /
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
где:
RMAILBX         - доменное имя (domain-name), определяющее почтовый ящик,
                  принимающий ответственность за список рассылки
                  или почтовый ящик. Если доменное имя определяет корень,
                  владелец записи MINFO (owner) отвечает за себя сам.
                  Следует помнить, что многие существующие списки рассылки
                  используют формат вида X-request для поля RMAILBX
                  списка рассылки X, например, Msgroups-request для списка
                  Msgroups. Это поле предоставляет более общий механизм.
EMAILBX         - доменное имя (domain-name), определяющее почтовый ящик,
                  который должен получать сообщения об ошибках, связанных
                  со списком рассылки или почтовым ящиком, определяемыми
                  владельцем записи MINFO (аналогично предлагавшемуся полю
                  ERRORS-TO). Если доменное имя определяет корень, ошибки
                  должны возвращаться отправителю письма.
```

MR mail rename (экспериментальная)

(Из документа RFC 1035, стр. 17)

Текстовое представление:

```
owner ttl class MR new-mbox
```

Пример:

```
eddie.movie.edu.  IN  MR  eddie.bornagain.edu.
```

Двоичное представление:

```
Код типа MR: 9
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
     /                    NEWNAME                    /
     /                                               /
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
где:
NEWNAME           - доменное имя (domain-name), определяющее почтовый ящик,
                    обозначающий переименование для указанного.
```

MX mail exchanger

(Из документа RFC 1035, стр. 17)

Текстовое представление:

```
owner ttl class MX preference exchange-dname
```

Пример:

```
ora.com.  IN  MX  0  ora.ora.com.
          IN  MX  10 ruby.ora.com.
          IN  MX  10 opal.ora.com.
```

Двоичное представление:

```
Код типа MX: 15
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
     |                  PREFERENCE                   |
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
     /                   EXCHANGE                    /
     /                                               /
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
где:
PREFERENCE      - 16-битное целое, определяющее приоритет этой записи среди
                  записей для одного имени. Больший приоритет
                  имеют меньшие значения.
EXCHANGE        - доменное имя (domain-name), определяющее узел, который
                  будет выступать в роли почтового ретранслятора
                  для доменовладельца записи.
```

NS name server

(Из документа RFC 1035, стр. 18)

Текстовое представление:

```
owner ttl class NS name-server-dname
```

Пример:

```
movie.edu.  IN   NS  terminator.movie.edu
```

Двоичное представление:

```
Код типа NS: 2
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
     /                   NSDNAME                     /
     /                                               /
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
где:
NSDNAME         - доменное имя (domain-name), определяющее узел,
                  который следует считать авторитативным
                  для указанного класса и домена.
```

NULL null (экспериментальная)

(Из документа RFC 1035, стр. 17)

Двоичное представление:

```
Код типа NULL: 10
```

```
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
     /                    что угодно                             /
     /                                                           /
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Поле RDATA может содержать какие угодно данные, если его длина не превышает 65535 октетов.

NULL-записи не реализованы в BIND.

PTR pointer

(Из документа RFC 1035, стр. 18)

Текстовое представление:

owner ttl class PTR *dname*

Пример:

```
1.249.249.192.in-addr.arpa.  IN PTR wormhole.movie.edu.
```

Двоичное представление:

```
Код типа PTR: 12
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
     /                   PTRDNAME                                /
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
где:
PTRDNAME        - доменное имя (domain-name), указывающее
                  на определенную точку пространства доменных имен.
```

SOA start of authority

(Из документа RFC 1035, стр. 19–20)

Текстовое представление:

owner ttl class SOA *source-dname mbox* (*serial refresh retry expire minimum*)

Пример:

```
movie.edu. IN SOA terminator.movie.edu. al.robocop.movie.edu. (
                       1        ; Порядковый номер
                       10800    ; Обновление через 3 часа
                       3600     ; Повторение попытки через 1 час
                       604800   ; Устаревание через 1 неделю
                       86400 )  ; Минимальное TTL в 1 день
```

Двоичное представление:

```
Код типа SOA: 6
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    /                     MNAME                     /
    /                                               /
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    /                     RNAME                     /
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                    SERIAL                     |
    |                                               |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                    REFRESH                    |
    |                                               |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                     RETRY                     |
    |                                               |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                    EXPIRE                     |
    |                                               |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                    MINIMUM                    |
    |                                               |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
где:
MNAME           - доменное имя DNS-сервера, который является изначальным
                  или первичным источником данных для этой зоны.
RNAME           - доменное имя, определяющее почтовый ящик человека,
                  ответственного за эту зону.
SERIAL          - положительный 32-битный номер версии исходной копии зоны.
                  Значение сохраняется при передаче зоны. Значение обнуляется
                  (wraps) и должно обрабатываться с использованием
                  непрерывного арифметического пространства.
REFRESH         - 32-битный временной интервал обновления зоны.
RETRY           - 32-битный временной интервал повторения попытки обновления.
EXPIRE          - 32-битный временной интервал истечения авторитативности зоны.
MINIMUM         - положительное 32-битное значение минимального времени
                  жизни, которое должно экспортироваться в составе любой
                  записи ресурсов из зоны.
```

TXT text

(Из документа RFC 1035, стр. 20)

Текстовое представление:

owner ttl class TXT *txt-strings*

Пример:

```
cujo.movie.edu.  IN  TXT  "Location: machine room dog house"
```

Двоичное представление:

```
Код типа TXT: 16
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    /                   TXT-DATA                    /
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
где:
TXT-DATA        - одна или несколько символьных строк.
```

WKS well-known services

(Из документа RFC 1035, стр. 21)

Текстовое представление:

```
owner ttl class WKS address protocol service-list
```

Пример:

```
terminator.movie.edu.  IN  WKS 192.249.249.3  TCP ( telnet smtp
                                              ftp shell domain )
```

Двоичное представление:

```
Код типа WKS: 11
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                    ADDRESS                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |       PROTOCOL        |                       |
    +--+--+--+--+--+--+--+--+                       |
    |                                               |
    /                     BIT MAP                   /
    /                                               /
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
где:
ADDRESS         - 32-битный адрес Интернета.
PROTOCOL        - 8-битный номер IP-протокола.
BIT MAP         - бит-карта переменной длины. Длина бит-карты должна быть
                  кратной восьми битам.
```

Новые типы по документу RFC 1183

AFSDB Andrew File System Data Base (экспериментальная)

Текстовое представление:

```
owner ttl class AFSDB subtype hostname
```

Пример:

```
fx.movie.edu.  IN  AFSDB  1 bladerunner.fx.movie.edu.
               IN  AFSDB  2 bladerunner.fx.movie.edu.
```

```
                IN  AFSDB  1 empire.fx.movie.edu.
                IN  AFSDB  2 aliens.fx.movie.edu.
```

Двоичное представление:

```
Код типа AFSDB: 18
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
     |                  SUBTYPE                      |
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
     /                  HOSTNAME                     /
     /                                               /
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
где:
SUBTYPE        - подтип 1 соответствует серверу базы данных AFS для клетки.
                 Подтип 2 это DNS-сервер с DCE-идентификацией.
HOSTNAME       - доменное имя, которое определяет узел, на котором работает
                 сервер для клетки, идентифицируемой владельцем (owner) записи.
```

ISDN Integrated Services Digital Network address (экспериментальная)

Текстовое представление:

owner ttl class ISDN *ISDN-address sa*

Пример:

```
delay.hp.com.    IN  ISDN  141555514539488
hep.hp.com.      IN  ISDN  141555514539488 004
```

Двоичное представление:

```
Код типа ISDN: 20
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
     /                ISDN ADDRESS                   /
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
     /                 SUBADDRESS                    /
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
где:
ISDN ADDRESS    - символьная строка, содержащая ISDN-номер владельца (owner)
                  владельца (owner) и DDI-номер (Direct Dial In),
                  если таковой существует.
SUBADDRESS      - необязательная символьная строка,
                  содержащая уточняющий адрес.
```

RP Responsible Person (экспериментальная)

Текстовое представление:

owner ttl class RP *mbox-dname txt-dname*

Пример:

```
; Текущий суффикс по умолчанию - fx.movie.edu
```

```
@            IN  RP   ajs.fx.movie.edu.   ajs.fx.movie.edu.
bladerunner  IN  RP   root.fx.movie.edu.  hotline.fx.movie.edu.
             IN  RP   richard.fx.movie.edu.  rb.fx.movie.edu.
ajs          IN  TXT  "Arty Segue, (415) 555-3610"
hotline      IN  TXT  "Movie U. Network Hotline, (415) 555-4111"
rb           IN  TXT  "Richard Boisclair, (415) 555-9612"
```

Двоичное представление:

```
Код типа RP: 17
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
     /                    MAILBOX                    /
     /                                               /
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
     /                    TXTDNAME                   /
     /                                               /
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
где:
MAILBOX          - доменное имя, определяющее почтовый ящик
                   для ответственного лица.
TXTDNAME         - доменное имя, для которого существуют TXT-записи.
                   Для получения этих TXT-записей могут быть выполнены
                   дополнительные запросы для txt-dname
```

RT Route Through (экспериментальная)

Текстовое представление:

owner ttl class RT *preference intermediate-host*

Пример:

```
sh.prime.com.  IN  RT  2   Relay.Prime.COM.
               IN  RT  10  NET.Prime.COM.
```

Двоичное представление:

```
Код типа RT: 21
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
     |                  PREFERENCE                   |
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
     /                 INTERMEDIATE                  /
     /                                               /
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
где:
PREFERENCE    - 16-битное целое, определяющее приоритет данной записи
                по сравнению с прочими записями для того же имени.
                Больший приоритет имеют меньшие значения.
EXCHANGE      - доменное имя, определяющее узел, который будет выступать про-
                межуточным при необходимости добраться до указанного (owner).
```

X25 X.25 address (экспериментальная)

Текстовое представление:

```
owner ttl class X25 PSDN-address
```

Пример:

```
relay.pink.com.  IN  X25   31105060845
```

Двоичное представление:

```
Код типа X25: 19
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
     /                 PSDN ADDRESS                  /
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
где:
PSDN ADDRESS     - символьная строка, идентифицирующая ассоциируемый
                   с владельцем (owner) адрес PSDN (Public Switched Data
                   Network) в плане нумерации X.121.
```

Новые типы по документу RFC 1664

PX pointer to X.400/RFC 822 mapping information

Текстовое представление:

```
owner ttl class PX preference RFC822 address X.400 address
```

Пример:

```
ab.net2.it.  IN  PX  10   ab.net2.it.  O-ab.PRMD-net2.ADMDb.C-it.
```

Двоичное представление:

```
Код типа PX: 26
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                 PREFERENCE                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    /                   MAP822                      /
    /                                               /
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    /                   MAPX400                     /
    /                                               /
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
где:
PREFERENCE     - 16-битное целое, определяющее приоритет данной записи
                 по сравнению с прочими записями для того же имени.
                 Больший приоритет имеют меньшие значения.
MAP822         - элемент доменного имени, состоящий из rfc822-domain,
                 части (по RFC 822) информации отображения согласно
                 документу RFC 1327.
```

MAPX400 – элемент доменного имени, содержащий значение *x400-in-domain-syntax*, производное от X.400-части информации отображения согласно документу RFC 1327.

Классы

(Из документа RFC 1035, стр. 13)

Поле CLASS присутствует в записях ресурсов. Определены следующие мнемоники и значения:

IN 1: класс Интернет

CS 2: класс CSNET (вышел из употребления, используется только в примерах устаревших документов RFC)

CH 3: класс CHAOS

HS 4: класс Hesiod

Сообщения DNS

Чтобы писать программы, разбирающие сообщения DNS, следует понимать формат этих сообщений. Запросы и ответы DNS чаще всего содержатся в UDP-пакетах. Каждое сообщение полностью содержится в одном UDP-пакете. Если запрос и ответ посылаются через TCP, к ним добавляются двухбайтовые префиксы, указывающие на размер запроса или ответа, без учета собственно префикса. Формат и содержание сообщений DNS описаны далее.

Формат сообщения

(Из документа RFC 1035, стр. 25)

Все взаимодействия в пределах доменного протокола происходят в пределах единственного формата, который описывает отдельное сообщение. На самом высоком уровне сообщение можно представить в виде пяти разделов (отдельные разделы в некоторых случаях могут быть пустыми):

```
+---------------------+
|      Заголовок      |
+---------------------+
|    Основная часть   | запрос к DNS-серверу
+---------------------+
|        Ответ        | RR-записи, являющиеся ответом
+---------------------+
|  Авторитативность   | RR-записи, ссылающиеся на авторитативный DNS-сервер
+---------------------+
|    Дополнительный   | RR-записи с дополнительной информацией
+---------------------+
```

Заголовок присутствует всегда. Он содержит поля, которые определяют, какие из оставшихся разделов присутствуют, а также указывает тип сообщения – запрос или ответ, стандартный запрос или код операции и т. д.

Имена разделов, следующих за заголовком, являются производными от их использования в стандартных запросах. Основной раздел содержит поля, описывающие вопрос к DNS-серверу. А именно: тип запроса (QTYPE), класс запроса (QCLASS) и доменное имя запроса (QNAME). Последние три раздела имеют одинаковый формат: список связанных RR-записей, возможно, пустой. Раздел ответа содержит записи, которые представляют собой ответ на вопрос; раздел авторитативности содержит записи, которые являются указателями на авторитативный DNS-сервер; дополнительный раздел содержит записи, которые имеют отношение к вопросу, но не являются точным ответом на него.

Формат заголовка

(Из документа RFC 1035, стр. 26–28)

```
                                    1  1  1  1  1  1
      0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                      ID                       |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |QR|   Opcode  |AA|TC|RD|RA|   Z    |   RCODE   |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                    QDCOUNT                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                    ANCOUNT                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                    NSCOUNT                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                    ARCOUNT                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
где:
ID              - 16-битный идентификатор, присвоенный программой, генерирующей
                  произвольный вид запроса. Этот идентификатор копируется
                  в соответствующий ответ и используется получателем ответа
                  для связывания получаемых ответов и посланных запросов.
QR              - однобитовое поле, определяющее, является
                  сообщение запросом (0) или ответом (1).
OPCODE          - четырехбитное поле, определяющее вид запроса в сообщении.
                  Это значение устанавливается отправителем запроса
                  и воспроизводится в ответе. Существуют следующие значения:
                  0            стандартный запрос (QUERY)
                  1            инверсный запрос (IQUERY)
                  2            запрос состояния сервера (STATUS)
                  3-15         зарезервированы
AA              - авторитативный ответ (Authoritative Answer). Этот бит
```

является действительным (valid) в получаемых ответах и определяет, является ли ответивший DNS-сервер авторитативным для доменного имени, о котором идет речь в основном разделе запроса. Следует иметь в виду, что содержимое раздела ответа может иметь несколько имен владеющего сервера из-за псевдонимов. Бит AA относится к имени, которое совпадает с именем в запросе, или к первому имени в разделе ответа.

TC – усечение (TrunCation) – указывает на усечение сообщения в результате превышения максимально доступной длины для данного способа передачи.

RD – желательна рекурсия (Recursion Desired). Бит может устанавливаться в запросе и воспроизводиться в ответе. Если бит RD установлен, это является указанием серверу имен производить рекурсивное разрешение. Поддержка рекурсивных запросов не является обязательной.

RA – рекурсия доступна (Recursion Available). Установленный или сброшенный в ответе, бит RA определяет, доступна ли поддержка рекурсивных запросов для используемого DNS-сервера.

Z – зарезервирован на будущее. Бит должен быть нулевым во всех запросах и ответах.

RCODE – код ответа. Четырехбитное поле, значение которого устанавливается в качестве части ответа. Значения интерпретируются следующим образом:

0	Ошибки нет.
1	Ошибка формата – DNS-сервер не смог интерпретировать запрос.
2	Сбой сервера – DNS-сервер не обработал запрос из-за внутренней проблемы.
3	Ошибка имени – имеет смысл только в ответах, полученных от авторитативного сервера имен; этот код означает, что доменное имя, указанное в запросе, не существует.
4	Отсутствует реализация – данный DNS-сервер не поддерживает такой вид запросов.
5	Отказано – DNS-сервер отказывается выполнять указанную операцию из соображений следования установленным для него правилам. К примеру, сервер имен может отказаться предоставлять информацию отдельно взятому клиенту либо выполнять конкретные операции (скажем, передачу зоны) для определенных данных.
6-15	Зарезервированы.

QDCOUNT – положительное 16-битное целое, определяющее число записей в разделе вопроса.

ANCOUNT – положительное 16-битное целое, определяющее число RR-записей в разделе ответа.

NSCOUNT – положительное 16-битное целое, определяющее число RR-записей серверов имен в разделе авторитативности.

ARCOUNT – положительное 16-битное целое, определяющее число RR-записей в разделе дополнительных записей.

Формат основного раздела

(Из документа RFC 1035, стр. 28–29)

Основной раздел большинства запросов, содержит собственно «вопрос», то есть параметры, определяющие, какие данные запрашиваются. Раздел содержит QDCOUNT (обычно 1) записей, каждая из которых имеет следующий формат:

```
                                    1  1  1  1  1  1
      0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                                               |
    /                     QNAME                     /
    /                                               /
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                     QTYPE                     |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                     QCLASS                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

где:

QNAME – доменное имя, представленное последовательностью меток, причем каждая метка состоит из октета длины, за которым следует указанное число октетов. Доменное имя завершается нулевым октетом длины (указанием пустой метки корня). Следует помнить, что это поле может содержать нечетное число октетов, поскольку автоматическое выравнивание (padding) не применяется.

QTYPE – код из двух октетов, определяющий тип запроса. Множество значений этого поля содержит все коды, допустимые для использования в поле TYPE, а также некоторые более общие, охватывающие более одного типа записей.

QCLASS – код из двух октетов, определяющий класс запроса. К примеру, для класса Интернет поле QCLASS имеет значение IN.

Значения QCLASS

(Из документа RFC 1035, стр. 13)

Поле QCLASS может присутствовать в основной части запроса. Набор значений для QCLASS является надмножеством значений для CLASS; каждое значение CLASS является допустимым для QCLASS. Помимо значений для CLASS определяется следующее значение для QCLASS:

* 255 Произвольный класс

Значения QTYPE

(Из документа RFC 1035, стр. 12–13)

Поле QTYPE может присутствовать в основной части запроса. Набор значений для QTYPE является надмножеством значений для TYPE,

следовательно, все TYPE-значения являются допустимыми для QTYPE. Помимо этого определены следующие значения QTYPE:

AXFR

252 Запрос передачи целой зоны

MAILB

253 Запрос записей, связанных с почтовыми ящиками (MB, MG или MR)

MAILA

254 Запрос RR-записей почтового агента (не используется, см. MX)

* 255 Запрос всех записей

Формат ответа, авторитативности и дополнительных разделов

(Из документа RFC 1035, стр. 29–30)

Разделы ответа, авторитативности и дополнительные разделы имеют общий формат: переменное число RR-записей, определяемое соответствующим полем-счетчиком в заголовке. Каждая RR-запись имеет следующий формат:

```
                                1  1  1  1  1  1
  0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                                               |
/                                               /
/                      NAME                     /
|                                               |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      TYPE                     |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                     CLASS                     |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      TTL                      |
|                                               |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                   RDLENGTH                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--|
/                     RDATA                     /
/                                               /
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

где:

NAME – доменное имя, к которому относится данная запись.

TYPE – два октета, содержащие один из кодов типов. Это поле определяет смысл данных, передаваемых в поле RDATA.

CLASS – два октета, определяющие класс данных, передаваемых в поле RDATA.

```
TTL             - 32-битное целое число, определяющее интервал времени
                  (в секундах), в течение которого разрешается
                  кэшировать запись. Нулевые значения интерпретируются таким
                  образом, что запись может использоваться только в ходе
                  текущей транзакции и не должна кэшироваться вообще.
RDLENGTH        - положительное 16-битное целое число, определяющее объем
                  данных, передаваемых в поле RDATA (в октетах).
RDATA           - строка октетов переменной длины, которая описывает ресурс.
                  Формат информации  варьируется в зависимости от параметров
                  TYPE и CLASS записи. К примеру, если TYPE имеет значение A,
                  а CLASS - значение IN, поле RDATA состоит из четырех
                  октетов, определяющих адрес ARPA/Интернет.
```

Порядок передачи данных

(Из документа RFC 1035, стр. 8–9)

Порядок передачи заголовка и данных, описанный в настоящем документе, регулируется на уровне октетов. Когда на диаграмме отображена группа октетов, порядок их передачи – это порядок их прочтения на английском языке. К примеру, октеты на следующей диаграмме передаются в том порядке, в каком они пронумерованы.

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       1       |       2       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       3       |       4       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       5       |       6       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Если октет представляет числовое значение, левый крайний бит диаграммы содержит старший или наиболее значимый бит. То есть старшим является бит с нулевым порядковым номером. К примеру, октет на следующей диаграмме содержит число 170 (десятичное).

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|1 0 1 0 1 0 1 0|
+-+-+-+-+-+-+-+-+
```

Аналогичным образом, если число представлено полем шириной в несколько октетов, старшим является крайний левый бит всего поля. При передаче такого поля первым передается старший бит.

Данные RR-записей

Формат данных

В дополнение к целым значениям, состоящих из двух и четырех октетов, данные RR-записей могут содержать *доменные имена* и *символьные строки*.

Доменное имя

(Из документа RFC 1035, стр. 10)

Доменные имена в сообщениях представляются последовательностями меток. Каждая метка содержит поле длины размером в один октет, а также указанное число символов. Поскольку каждое доменное имя заканчивается пустой меткой корня, произвольное доменное имя завершается нулевым байтом длины. Старшие два бита каждого октета длины должны быть нулевыми, оставшиеся шесть битов поля длины ограничивают длину одной метки до 63 октетов.

Упаковка сообщений

(Из документа RFC 1035, стр. 30)

Чтобы уменьшить размер сообщений, в системе доменных имен используется алгоритм сжатия, исключающий необходимость повторять доменные имена в одном сообщении. По этому алгоритму целое доменное имя или несколько меток в конце доменного имени могут заменяться ссылкой на предшествующее вхождение того же имени.

Указатель состоит из двух октетов:

```
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| 1  1|                OFFSET                   |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Первые два бита устанавливаются в единицу, что позволяет отличить указатель от метки; метка должна начинаться с двух нулевых битов, поскольку длина меток ограничена до 63 октетов. (Битовые комбинации 10 и 01 зарезервированы на будущее.) Поле OFFSET определяет смещение от начала сообщения (то есть первого октета поля ID в доменном заголовке). Нулевое смещение указывает на первый байт поля ID.

Символьная строка

(Из документа RFC 1035, стр. 13)

Символьная строка – это единственный октет длины, за которым следует указанное число символов. *Символьная строка* считается двоичными данными, ее длина не может превышать 256 символов, включая октет длины.

B

Таблица совместимости BIND

В табл. B.1 отражена поддержка разнообразных свойств различными версиями пакета BIND.

Таблица B.1. Таблица совместимости BIND

	Версия BIND			
Свойство	**4.9.7**	**8.1.2**	**8.2.3**	**9.1.0**
Поддержка нескольких процессоров				X
Динамические обновления		X	X	X
Динамические обновления с TSIG-подписями			X	X
Правила обновлений на основе TSIG-подписей				X
NOTIFY		X	X	X
Инкрементальная передача зоны			X	X
Ретрансляция	X	X	X	X
Зоны ретрансляции			X	X
Использование метрики RTT для ретрансляторов			X	
Виды				X
Round robin (система круговых перестановок)	X	X	X	X
Изменяемый порядок RRset			X	

Таблица B.1 (продолжение)

Свойство	Версия BIND 4.9.7	8.1.2	8.2.3	9.1.0
Изменяемый список сортировки	X		X	X
Выключение рекурсии	X	X	X	X
Списки доступа для рекурсии			X	X
IPv6				
AAAA-записи	X	X	X	X
A6-записи				X
DNAME-записи				X
Бит-строковые метки				X
Работа с цепочками DNAME и A6				X
Списки доступа для запросов	X[a]	X	X	X
Списки доступа для передачи зон	X[b]	X	X	X
DNSSEC				
Загрузка записей SIG, KEY и NXT			X	X
Проверка по «цепи доверия»				X
Динамические обновления защищенных зон				X

[a] С помощью TXT-записей secure_zone.

[b] С помощью xfrnets.

C

Сборка и установка BIND на Linux-системах

Версии пакета BIND, поставляемые с большинством версий Linux, являются достаточно свежими, для большинства недавно выпущенных дистрибутивов Linux – это, как правило, BIND версии 8.2.2. Однако наиболее современной версией пакета является 8.2.3, при этом консорциум ISC рекомендует обновление до BIND версии 9. Это приложение для тех читателей, которые не желают ждать выхода соответствующих обновлений для своих Linux-систем.

BIND 8.2.3

Сборка и установка BIND версии 8.2.3 – очень простое действие. Ниже приводятся подробные инструкции.

Загрузите исходные тексты

Во-первых, необходимо получить исходные тексты пакета. Их копия хранится на сервере *ftp.isc.org*, и доступна для анонимного FTP-копирования:

```
% cd /tmp
% ftp ftp.isc.org.
Connected to isrv4.pa.vix.com.
220 ProFTPD 1.2.0 Server (ISC FTP Server) [ftp.isc.org]
Name (ftp.isc.org.:user): ftp
331 Anonymous login ok, send your complete e-mail address as password.
Password:
230 Anonymous access granted, restrictions apply.
```

```
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Теперь следует найти нужный файл:

```
ftp > cd /isc/bind/src/cur/bind-8
250 CWD command successful.
ftp > binary
200 Type set to I.
ftp > get bind-src.tar.gz
local: bind-src.tar.gz remote: bind-src.tar.gz
200 PORT command successful.
150 Opening BINARY mode data connection for bind-src.tar.gz (1309147 bytes).
226 Transfer complete.
1309147 bytes received in 23 seconds (56 Kbytes/s)
ftp > quit
221 Goodbye.
```

Распакуйте архив исходных текстов

Теперь у нас есть упакованный *tar*-файл, содержащий исходные тексты пакета BIND. Следует просто использовать команду *tar* для распаковки и извлечения файлов:

```
% tar -zxvf bind-src.tar.gz
```

(Подразумевается, что доступна версия программы *tar*, которая умеет обрабатывать архивы, упакованные с помощью *gzip*; в противном случае копию такой реализации *tar* можно получить по FTP с сервера *ftp.gnu.org* (файл называется */gnu/tar/tar-1.13.tar*.)). В результате будет создан каталог *src*, содержащий несколько подкаталогов, в частности *bin*, *include*, *lib*, а также *port*. Эти подкаталоги имеют следующий состав:

bin

Исходные тексты всех исполняемых файлов пакета BIND, включая *named*.

include

Копии включаемых файлов, используемых кодом BIND. Их следует использовать для сборки сервера имен вместо поставляемых с используемой операционной системой, поскольку они были соответствующим образом обновлены.

lib

Исходные тексты библиотек, используемых пакетом BIND.

port

Информация, используемая пакетом BIND для настройки окружения сборки и выбора параметров компиляции для различных операционных систем.

Используйте правильные параметры компиляции

Чтобы производить сборку, необходим компилятор языка C. Практически все версии и дистрибутивы Linux включают *gcc*, компилятор языка C проекта GNU, который отлично подходит для нашей задачи. В случае необходимости получить *gcc*, следует обращаться по адресу *http://www.fsf.org/software/gcc/gcc.html*.

По умолчанию BIND считает, что используется компилятор GNU C, а также прочие GNU-инструменты, такие как *flex* и *byacc*. Они являются стандартными компонентами большинства сред разработки Linux-систем. Если в используемой версии Linux присутствует другой набор программ, следует изменить соответствующим образом файл *port/linux/Makefile.set*. BIND узнает, какими инструментами следует пользоваться, из этого файла.

Произведите сборку

Теперь следует произвести полную сборку пакета. Прежде всего, выполним команду:

```
% make stdlinks
```

Затем:

```
% make clean
% make depend
```

Эти команды удаляют все объектные файлы, которые могли остаться от предшествующих попыток сборки, и обновляет зависимости для файла сборки (*Makefile*). Теперь можно произвести компиляцию исходных текстов:

```
% make all
```

Компиляция должна пройти без ошибок. Теперь можно установить свежесобранные программы *named* и *named-xfer* в каталог */usr/sbin*. Чтобы выполнить эту операцию, необходимы полномочия суперпользователя (root). Воспользуемся командой

```
# make install
```

BIND 9.1.0

Ниже приводятся инструкции по сборке и установке BIND версии 9.1.0 на Linux-системе.

Загрузите исходные тексты

Как и в случае с BIND версии 8.2.3, следует, прежде всего, получить исходные тексты. Разумеется, обратившись по FTP к серверу *ftp.isc.org*:

```
% cd /tmp
% ftp ftp.isc.org.
Connected to isrv4.pa.vix.com.
220 ProFTPD 1.2.1 Server (ISC FTP Server) [ftp.isc.org]
Name (ftp.isc.org.:user): ftp
331 Anonymous login ok, send your complete email address as your password.
Password:
230 Anonymous access granted, restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Измените рабочий каталог и скопируйте файл с исходными текстами:

```
ftp> cd /isc/bind9/9.1.0/
250 CWD command successful.
ftp> get bind-9.1.0.tar.gz
local: bind-9.1.0.tar.gz remote: bind-9.1.0.tar.gz
200 PORT command successful.
150 Opening BINARY mode data connection for bind-9.1.0.tar.gz (3299471
bytes).
226 Transfer complete.
3299471 bytes received in 92.4 secs (35 Kbytes/sec)
ftp> quit
221 Goodbye.
```

Распакуйте архив исходных текстов

Для извлечения файлов используем команду *tar*:

```
% tar zxvf bind-9.1.0.tar.gz
```

В отличие от случая дистрибутива BIND версии 8.2.3 будет создан отдельный подкаталог в текущем рабочем каталоге, содержащий все исходные тексты BIND (*bind-9.1.0*). Дистрибутивы BIND 8 всегда содержали непосредственно подкаталоги дерева исходных текстов пакета. В каталоге *bind-9.1.0* присутствуют следующие подкаталоги:

bin

Исходные тексты всех исполняемых файлов пакета BIND, включая *named*.

contrib

Инструментарий от сторонних разработчиков.

doc

Документация к пакету BIND, включающая бесценное руководство по ресурсам для администратора (Administrator Resource Manual).

lib

Исходные тексты библиотек, используемых пакетом BIND.

make

Файлы сборки.

Выполните configure и произведите сборку

Еще одно отличие версии 9 в том, что в ней используется почти волшебный сценарий *configure*, самостоятельно определяющий необходимость включения тех или иных библиотек и использования тех или иных параметров компиляции. В файле README содержится информация по существующим дополнительным настройкам. *configure* предоставляет ключи командной строки, которые позволяют производить сборку без механизма threads, использовать произвольный каталог для установки, а также производить настройку прочих параметров. Выполним *configure*:

```
%./configure
```

Или, для случая, когда следует отключить threads:

```
%./configure disable-threads
```

Сборка BIND:

```
% make all
```

Компиляция исходных текстов должна пройти без ошибок. Чтобы установить BIND, необходимо дать следующую команду от пользователя root:

```
# make install
```

И это все!

D

Домены высшего уровня

В следующей таблице перечислены все двухбуквенные коды стран и все домены высшего уровня, не связанные со странами. На момент написания книги не все страны зарегистрированы в пространстве имен сети Интернет, но не столь многие отсутствуют.

Домен	Страна или организация	Домен	Страна или организация
AC	Остров Вознесения	AT	Австрия
AD	Андорра	AU	Австралия
AE	Объединенные Арабские Эмираты	AW	Аруба, остров (Нидерландские Антильские Острова)
AF	Афганистан	AZ	Азербайджан
AG	Антигуа и Барбуда	BA	Босния и Герцеговина
AI	Ангила, острова	BB	Барбадос
AL	Албания	BD	Бангладеш
AM	Армения	BE	Бельгия
AN	Нидерландские Антильские Острова	BF	Буркина-Фасо
AO	Ангола	BG	Болгария
AQ	Антарктика	BH	Бахрейн
AR	Аргентина	BI	Бурунди
ARPA	ARPA Internet	BJ	Бенин
AS	Американское Самоа	BM	Бермудские Острова

Домен	Страна или организация	Домен	Страна или организация
BN	Бруней Дар-эс-Салам	CY	Кипр
BO	Боливия	CZ	Республика Чехия
BR	Бразилия	DE	Германия
BS	Багамские Острова	DJ	Джибути
BT	Бутан	DK	Дания
BV	Буве, остров	DM	Доминика
BW	Ботсвана	DO	Доминиканская Республика
BY	Белоруссия	DZ	Алжир
BZ	Белиз	EC	Эквадор
CA	Канада	EDU	Образование
CC	Кокосовые острова (острова Килинг)	EE	Эстония
CD	Демократическая Республика Конго	EG	Египет
CF	Центрально- Африканская Республика	EH	Западная Сахара
CG	Конго	ER	Эритрея
CH	Швейцария	ES	Испания
CI	Кот-д'Ивуар	ET	Эфиопия
CK	Острова Кука	FI	Финляндия
CL	Чили	FJ	Фиджи
CM	Камерун	FK	Фолклендские (Мальвинские) Острова
CN	Китай	FM	Микронезия
CO	Колумбия	FO	Фарерские острова
COM	Родовой (изначально – Коммерческий)	FR	Франция
CR	Коста-Рика	FX	Территория Франции
CU	Куба	GA	Габон
CV	Кабо-Верде	GB	Соединенное Королевство Великобритания[a]
CX	Остров Рождества	GD	Гренада

[a] На практике Соединенное Королевство использует суффикс «UK» в качестве домена высшего уровня.

Домен	Страна или организация	Домен	Страна или организация
GE	Грузия	IN	Индия
GF	Французская Гвиана	INT	Международные объекты
GG	Гернси, Олдерни и Сарк (острова в проливе Ла-Манш)	IO	Британская территория в Индийском океане
GH	Гана	IQ	Ирак
GI	Гибралтар	IR	Иран
GL	Гренландия	IS	Исландия
GM	Гамбия	IT	Италия
GN	Гвинея	JE	Джерси, остров
GOV	Федеральное правительство США	JM	Ямайка
GP	Гваделупа	JO	Иордания
GQ	Экваториальная Гвинея	JP	Япония
GR	Греция	KE	Кения
GS	Южная Георгия и Южные Сандвичевы острова	KG	Кыргызстан
GT	Гватемала	KH	Камбоджа
GU	Гуам	KI	Кирибати
GW	Гвинея-Биссау	KM	Коморские Острова
GY	Гайана	KN	Сент-Китс и Невис
HK	Гонконг	KP	Корейская Народно-Демократическая Республика
HM	Херд и Мак-Дональд, острова	KR	Корейская Республика
HN	Гондурас	KW	Кувейт
HR	Хорватия	KY	Каймановы Острова
HT	Гаити	KZ	Казахстан
HU	Венгрия	LA	Лаосская Народно-Демократическая Республика
ID	Индонезия	LB	Ливан
IE	Ирландия	LC	Сент-Люсия
IL	Израиль	LI	Лихтенштейн
IM	Мэн, остров	LK	Шри-Ланка

Домен	Страна или организация	Домен	Страна или организация
LR	Либерия	MZ	Мозамбик
LS	Лесото	NA	Намибия
LT	Литва	NATO	НАТО, Североатлантический союз
LU	Люксембург	NC	Новая Каледония
LV	Латвия	NE	Нигер
LY	Ливийская Арабская Джамахирия	NET	Родовой (ранее – сетевые организации)
MA	Марокко	NF	Норфолк, остров
MC	Монако	NG	Нигерия
MD	Республика Молдова	NI	Никарагуа
MG	Мадагаскар	NL	Нидерланды
MH	Маршалловы Острова	NO	Норвегия
MIL	Армия США	NP	Непал
MK	Македония, бывшая республика Югославии	NR	Науру
ML	Мали	NU	Ниуэ, остров
MM	Мьянма	NZ	Новая Зеландия
MN	Монголия	OM	Оман
MO	Макао	ORG	Родовой (ранее – организации)
MP	Северные Марианские Острова	PA	Панама
MQ	Мартиника	PE	Перу
MR	Мавритания	PF	Французская Полинезия
MS	Монтсеррат, остров	PG	Папуа – Новая Гвинея
MT	Мальта	PH	Филиппины
MU	Маврикий	PK	Пакистан
MV	Мальдивы	PL	Польша
MW	Малави	PM	Сен-Пьер и Микелон
MX	Мексика	PN	Питкэрн, остров
MY	Малайзия	PR	Пуэрто-Рико

Домен	Страна или организация	Домен	Страна или организация
PS	Палестинская автономия	SY	Сирийская Арабская Республика
PT	Португалия	SZ	Свазиленд
PW	Палау	TC	Теркс и Кайкос
PY	Парагвай	TD	Чад
QA	Катар	TF	Французские южные территории
RE	Реюньон	TG	Того
RO	Румыния	TH	Таиланд
RU	Российская Федерация	TJ	Таджикистан
RW	Руанда	TK	Токелау
SA	Саудовская Аравия	TM	Туркменистан
SB	Соломоновы Острова	TN	Тунис
SC	Сейшельские Острова	TO	Тонга
SD	Судан	TP	Восточный Тимор
SE	Швеция	TR	Турция
SG	Сингапур	TT	Тринидад и Тобаго
SH	Остров Св. Елены	TV	Тувалу
SI	Словения	TW	Тайвань, провинция Китая
SJ	Свальбард и Ян-Майен, острова	TZ	Объединенная Республика Танзания
SK	Словакия	UA	Украина
SL	Сьерра-Леоне	UG	Уганда
SM	Сан-Марино	UK	Соединенное Королевство
SN	Сенегал	UM	Малые удаленные острова (США)
SO	Сомали	US	США
SR	Суринам	UY	Уругвай
ST	Сан-Томе и Принсипи	UZ	Узбекистан
SU	Союз Советских Социалистических Республик	VA	Святейший престол (город-государство Ватикан)
SV	Сальвадор	VC	Сент-Винсент и Гренадины

Домен	Страна или организация	Домен	Страна или организация
VE	Венесуэла	YE	Йемен
VG	Британские Виргинские Острова (Брит.)	YT	Майотта, остров
VI	Виргинские Острова (США)	YU	Югославия
VN	Вьетнам	ZA	Южная Африка
VU	Вануату	ZM	Замбия
WF	Острова Уоллис и Футуна	ZR	Республика Заир
WS	Самоа	ZW	Зимбабве

E

Настройка DNS-сервера и клиента BIND

Инструкции и операторы файла загрузки и файла настройки BIND

Ниже приводится удобный перечень инструкций файла загрузки и операторов файла настройки DNS-сервера BIND, а также инструкции настройки DNS-клиента BIND. Некоторые из инструкций и операторов существуют только в более поздних версиях пакета, и, как следствие, поддерживаются не всеми существующими установками. Более новые инструкции и операторы отмечены конкретными версиями BIND, в которых они появились (к примеру, 8.2+). Инструкции и операторы, которые существуют уже давно, никак не выделены.

Инструкции файла загрузки BIND 4

directory

Функциональность:

Указание рабочего каталога DNS-сервера

Синтаксис:

```
directory new-directory
```

Пример:

```
directory /var/named
```

См. также:

Оператор *options* для версий 8.x.x и 9.x.x, предписание *directory*

Рассматривается в главе 4 «Установка BIND».

primary

Функциональность:

Настройка DNS-сервера в качестве первичного мастера для зоны

Синтаксис:

```
primary domain-name-of-zone file
```

Пример:

```
primary  movie.edu   db.movie.edu
```

См. также:

Оператор *zone* для версий 8.x.x и 9.x.x, тип *master*

Рассматривается в главе 4 «Установка BIND».

secondary

Функциональность:

Настройка DNS-сервера в качестве вторичного для зоны

Синтаксис:

```
secondary domain-name-of-zone ip-address-list [backup-file]
```

Пример:

```
secondary  movie.edu   192.249.249.3 bak.movie.edu
```

См. также:

Оператор *zone* для версий 8.x.x и 9.x.x, тип *slave*

Рассматривается в главе 4 «Установка BIND».

cache

Функциональность:

Указание имени файла, из которого следует загружать указатели (имена и адреса) корневых DNS-серверов

Синтаксис:

```
cache    .    file
```

Пример:

```
cache    .    db.cache
```

См. также:

Оператор *zone* для версий 8.x.x и 9.x.x, тип *hint*

Рассматривается в главе 4 «Установка BIND».

forwarders

Функциональность:

Указание сервера или DNS-серверов, которым следует посылать неразрешенные запросы

Синтаксис:

```
forwarders ip-address-list
```

Пример:

```
forwarders 192.249.249.1 192.249.249.3
```

См. также:

Оператор *options* для версий 8.x.x и 9.x.x, предписание *forwarders*

Рассматривается в главе 10 «Дополнительные возможности».

sortlist

Функциональность:

Указание предпочтительной сети

Синтаксис:

```
sortlist network-list
```

Пример:

```
sortlist 10.0.0.0
```

См. также:

Оператор *options* для версий 8.2+ и 9.1.0+, предписание *sortlist*

Рассматривается в главе 10 «Дополнительные возможности».

slave

Инструкция идентична options *forward-only* для версий 4.9.x и предписанию *forward* оператора *options* для версий 8.x.x и 9.x.x.

include (4.9+)

Функциональность:

Включение содержимого другого файла в *named.boot*

Синтаксис:

```
include file
```

Пример:

```
include  bootfile.primary
```

См. также:

Оператор *include* для версий 8.x.x и 9.x.x

Рассматривается в главе 7 «Работа с BIND».

stub (4.9+)

Функциональность:

Указание дочерней зоны, информацию о делегировании которой DNS-сервер должен периодически получать

Синтаксис:

```
stub domain-name-of-zone ip-address-list [backup-file]
```

Пример:

```
stub movie.edu 192.249.249.3 stub.movie.edu
```

См. также:

Оператор *zone* для версий 8.x.x и 9.x.x, тип *stub*

Рассматривается в главе 9 «Материнство».

options (4.9+)

options forward-only

Функциональность:

Предписание DNS-серверу не производить разрешение доменных имен в обход ретранслятора

См. также:

Оператор *option* для версий 8.x.x и 9.x.x, предписание *forward*

Рассматривается в:

Главе 10 «Дополнительные возможности».

options no-recursion

Функциональность:

Предписание DNS-серверу не производить рекурсивное разрешение доменных имен

См. также:

Оператор *options* для версий 8.x.x и 9.x.x, предписание *recursion*

Рассматривается в главе 10 «Дополнительные возможности» и в главе 11 «Безопасность».

options no-fetch-glue

Функциональность:

Предотвращает поиск DNS-сервером отсутствующих связующих записей при создании ответа

См. также:

Оператор *options* для версий 8.x.x, предписание *fetch-glue*

Рассматривается в главе 10 «Дополнительные возможности» и в главе 11 «Безопасность».

options query-log

Функциональность:

Занесение в log-файл всех запросов, полученных DNS-сервером

См. также:

Оператор *logging* для версий 8.x.x и 9.1.0+, категория *queries*

Рассматривается в главе 7 «Работа с BIND», а также в главе 14 «Разрешение проблем DNS и BIND».

options fake-iquery

Функциональность:

Предписание DNS-серверу реагировать на старомодные инверсные запросы выдуманным ответом, а не ошибкой

См. также:

Оператор *options* для версий 8.x.x, предписание *fake-iquery*

Рассматривается в главе 12 «nslookup и dig».

limit (4.9+)

limit transfers-in

Функциональность:

Ограничение общего числа попыток передачи зоны, совершаемых DNS-сервером в один прием

См. также:

Оператор *options* для версий 8.x.x и 9.x.x, предписание *transfers-in*

limit transfers-per-ns

Функциональность:

Ограничение числа зон параллельно получаемых от любого другого DNS-сервера

См. также:

Оператор *options* для версий 8.x.x и 9.x.x, предписание *transfers-per-ns*

limit datasize

Функциональность:

Увеличение размера сегмента данных, используемого *named* (работает лишь в некоторых операционных системах)

См. также:

Оператор *options* для версий 8.x.x и 9.1.0+, предписание *datasize*

Также рассматривается в главе 10 «Дополнительные возможности».

xfrnets (4.9+)

Функциональность:

Ограничение получателей зоны списком IP-адресов получателей или сетей

Синтаксис:

```
xfrnets ip-address-or-network-list
```

Пример:

```
xfrnets 15.0.0.0 128.32.0.0
```

См. также:

Операторы *options* и *zone* для версий 8.x.x и 9.x.x, предписание *allow-transfer*

Рассматривается в главе 11 «Безопасность».

bogusns (4.9+)

Функциональность:

Предписывает DNS-серверу не посылать запросы серверам из списка, поскольку известно, что они возвращают некорректные ответы

Синтаксис:

```
bogusns ip-address-list
```

Пример:

```
bogusns 15.255.152.4
```

См. также:

Оператор *server* для версий 8.x.x и 9.1.0+, предписание *bogus*

Рассматривается в главе 10 «Дополнительные возможности».

check-names (4.9.4+)

Функциональность:

Настройка механизма проверки имен

Синтаксис:

```
check-names primary|secondary|response-fail|warn|ignore
```

Пример:

```
check-names primary ignore
```

См. также:

Операторы *options* и *zone* для версий 8.x.x, предписание *check-names*

Рассматривается в главе 4 «Установка BIND».

Операторы файла настройки BIND 8

acl

Функциональность:

Создание именованного списка адресов

Синтаксис:

```
acl name {
    address_match_list;
};
```

Рассматривается в главе 10 «Дополнительные возможности» и в главе 11 «Безопасность».

controls (8.2+)

Функциональность:

Настройка канала, используемого *ndc* для управления DNS-сервером

Синтаксис:

```
controls {
   [ inet ( ip_addr | * ) port ip_port allow
      address_match_list; ]
   [ unix path_name perm number owner number group number; ]
};
```

Рассматривается в главе 7 «Работа с BIND».

include

Функциональность:

Включение указанного файла в точке, где встречен оператор *include*

Синтаксис:

```
include path_name;
```

Рассматривается в главе 7 «Работа с BIND».

key (8.2+)

Функциональность:

Создание идентификатора ключа, который может использоваться в операторе *server*, либо в списке адресов для связывания TSIG-ключа с определенным DNS-сервером

Синтаксис:

```
key key_id {
   algorithm algorithm_id;
   secret secret_string;
};
```

Рассматривается в главе 10 «Дополнительные возможности» и в главе 11 «Безопасность».

logging

Функциональность:

Настройка параметров, связанных с ведением log-файла

Синтаксис:

```
logging {
   [ channel channel_name {
     ( file path_name
       [ versions ( number | unlimited ) ]
        [ size size_spec ]
      | syslog ( kern | user | mail | daemon | auth | syslog
               | lpr | news | uucp | cron | authpriv | ftp |
                 local0 | local1 | local2 | local3 |
                 local4 | local5 | local6 | local7 )
      | null );

     [ severity ( critical | error | warning | notice |
                  info  | debug [ level ] | dynamic ); ]
     [ print-category yes_or_no; ]
     [ print-severity yes_or_no; ]
     [ print-time yes_or_no; ]
   }; ]

   [ category category_name {
     channel_name; [ channel_name; ... ]
   }; ]
   ...
};
```

Рассматривается в главе 7 «Работа с BIND».

options

Функциональность:

Изменение общих настроек

Синтаксис:

```
options {
   [ allow-query { address_match_list }; ]
   [ allow-recursion { address_match_list }; ]
   [ allow-transfer { address_match_list }; ]
   [ also-notify { ip_addr; [ ip_addr; ... ] }; ]
   [ auth-nxdomain yes_or_no; ]
   [ blackhole { address_match_list }; ]
   [ check-names ( master | slave | response ) ( warn | fail  ignore ); ]
   [ cleaning-interval number; ]
   [ coresize size_spec; ]
```

```
    [ datasize size_spec; ]
    [ deallocate-on-exit yes_or_no; ]
    [ dialup yes_or_no; ]
    [ directory path_name; ]
    [ dump-file path_name; ]
    [ fake-iquery yes_or_no; ]
    [ fetch-glue yes_or_no; ]
    [ files size_spec; ]
    [ forward ( only | first ); ]
    [ forwarders { [ ip_addr ; [ ip_addr ; ... ] ] }; ]
    [ has-old-clients yes_or_no; ]
    [ heartbeat-interval number; ]
    [ host-statistics yes_or_no; ]
    [ interface-interval number; ]
    [ lame-ttl number; ]
    [ listen-on [ port ip_port ] { address_match_list }; ]
    [ maintain-ixfr-base yes_or_no; ]
    [ max-ixfr-log-size number; ]
    [ max-ncache-ttl number; ]
    [ max-transfer-time-in number; ]
    [ memstatistics-file path_name; ]
    [ min-roots number; ]
    [ multiple-cnames yes_or_no; ]
    [ named-xfer path_name; ]
    [ notify yes_or_no; ]
    [ pid-file path_name; ]
    [ query-source [ address ( ip_addr | * ) ] [ port ( ip_port | * ) ]; ]
    [ recursion yes_or_no; ]
    [ rfc2308-type1 yes_or_no; ]
    [ rrset-order { order_spec; [ order_spec; ... ] }; ]
    [ serial-queries number; ]
    [ sortlist { address_match_list }; ]
    [ stacksize size_spec; ]
    [ statistics-file path_name; ]
    [ statistics-interval number; ]
    [ topology { address_match_list }; ]
    [ transfer-format ( one-answer | many-answers ); ]
    [ transfer-source ( ip_addr | * ); ]
    [ transfers-in  number; ]
    [ transfers-per-ns number; ]
    [ treat-cr-as-space yes_or_no; ]
    [ use-id-pool yes_or_no; ]
    [ use-ixfr yes_or_no; ]
    [ version version_string; ]
};
```

Рассматривается в главе 4 «Установка BIND», в главе 10 «Дополнительные возможности», в главе 11 «Безопасность» и в главе 16 «Обо всем понемногу».

server

Функциональность:

Определение характеристик, связанных с удаленным DNS-сервером

Синтаксис:

```
server ip_addr {
   [ bogus yes_or_no; ]
   [ keys { key_id [ key_id ... ] }; ]
   [ support-ixfr yes_or_no; ]
   [ transfer-format ( one-answer | many-answers ); ]
};
```

Рассматривается в главе 10 «Дополнительные возможности» и в главе 11 «Безопасность».

trusted-keys (8.2+)

Функциональность:

Настройка открытых ключей корневых зон сегментов DNSSEC

Синтаксис:

```
trusted-keys {
domain-name flags protocol_id algorithm_id public_key_string;
[ domain-name flags protocol_id algorithm_id public_key_string; [ ... ] ]
};
```

Рассматривается в главе 11 «Безопасность».

zone

Функциональность:

Настройка параметров зоны, обслуживаемой DNS-сервером

Синтаксис:

```
zone "domain_name" [ ( in | hs | hesiod | chaos ) ] {
   type master;
   file path_name;
   [ allow-query { address_match_list }; ]
   [ allow-transfer { address_match_list }; ]
   [ allow-update { address_match_list }; ]
   [ also-notify { ip_addr; [ ip_addr; ... ]
   [ check-names ( warn | fail | ignore ); ]
   [ dialup yes_or_no | notify; ]
   [ forward ( only | first ); ]
   [ forwarders { [ ip_addr; [ ip_addr; ... ] ] }; ]
   [ ixfr-base path_name; ]
   [ ixfr-tmp-file path_name; ]
   [ maintain-ixfr-base yes_or_no; ]
   [ notify yes_or_no; ]
```

```
   [ pubkey flags protocol_id algorithm_id public_key_string; ]
};

zone "domain_name" [ ( in | hs | hesiod | chaos ) ] {
   type slave;
   masters [ port ip_port ] { ip_addr; [ ip_addr; ... ] };
   [ allow-query { address_match_list }; ]
   [ allow-transfer { address_match_list }; ]
   [ allow-update { address_match_list }; ]
   [ also-notify { ip_addr; [ ip_addr; ... ] };
   [ check-names ( warn | fail | ignore ); ]
   [ dialup yes_or_no; ]
   [ file path_name; ]
   [ forward ( only | first ); ]
   [ forwarders { [ ip_addr; [ ip_addr; ... ] ] }; ]
   [ ixfr-base path_name; ]
   [ max-transfer-time-in number; ]
   [ notify yes_or_no; ]
   [ pubkey flags protocol_id algorithm_id public_key_string; ]
   [ transfer-source ip_addr; ]
};

zone "domain_name" [ ( in | hs | hesiod | chaos ) ] {
   type stub;
   masters [ port ip_port ] { ip_addr; [ ip_addr; ... ] };
   [ allow-query { address_match_list }; ]
   [ allow-transfer { address_match_list }; ]
   [ allow-update { address_match_list }; ]
   [ check-names ( warn | fail | ignore ); ]
   [ dialup yes_or_no; ]
   [ file path_name; ]
   [ forward ( only | first ); ]
   [ forwarders { [ ip_addr ; [ ip_addr ; ... ] ] }; ]
   [ max-transfer-time-in number; ]
   [ pubkey flags protocol_id algorithm_id  public_key_string; ]
   [ transfer-source ip_addr; ]
};

zone "domain_name" [ ( in | hs | hesiod | chaos ) ] {
   type forward;
   [ forward ( only | first ); ]
   [ forwarders { [ ip_addr ; [ ip_addr ; ... ] ] }; ]
};

zone "." [ ( in | hs | hesiod | chaos ) ] {
   type hint;
   file path_name;
   [ check-names ( warn | fail | ignore ); ]
};
```

Рассматривается в главе 4 «Установка BIND» и в главе 10 «Дополнительные возможности».

Операторы файла настройки BIND 9

acl

Функциональность:

Создание именованного списка адресов

Синтаксис:

```
acl name {
    address_match_list;
};
```

Рассматривается в главе 10 «Дополнительные возможности» и в главе 11 «Безопасность».

controls

Функциональность:

Настройка канала, используемого *rndc* для управления DNS-сервером

Синтаксис:

```
controls {
    [ inet ( ip_addr | * ) port ip_port allow address_match_list keys
key_list; ]
    [ inet ... ; ]
};
```

Рассматривается в главе 7 «Работа с BIND».

include

Функциональность:

Включение указанного файла в точке, где встречен оператор *include*

Синтаксис:

```
include path_name;
```

Рассматривается в главе 7 «Работа с BIND».

key

Функциональность:

Создание идентификатора ключа, который может использоваться в операторе *server*, либо в списке адресов для связывания TSIG-ключа с определенным DNS-сервером

Синтаксис:

```
key key_id {
   algorithm algorithm_id;
   secret secret_string;
};
```

Рассматривается в главе 10 «Дополнительные возможности» и в главе 11 «Безопасность».

logging

Функциональность:

Настройка параметров, связанных с ведением log-файла

Синтаксис:

```
logging {
  [ channel channel_name {
    ( file path_name
    [ versions ( number | unlimited ) ]
        [ size size_spec ]
      | syslog ( kern | user | mail | daemon | auth | syslog  | lpr |
                 news | uucp | cron | authpriv | ftp |
                 local0 | local1 | local2 | local3 |
                 local4 | local5 | local6 | local7 )
      | stderr
      | null );

    [ severity ( critical | error | warning | notice |
                 info  | debug [ level ] | dynamic ); ]
    [ print-category yes_or_no; ]
    [ print-severity yes_or_no; ]
    [ print-time yes_or_no; ]
  }; ]

   [ category category_name {
    channel_name; [ channel_name; ... ]
  }; ]
   ...
};
```

Рассматривается в главе 7 «Работа с BIND».

options

Функциональность:

Изменение общих настроек

Синтаксис:

```
options {
   [ additional-from-auth yes_or_no; ]
   [ additional-from-cache yes_or_no; ]
   [ allow-notify { address_match_list }; ]
   [ allow-query { address_match_list }; ]
   [ allow-recursion { address_match_list }; ]
   [ allow-transfer { address_match_list }; ]
   [ also-notify { ip_addr [ port ip_port ] ; [ ip_addr [ port ip_port ] ; ... ] }; ]
   [ auth-nxdomain yes_or_no; ]
   [ blackhole { address_match_list }; ]
   [ cleaning-interval number; ]
```

```
    [ coresize size_spec; ]
    [ datasize size_spec; ]
    [ dialup yes_or_no; ]
    [ directory path_name; ]
    [ dump-file path_name; ]
    [ files size_spec; ]
    [ forward ( only | first ); ]
    [ forwarders { [ ip_addr ; [ ip_addr ; ... ] ] }; ]
    [ heartbeat-interval number; ]
    [ interface-interval number; ]
    [ lame-ttl number; ]
    [ listen-on [ port ip_port ] { address_match_list }; ]
    [ listen-on-v6 [ port ip_port ] { address_match_list }; ]
    [ max-cache-ttl number; ]
    [ max-ncache-ttl number; ]
    [ max-refresh-time number; ]
    [ max-retry-time number; ]
    [ max-transfer-idle-in number; ]
    [ max-transfer-idle-out number; ]
    [ max-transfer-time-in number; ]
    [ max-transfer-time-out number; ]
    [ min-refresh-time number; ]
    [ min-retry-time number; ]
    [ notify yes_or_no | explicit; ]
    [ notify-source ( ip_addr | * ) [ port ip_port ]; ]
    [ notify-source-v6 ( ip6_addr | * ) [ port ip_port ]; ]
    [ pid-file path_name; ]
    [ port ip_port; ]
    [ query-source [ address ( ip_addr | * ) ] [ port ( ip_port | * ) ]; ]
    [ query-source-v6 [ address ( ip6_addr | * ) ] [ port ( ip_port | * ) ]; ]
    [ recursion yes_or_no; ]
    [ recursive-clients number; ]
    [ sig-validity-interval number; ]
    [ sortlist { address_match_list }; ]
    [ stacksize size_spec; ]
    [ statistics-file path_name; ]
    [ tcp-clients number; ]
    [ tkey-dhkey key_name key_tag; ]
    [ tkey-domain domain_name; ]
    [ transfer-format ( one-answer | many-answers ); ]
    [ transfer-source ( ip_addr | * ) [ port ip_port ]; ]
    [ transfer-source-v6 ( ip6_addr | * ) [ port ip_port ]; ]
    [ transfers-in  number; ]
    [ transfers-out number; ]
    [ transfers-per-ns number; ]
    [ version version_string; ]
    [ zone-statistics yes_or_no; ]
};
```

Рассматривается в главе 4 «Установка BIND», в главе 10 «Дополнительные возможности», в главе 11 «Безопасность» и в главе 16 «Обо всем понемногу».

server

Функциональность:

Определение характеристик, связанных с удаленным DNS-сервером

Синтаксис:

```
server ip_addr {
   [ bogus yes_or_no; ]
   [ keys { key_id [ key_id ... ] }; ]
   [ provide-ixfr yes_or_no; ]
   [ request-ixfr yes_or_no; ]
   [ transfers number; ]
   [ transfer-format ( one-answer | many-answers ); ]
};
```

Рассматривается в главе 10 «Дополнительные возможности» и в главе 11 «Безопасность».

trusted-keys

Функциональность:

Настройка открытых ключей корневых зон сегментов DNSSEC

Синтаксис:

```
trusted-keys {
   domain-name flags protocol_id algorithm_id public_key_string;
   [ domain-name flags protocol_id algorithm_id public_key_string; [ ... ] ]
};
```

Рассматривается в главе 11 «Безопасность».

view

Функциональность:

Создание и настройка вида

Синтаксис:

```
view "view_name" [ ( in | hs | hesiod | chaos ) ] {
   match-clients { address_match_list };
   [ allow-notify { address_match_list }; ]
   [ allow-query { address_match_list }; ]
   [ allow-recursion { address_match_list }; ]
   [ allow-transfer { address_match_list }; ]
   [ also-notify { ip_addr; [ ip_addr; ... ] }; ]
   [ auth-nxdomain yes_or_no; ]
   [ cleaning-interval number; ]
   [ forward ( only | first ); ]
   [ forwarders { [ ip_addr; [ ip_addr; ... ] ] }; ]
   [ key ... ]
   [ lame-ttl number; ]
   [ min-refresh-time number; ]
   [ min-retry-time number; ]
```

```
    [ max-cache-ttl number; ]
    [ max-ncache-ttl number; ]
    [ max-transfer-idle-out number; ]
    [ max-transfer-time-out number; ]
    [ max-refresh-time number; ]
    [ max-retry-time number; ]
    [ notify yes_or_no | explicit; ]
    [ provide-ixfr yes_or_no; ]
    [ query-source [ address ( ip_addr | * ) ] [ port ( ip_port | * ) ]; ]
    [ query-source-v6 [ address ( ip6_addr | * ) ] [ port ( ip_port | * ) ]; ]
    [ recursion yes_or_no; ]
    [ request-ixfr yes_or_no; ]
    [ server ... ]
    [ sig-validity-interval number; ]
    [ sortlist { address_match_list }; ]
    [ transfer-format ( one-answer | many-answers ); ]
    [ transfer-source ( ip_addr | * ) [ port ip_port ]; ]
    [ transfer-source-v6 ( ip6_addr | * ) [ port ip_port ]; ]
    [ trusted-keys ... ]
    [ zone ... ]
};
```

Рассматривается в главе 10 «Дополнительные возможности» и в главе 11 «Безопасность».

zone

Функциональность:

Настройка параметров зоны, обслуживаемой DNS-сервером

Синтаксис:

```
zone "domain_name" [ ( in | hs | hesiod | chaos ) ] {
    type master;
    file path_name;
    [ allow-notify { address_match_list }; ]
    [ allow-query { address_match_list }; ]
    [ allow-transfer { address_match_list }; ]
    [ allow-update { address_match_list }; ]
    [ allow-update-forwarding { address_match_list }; ]
    [ also-notify { ip_addr [ port ip_port ]; [ ip_addr [ port ip_port ]; ... ]
    [ database string; [ string; ... ] ]
    [ dialup yes_or_no | notify; ]
    [ forward ( only | first ); ]
    [ forwarders { [ ip_addr; [ ip_addr; ... ] ] }; ]
    [ max-refresh-time number; ]
    [ max-retry-time number; ]
    [ max-transfer-idle-out number; ]
    [ max-transfer-time-out number; ]
    [ min-refresh-time number; ]
    [ min-retry-time number; ]
    [ notify yes_or_no | explicit; ]
    [ sig-validity-interval number; ]
```

```
   [ update-policy { update_policy_rule; [ ... ] }; ]
};

zone "domain_name" [ ( in | hs | hesiod | chaos ) ] {
   type slave;
   masters [ port ip_port ] { ip_addr [ port ip_port ] [ key key_id ];
[ ip_addr [ port ip_port ] [ key key_id ]; ... ] };
   [ allow-query { address_match_list }; ]
   [ allow-transfer { address_match_list }; ]
   [ allow-update { address_match_list }; ]
   [ allow-update-forwarding { address_match_list }; ]
   [ also-notify { ip_addr [ port ip_port ]; [ ip_addr [ port ip_port ]; ... ]
};
   [ dialup yes_or_no | notify | notify-passive | refresh | passive; ]
   [ file path_name; ]
   [ forward ( only | first ); ]
   [ forwarders { [ ip_addr; [ ip_addr; ... ] ] }; ]
   [ max-refresh-time number ; ]
   [ max-retry-time number ; ]
   [ max-transfer-idle-in number; ]
   [ max-transfer-idle-out number; ]
   [ max-transfer-time-in number; ]
   [ max-transfer-time-out number; ]
   [ min-refresh-time number ; ]
   [ min-retry-time number ; ]
   [ notify yes_or_no | explicit; ]
   [ transfer-source ( ip_addr | * ) [ port ip_port ]; ]
   [ transfer-source-v6 ( ip6_addr | * ) [ port ip_port ]; ]
};

 zone "domain_name" [ ( in | hs | hesiod | chaos ) ] {
   type stub;
   masters [ port ip_port ] { ip_addr [ [port ip_port ] [ key key_id ]; [ ip_addr
[ port ip_port ] [ key key_id ]; ... ] };
   [ allow-query { address_match_list }; ]
   [ allow-transfer { address_match_list }; ]
   [ allow-update { address_match_list }; ]
   [ allow-update-forwarding { address_match_list }; ]
   [ dialup yes_or_no | passive | refresh; ]
   [ file path_name; ]
   [ forward ( only | first ); ]
   [ forwarders { [ ip_addr ; [ ip_addr ; ... ] ] }; ]
   [ max-refresh-time number ; ]
   [ max-retry-time number ; ]
   [ max-transfer-idle-in number; ]
   [ max-transfer-idle-out number; ]
   [ max-transfer-time-in number; ]
   [ max-transfer-time-out number; ]
   [ min-refresh-time number ; ]
   [ min-retry-time number ; ]
   [ transfer-source ( ip_addr | * ) [ port ip_port ]; ]
   [ transfer-source-v6 ( ip6_addr | * ) [ port ip_port ]; ]
};
```

```
zone "domain_name" [ ( in | hs | hesiod | chaos ) ] {
   type forward;
   [ forward ( only | first ); ]
   [ forwarders { [ ip_addr ; [ ip_addr ; ... ] ] }; ]
};

zone "." [ ( in | hs | hesiod | chaos ) ] {
   type hint;
   file path_name;
};
```

Рассматривается в главе 4 «Установка BIND» и в главе 10 «Дополнительные возможности».

Операторы DNS-клиента BIND

Следующие операторы могут присутствовать в файле настройки клиента, */etc/resolv.conf*.

domain

Функциональность:

Задание локального доменного имени клиента

Синтаксис:

```
domain domain-name
```

Пример:

```
domain corp.hp.com
```

Рассматривается в главе 6 «Конфигурирование узлов».

search

Функциональность:

Задание локального доменного имени и списка поиска клиента

Синтаксис:

```
search local-domain-name next-domain-name-in-search-list
 ... last-domain-name-in-search-list
```

Пример:

```
search corp.hp.com pa.itc.hp.com hp.com
```

Рассматривается в главе 6 «Конфигурирование узлов».

nameserver

Функциональность:

Предписывает клиенту работать с определенным DNS-сервером

Синтаксис:

```
nameserver IP-address
```

Пример:

```
nameserver 15.255.152.4
```

Рассматривается в главе 6 «Конфигурирование узлов».

; и # (4.9+)

Функциональность:

Добавление комментария к файлу настройки клиента

Синтаксис:

```
; комментарий в свободной форме
```

или

```
# комментарий в свободной форме
```

Пример:

```
# Для совместимости с версией 4.8.3 к списку поиска добавлено
# доменное имя родительской зоны
```

Рассматривается в главе 6 «Конфигурирование узлов».

sortlist (4.9+)

Функциональность:

Перечисление предпочтительных сетей

Синтаксис:

```
sortlist network-list
```

Пример:

```
sortlist 128.32.4.0/255.255.255.0 15.0.0.0
```

Рассматривается в главе 6 «Конфигурирование узлов».

options ndots (4.9+)

Функциональность:

Указание числа точек, которое должно содержаться в аргументе, чтобы клиент произвел поиск по буквальному имени, прежде чем начать использовать список поиска

Синтаксис:

```
options ndots:number-of-dots
```

Пример:

```
options ndots:1
```

Рассматривается в главе 6 «Конфигурирование узлов».

options debug (4.9+)

Функциональность:

Включение отладочной диагностики в клиенте

Синтаксис:

```
options debug
```

Пример:

```
options debug
```

Рассматривается в главе 6 «Конфигурирование узлов».

options no-check-names (8.2+)

Функциональность:

Выключение проверки имен в клиенте

Синтаксис:

```
options no-check-names
```

Пример:

```
options no-check-names
```

Рассматривается в главе 6 «Конфигурирование узлов».

options attempts (8.2+)

Функциональность:

Число запросов, посылаемых клиентом каждому DNS-серверу

Синтаксис:

options attempts:*number-of-attempts*

Пример:

```
options attempts:2
```

Рассматривается в главе 6 «Конфигурирование узлов».

options timeout (8.2+)

Функциональность:

Интервал ожидания для каждого DNS-сервера

Синтаксис:

options timeout:*timeout-in-seconds*

Пример:

```
options timeout:1
```

Рассматривается в главе 6 «Конфигурирование узлов».

options rotate (8.2+)

Функциональность:

Изменение порядка, в котором клиент опрашивает DNS-серверы

Синтаксис:

```
options rotate
```

Пример:

```
options rotate
```

Рассматривается в главе 6 «Конфигурирование узлов».

Алфавитный указатель

Символы

- (дефис), 106
* (звездочка), 582
# (знак решетки), 149
    начало комментариев, 99
/* */, комментарии, 99
//, комментарии, 99
() (круглые скобки), 89, 611
/ (слэш), 24, 33
. (точка), 34
    ndots, настройка, 661
    в конце имени, 88
        необходимость наличия, 102
    завершающая имя
        абсолютное доменное, 610
    и локальные доменные имена, 137
    метка корневого узла, 24
    последняя, 138
    установка значения ndots, 147
; (точка с запятой)
    комментарии в файле resolv.conf, 149
    начало комментария, 87, 98, 611, 661
@, запись имен, 102, 611
_ (подчеркивание), 106

A

A-записи, 89, 612
    использование вместо CNAME-записей, 91
    порядок следования в файлах данных зоны, 86
    почтовые ретрансляторы, 132
    пункты назначения почты, 128
    распределение нагрузки, round robin, 328
    статистика запросов, 232
    устаревшие, решение проблем, 536
A6-записи, 361
    прямое отображение и, 362
aa, флаг, авторитативный ответ, 294
AAAA-записи, 360
ACL, 303
acl, оператор, 303
AFS (Andrew File System), 593, 619
AFSDB-записи, 593, 619
AIX, настройка DNS-клиентов, 165
allow-query, предписание, 375
allow-tranfer, предписание, 377
allow-update, предписание, 309
    TSIG, 372
allow-update-forwarding, предписание, 309
any (именованный список отбора адресов), 304
ANY-запросы, статистика, 233
APNIC (Asia Pacific Network Information Center), регистр, 80
ARIN (American Registry of Internet Numbers), регистр, 80
ARPAnet, 20
ASCII-символы в метках, 106
attempts, параметр, 147
attempts, предписание, 662
auth-nxdomain, предписание, 353, 529
AXFR-запросы, 232
    и IXFR-запросы, 318

B

BIND (Berkeley Internet Name Domain)
    DNS-клиенты, 136
    GSS-TSIG, динамические обновления, 605
    Windows 2000, возможные проблемы, 604
    версии, 11, 62, 374, 589
        важность использования свежих, 393

проблемы взаимодействия, 525
проблемы перехода, 524
инструкции и операторы (перечень), 643
история, 29
исходные тексты, получение, 61, 632
новые возможности, 62, 302
произношение, 98
работа с наименьшими полномочиями, 379
сопровождение, 180
списки рассылки пользователей и конференции Usenet, 64
список поиска в версии 4.8.3, 139
список поиска в версии 4.9, 140
статистика, 227
установка значений TTL в различных версиях, 87
файл настройки, 86, 98
версии и различия в синтаксисе, 98
инструкции, 643
операторы BIND 8, 649
операторы BIND 9, 654
создание, 98
BIND 8/9
изменение формата сообщений, 213
интервал сердцебиений, 589
механизм NOTIFY, 252
поддержка пошаговой передачи зоны, 63
сборка и установка на Linux-системах, 632
bindgraph, инструмент, 247
bind-users, список рассылки
помещение обновленного файла корневых указателей, 97
blackhole, предписание, 341
bogusns, инструкция, 341, 648
BSD Unix, операционная система, 21
bstat, инструмент, 233

C

C и C++
стиль комментариев, 98
cache, инструкция, 403, 644
check_soa, программа
пример на языке C, 561
пример на языке Perl, 575
check-names, инструкция, 648
chroot, команда, 380
CIDR (classless inter-domain routing)
бесклассовая междоменная маршрутизация, 79
запись адресов, 79
CLASS-поля (файлы данных зоны), 623
cname, категория, 214
CNAME-записи, 89, 153, 577, 612
и сообщение, 527
использование, 577
использование вместо A-записей, 91
множественные, 579
перестановка адресов, 330
переход к поддоменам, 298
порядок следования в файлах данных зоны, 87
статистика запросов, 232
удаление, 300
цепочки, 578
Compaq, 166
config, категория, 214
contrib/named-bootconf, 98
critical, приоритет, 207

D

d2, настройка (nslookup), 447, 453
datasize, предписание, 347, 647
db, категория, 214
db.cache, файл, 95
и временный корневой сервер имен, 271
обновление, 199
db.root, файл
внутренние корневые DNS-серверы, 401
создание для временных корневых DNS-серверов, 270
debug, настройка (nslookup), 446, 453
debug, предписание, 662
debug, приоритет, 207
default, категория, 209
BIND 8, 213
BIND 9, 215
defaultrouter, файл, 266
default_stderr, канал, 212
defname, настройка (nslookup), 446
DHCP, протокол, и динамические обновления, 304

Dial-Up Networking, 171
коммутируемые соединения, 584
удаленный доступ к сети, 171
Windows 98, 173
dialup, предписание, 589
dig, инструмент, 200
выполнение передачи зон, 471
ключи, 472
против nslookup, инструмента, 467
работа, 467
Digital Unix, настройка DNS-клиентов, 166
distfile, файл, 250
dname, параметр (res_search), 548
DNAME-записи, 361
обратное отображение, 364
создание псевдонимов, 578
DNS
RR-записи, 86
динамические обновления, 304
сообщения, 223
DNS (Domain Name System), 9, 24
email и, 125
NIS, совместное использование, 159
Windows 2000 и, 602
WINS и, 600
и файловые системы, сравнение, 24, 32, 34
история, 22
настройка узлов, 135
недостаточная документированность, 10
последствия настройки узлов, 151
принципы работы, 32
причины использования, 30
расширения системы безопасности, 414
расширения, рабочий комитет, 65
формат сообщений, 455, 545
DNS NOTIFY, 122, 312
поддержка в BIND, 63
DNS UPDATE, 63
DNSSEC (DNS Security Extensions), 414
динамические обновления, 434
использование записей, 426
производительность, 428
DNS-клиенты, 24, 49
BIND 4.9, 149
BIND 8.2.3, 136
nslookup и, 443
алгоритм поиска, 491
безопасность, 384
библиотечные функции, 547, 576
в отсутствие DNS-сервера, 150
длительные задержки при ответах, разрешение проблем, 534
имитация с помощью nslookup, 456
настройка, 135
параметры настройки, 147
примеры настройки, 149, 280
специфика настройки в различных системах, 157
обновление BIND и, 524
объекты Perl, 573
ограничение числа, 349
операторы файла настройки, 660
отказы в доступе, разрешение проблем, 535
перебои электроэнергии и, 265
программирование, 538
соместимость с DNS-серверами, 353
сообщение и, 527
сортировка адресов, 332
DNS-серверы, 24, 44
NS-записи, 616
root, команда, 448
авторитативные, 44
подключение по необходимости, 588
библиотечные функции, 556
ближайшие известные, 52
борьба с переутомлением, 247
время непрерывной работы, 220
время передачи сигнала, 54
вторичный, 109
выбор авторитативного, 54
выбор для анализатора, 660
выбор размещения, 242
выключенная проверка контрольных сумм UDP, 528
два в одном, 385
добавление, 249
в NOTIFY-списки, 317
доступ, 49
защита, 367
имитация запросов с помощью nslookup, 456
инициализация (пример отладки), 479
команда запуска, 184
команда копирования внутренней базы данных, 184

DNS-серверы
команда
немедленного прекращения работы, 188
останова и продолжения работы, 184
отображения информации о состоянии, 183
перезагрузки, 184
перенастройки, 184
создания отладочной информации, 185
создания статистики, 184
корневые, 50, 339
локальные, настройка, 150
множественные, nslookup, 443
наблюдение, 218
настройка узлов на использование, 135
нерекурсивные, 339
обновление BIND и, 524
ограничение частоты запросов, 342
ограниченные ретрансляцией, 323
опрашиваемые клиентами (поисковыми анализаторами), 142
основные положения для оценки необходимого количества, 240
отсутствие локальных для клиентов, 150
параллельная работа и сообщения syslog, 227
параметры настройки, 147
первичный мастер, 109, 323
переключение в nslookup, 451
подчиненные, 47
поиск для корневой зоны, 95
работающие вне основной площадки, 241
разделение функций, 382
разрешение проблем в случае отсутствия ответа, 462
расшифровка показателей статистики, 227
регистрация, 255
резервирование мощностей, 243
резервные, 151
сети и предпочтения, 338
служебные операции, 350
советы по размещению при добавлении новых, 249
соместимость с DNS-клиентами, 353
DNS-серверы
специализирующиеся на кэшировании, 252
типы, 47
удостоверенные (authenticated), 593
управление, 180
управляющие сообщения, настройка на прием, 182
установка, 83
важность наличия нескольких, 48
фальшивые, обход, 340
частично вторичные, 254
DNS-трафик, 246
оценка объема, 246
domain, инструкция, 136, 660
в BIND 4.9, 141
установка локального доменного имени, 137
domainname, NIS, 159
dumpdb, команда, 184
dynamic, приоритет, 207

E

email
и DNS, 125
error, приоритет, 207
/etc/defaultrouter, файл
превентивные меры на случай бедствий, 266
/etc/exports, файл, 152
/etc/host.aliases, файл, 156
/etc/host.conf, файл, Linux и, 169
/etc/hosts, файл, 22
nslookup, 444
и перебои, 269
пустой, 161
/etc/named.conf, файл
редактирование на узле вторичного DNS-сервера, 117
/etc/named.pid, файл, 189
/etc/netgroups, файл, 152
/etc/netsvc.conf, файл, 166
/etc/resolv.conf, 112
eventlib, категория, 214
exec, команда, 184
exports, файл, 152
extranet-сети, 22

F

failure, показатель, 239
fetch-glue, предписание, 339
FORMERR-ответы, 235
forward, предписание, 646
forwarders, инструкция, 322, 645
forwarders, предписание, 325
FP (Format Prefix), префикс формата, 356
FQDNs (fully qualified domain names), полные имена доменов, 34, 138
FTP, использование списка поиска, 152
FTP-архивы
  host, инструмент, 293
  и разрешение проблем, 511
FTP-сайты
  получение исходных текстов BIND, 62

G

gcc, компилятор
  параметры сборки BIND, 634
getpid, команда, 183
getrlimit(), системный вызов
  и сообщение, 219
gTLDs (generic top-level domains)
  родовые домены высшего уровня, 40, 76
    сервер неправильного типа, разрешение проблем, 536

H

h2n, инструмент, 108, 197
  переход к поддоменам, 299
halt, команда, 188
h_errno, переменная, 552
herror, функция, 552
HINFO-записи, 592, 613
  статистика запросов, 232
HMAC-MD5, алгоритм, 186
«host unreachable», сообщение, 144
host, инструмент
  получение и сборка, 293
  применение, 295
host.aliases, файл, 156
HOSTALIASES, переменная среды, 156
hostmaster в качестве адреса электронной почты, 88
hostname, команда
  доменное имя по умолчанию для первичного DNS-мастер-сервера, 112
  установка доменного имени по умолчанию для вторичного DNS-сервера, 119
  установка локального доменного имени, 137
hostresorder, инструкция, 168
HOSTRESORDER, переменная среды, 169
hosts, файл
  и перебои, 269
  пустой, 161
hosts.equiv, файл
  обновление, 155
HOSTS.TXT, файл, 22
  имена и номера сетей, 590
HP-UX, настройка DNS-клиента, 163

I

IBM, 165
ICANN (Internet Corporation for Assigned Names and Numbers), 40
ICMP (Internet Control Message Protocol), 266
ICMP Router Discovery Messages, 266
ICMP-сообщения об ошибках, 143
  и перебои электроэнергии, 265
ifconfig, команда, 264
ignoretc, настройка (nslookup), 447
IN, класс, 88
in-addr.arpa
  домен
    внутренние корневые серверы, 401
    делегирование, 285
    некорректно настроенные серверы, 295
    некорректное делегирование, разрешение проблем, 535
    поддомены, 287
  зона
    проверка делегирования, 81
    регистрация, 82
$INCLUDE, директива, 200
$INCLUDE, управляющий оператор, 204, 277, 610
include, инструкция, 645
include, оператор, 200

inet, предписание, 185
info, приоритет, 207
insist, категория, 214
Integrated Services Digital Network (ISDN), 594
InterNIC, 78
ip6.arpa, серверы имен, 365
ip6.int, пространство имен, 361
IPv4, транспорт
    адреса, 355
    настройка, 356
IPv6
    адреса, 354
        и IP-адреса версии 4, 355
        отображение, 360
    транспорт
        настройка, 359
IP-адреса, 56
    IP-сети, 78
    инструкция nameserver, 142
    поиск, 65
    превентивные меры на случай бедствий, 266
IP-сети
    IP-адреса, 78
    проверка регистрации, 78
IRIX
    архив обновлений, 168
    настройка DNS-клиентов, 168
irs.conf, файл, 165
ISDN-записи, 594, 620
ISO 3166, 41
IXFR, 317
ixfr-base, предписание, 320

J

JEEVES, 29

K

key, оператор, 185
keys, предписание, 372
keys, раздел, 185
KEY-записи, 416
    отправка на подпись, 430

L

lame-servers, категория, 214
LAN (Local Area Network)
    DNS, 31
    трафик и, 246
limit, инструкция, 647
    limit datasize, 347, 647
    limit stacksize, 348
    limit transfers-in, 343, 647
    limit transfers-per-ns, 342, 647
Linux
    BIND 8/9, сборка и установка, 632
    настройка DNS-клиентов, 169
listen-on, предписание, 356
listen-on-v6, предписание, 359
load, категория, 214
LOC-записи, 596
LOCALDOMAIN, переменная среды, установка локального доменного имени, 137
localhost (именованный список отбора адресов), 304
localnets (именованный список отбора адресов), 304
logging, оператор, 207
log-файл (BIND 8/9), 206
log-файлы журналов, 308
loopback-адрес, 94
loopback-сеть, 94

M

magic cookie, 159
maintain-ixfr-base, предписание, 319
maintenance, категория, 214
many-answers, формат, 346
masters, предписание, 357
match-clients, предписание оператора view, 326
max-ncache-ttl, предписание options, 352
max-refresh-time, предписание, 345
max-retry-time, предписание, 345
max-transfer-idle-in, предписание, 345
max-transfer-idle-out, предписание, 345
max-transfer-time-in, предписание, 344
max-transfer-time-out, предписание, 345
MB-записи, 70, 613
MD- и MF-записи, 614
MD5, 369
MD-записи, 126
MF-записи, 126
MG-записи, 70, 614
Microsoft, 170

MINFO-записи, 614
min-refresh-time, предписание, 345
min-retry-time, предписание, 345
Modify Tool, 286
MR-записи, 615
mtime (временная отметка изменения файла Unix), 313
multiple-cnames, предписание, 330
MX-записи, 126, 131, 615
  sendmail, 154
  исключение, 131
  ограничение производительности, 583
  статистика запросов, 232
  электронная почта и брандмауэры, 404
MX-записи и RT-записи, 595

N

named (демон DNS-сервера), 98
  syslog-сообщения, 218
named, страницы руководства
  поиск исполняемого файла, 109
named.conf, файл
  два DNS-сервера в одном, 385
  добавление оператора key, 186
  разница между первичным DNS-мастер-сервером и вторичным DNS-сервером, 116
  специального кэширующего DNS-сервера, 253
  узел-бастион, 410
  частично вторичного DNS-сервера, 254
named.pid, файл, 189
  изменение места жительства, 205
named.root, файл, 95
named.run, файл, 208, 477
named.stats, файл, 184
  изменение места жительства, 205
  создание статистики в BIND 9, 237
  сохранение статистики в BIND 4.8 и BIND 8, 230
named_dump.db, файл, 499
  изменение места жительства, 205
named-bootconf, программа, 98
namedroppers, список рассылки, 65
  помещение обновленного файла корневых указателей, 97
named-xfer, инструмент, 190, 496
  изменение места жительства, 205
nameserver, инструкция, 136, 142, 660
nametype, 310
ncache, категория, 214
ndc (name daemon controller), 111, 181
  syslog-сообщения, 218
  команды, эквивалентные сигналам (перечень), 188
ndots, настройка, 661
ndots, параметр, 147
negative_timeout, параметр, 168
Net::DNS, модуль Perl, 573
    динамические обновления с TSIG-подписями, 373
netgroups, файл, 152
netsvc.conf, файл, 166
Network Solutions Inc.
  регистр и регистратор, 67, 78
  цены на регистрацию, 81
«network unreachable», сообщение, 144
NFS (Network File System), 152
NFSNET, отчет по трафику, 247
NIC (Network Information Center), 22, 517
NIS (Network Information Service), 31, 157
  nslookup, 444
  SunOS 4.x, неиспользование, 161
  разрешение проблем, 495
NIS, файл сборки, 159
NLA (Next-Level Aggregators), 355
no-check-names, параметр, 148
no-check-names, предписание, 662
nod2, настройка (nslookup), 447
nodebug, настройка (nslookup), 446
nodefname, настройка (nslookup), 446
–noexpired, ключ, 184
no-fetch-glue, настройка, 340, 383, 646
noignoretc, настройка (nslookup), 447
none (именованный список отбора адресов), 304
norecurse, настройка (nslookup), 447
no-recursion, настройка, 339, 383, 646
nosearch, настройка (nslookup), 447, 449
NOTFOUND, условие, 162
notice, приоритет, 207
notify, категория, 214
NOTIFY, механизм (BIND 8/9)
  время распространения, 252
notify-source, предписание, 359
notify-source-v6, предписание, 360

NOTIFY-сообщения
исходный адрес, 359
объявления и ответы, 313
посылка на альтернативный порт, 357
notrace, команда, 185
novc, настройка (nslookup), 447
NS-записи, 89, 616
и сообщение, 526
порядок следования в файлах данных зоны, 86
статистика запросов, 232
NSAP-записи
статистика запросов, 233
nscd, демон, 163
nsd, демон, 168
NSFNET, сеть, 21
ns_get32, функция, 561
ns_init_parse, функция, 556
nslookup, инструмент, 11, 442
IP-адреса, поиск, 65
завершение сеанса, 467
замена dig, инструментов, 467
имитация DNS-сервера, 456
настройки-переключатели, 446
обратные запросы, 58
пакетный и диалоговый режимы, 445
переключение между DNS-серверами, 451
поиск
CNAME-записей, 580
SOA-записей, 76
поддоменов, 68
предмет поиска, 465
программирование на языке командного интерпретатора, 539
разрешение проблем, 461
решаемые задачи, 449
тестирование вторичных DNS-серверов, 119
тестирование первичного DNS-мастер-сервера, 111
.nslookuprc, файл, 449
ns_msg_count, функция, 557
ns_msg_get_flag, функция, 557
ns_msg_id, функция, 557
ns_name_compress, функция, 559
ns_name_skip, функция, 560
ns_name_uncompress, функция, 560
NSORDER, переменная среды, 166
ns_parserr, функция, 558
ns_put32, функция, 561
nsswitch.conf, файл, 162, 164, 496
Linux и, 169
NSTATS-сообщения, 221
ns_update(), функция DNS-клиента, 306
nsupdate, инструмент, 306
динамические обновления с TSIG-подписями, 373
NTP (Network Time Protocol), 372
null, канал, 212
NULL-записи, 616
nxdomain, показатель, 238
nxrrset, показатель, 238
NXT-записи, 422

O

one-answer, формат, 346
options, инструкция, 136, 147, 646
options allow-query, 375
options auth-nxdomain, 529
options cleaning-interval, 350
options coresize, 348
options debug, 147, 662
options fetch-glue, 339
options files, 349
options forwarders, 322
options forward-only, 323, 646
options interface-interval, 351
options listen-on, 357
options listen-on-v6, 359
options ndots, 147, 661
options no-check-names, 662
options no-fetch-glue, 340, 383, 646
options no-recursion, 339, 383, 646
options notify-source, 359
options notify-source-v6, 360
options query-log, 226, 504, 647
options query-source, 358
options recursion, 339
options recursive-clients, 349
options rfc2308-type1, 353
options rotate, 663
options serial-queries, 350
options statistics-interval, 351
options timeout, 662
options transfer-format, 346
options transfer-source, 358
options transfer-source-v6, 360
options treat-cr-as-space, 353

$ORIGIN, управляющий оператор, 200, 277, 610
изменение суффикса по умолчанию, 204
os, категория, 214

P

packet, категория, 214
panic, категория, 215
parser, категория, 215
Perl
h2n (инструмент на этом языке), 197
Net::DNS, модуль, программирование с, 573
Socket.pm, 493
ping, инструмент, 91
и параметр rotate, 148
port, настройка (nslookup), 447
«port unreachable», сообщение, 143
postmaster в качестве адреса электронной почты, 88
prereq, команды (nsupdate), 306
primary, инструкция
добавление, 123
primary, строка файла настройки (BIND 4), 99, 101
provide-ixfr-server, предписание, 320
ps, команда, 189
PTR-записи, 92, 617
и сообщение, 527
отсутствие, 463, 510
порядок следования в файлах данных зоны, 86
PTR-запросы
статистика запросов, 232
PX-записи, 622

Q

QCLASS-поля, 626
QTYPE-поля, 626
queries, категория, 215
querylog (qrylog), команда, 185, 227
query-log, предписание, 647
query-source, предписание, 358
querytype, настройка (nslookup), 447, 449
quit, команда, 185

R

RAXFR, показатель, 236
RCODE, поле, 625
rcodes (response codes), коды ответов, 455
rdist, инструмент, 250
RDupQ, показатель, 235
RDupR, показатель, 235
reconfig, команда, 184
recurse, настройка (nslookup), 447, 541
recursion, показатель, 238
recursive-clients, предписание, 349
referral, показатель, 238
refresh, команда, 188
reload, команда, 184
request-ixfr, предписание, 321
RErr, показатель, 235
_res, структура, 552
RES_DEBUG, флаг, 147
res_init, функция, 551
res_mkquery, функция, 550
resolv+, библиотека, 158
Linux и, 169
resolv.conf, файл, 136
AIX и, 165
Linux и, 169
nslookup и, 443, 465
SunOS и, 160
инструкции nameserver, 143
комментарии в, 149
ошибки синтаксиса, 521
перебои и, 264
разрешение проблем, первый из DNS-серверов не отвечает, 465
система HP-UX, 163
установка локального доменного имени, 112
response-checks, категория, 215
Responsible Person, ответственное лицо (RP-записи), 196
res_query, функция, 549, 566
RES_RETRY, переменная среды, 165
res_search, функция, 548
res_send, функция, 551
restart, команда, 184
RES_TIMEOUT, переменная среды,165
RFail, показатель, 235
RFC 1034, 24, 283
RFC 1035, 24, 538
файлы данных зоны, 609
формат сообщений, 623

RFC 1101, 590
RFC 1183, 592, 619
RFC 1664, 622
RFC 1876, 596
RFC 1886, 360
RFC 1918, 341
RFC 2136, 304
RFC 2308, 529
RFC 2672, 364
RFC 2673, 364
RFC 2845, 368
RFC 882/883, 23
rfc2308-type1, предписание, 353
RFErr, показатель, 235
RFwdQ, показатель, 235
RFwdR, показатель, 235
.rhosts, файл, 534
    обновление, 155
RIPE, регистр, 80
RIQ, показатель, 234
RLame, показатель, 236
rlogin, команда, в доступе отказано, 534
    и список поиска, 152
rndc, 111, 185
    создание статистики (BIND 9), 237
RNotNsQ, показатель, 237
RNXD, показатель, 235
root в качестве адреса электронной почты, 88
root, команда, 448
root, пользователь, работа BIND, 379
ROpts, показатель, 236
rotate, параметр, 148
rotate, предписание, 663
round robin, распределение нагрузки, 328
    и CNAME-записи, 579
    и демон nscd, 163
route, команда, 264
RPC (Remote Procedure Call), 153
    и резервирование мощностей, 244
RP-записи, 89, 196, 620
RQ, показатель, 234, 246
RR, показатель, 234
rrset-order, предписание, 330
RRset-наборы и динамические обновления, 305
RR-записи, 38, 86, 609
    внезональные, 524
    динамические обновления, 304
RR-записи
    и функция ns_parserr, 558
    объекты Perl, 574
    повтор последнего имени, 102
    типы, 592
        и полный перечень, 612
    устаревшие, разрешение проблем, 536
    формат данных, 629
    явное задание значений TTL, 260
rsh, команда, в доступе отказано, 534
    и список поиска, 152
rsync, инструмент, 250
RT-записи, 595, 621
RTCP, показатель, 236
RTT (round-trip time), см. время передачи сигнала, 54

S

SAns, показатель, 236, 247
SDupQ, показатель, 236
search, инструкция, 136, 140, 660
search, настройка (nslookup), 447
secondary, инструкция, 644
secondary, операторы
    добавление, 123
secure_zone, записи, 376
security, категория, 215
sendmail, 128
    DNS и, 153
    записи ресурсов, 91
    класс w и, 134
    параметр rotate и, 148
sendmail.cf, файл, 154, 155
sendto(), системные вызовы, 237
serial-queries, предписание, 350
SErr, показатель, 237
server, команда (nslookup), 451
    фальшивые серверы, 341
SERVFAIL, ошибки, 508
SERVFAIL-ответы, 235
set all, команда (nslookup), 521, 531
set, команда (nslookup), 446
setrlimit(), системный вызов
    и сообщение, 219
SFail, показатель, 237
SFErr, показатель, 237
SFwdQ, показатель, 236
SFwdR, показатель, 236
SIG-записи, 420

Silicon Graphics, 168
slave, инструкция, 645
SMIT (System Management Interface Tool), инструмент управления системой AIX, 166
SNaAns, показатель, 237
SNXD, показатель, 237
SOA-записи, 70, 88, 540, 617
  @-нотация в, 102
  TTL и, 260
  значения полей, 120
  значения, рекомендуемые документом RFC 1537, 122
  изменение значений, 261
  ограничение числа запросов, 350
  отсутствие NS-записей, 526
  поиск для поддоменов, 75
  получение порядкового номера в программе на языке C, 561
  получение порядкового номера на Perl, 575
  порядок следования в файлах данных зоны, 86
  статистика запросов, 232
  увеличение порядковых номеров, 192
Socket.pm, инструмент, 493
Solaris 2.x, настройка DNS-клиентов, 161
sortlist, инструкция, 136, 146, 336, 645, 661
sortlist, предписание, 337
spcl.DOMAIN, файл, 199
special, инструкция, 251
src/bin/named-bootconf, 98
SRI-NIC, узел, 22
SRV-записи, 597
SSysQ, показатель, 236
start, команда, 184
statistics, категория, 215
stats, команда, 184
status, команда, 183
stderr, канал, 212
stop, команда, 184
stub, инструкция, 646
success, показатель, 238
SUCCESS, условие, 162
Sun, 161
SunOS 4.x, 528
  настройка DNS-клиентов, 157
support-ixfr, предписание, 320
svc.conf, файл, 167
svcsetup, инструмент, 167
syslog
  запуск перед стартом сервера имен, 109
  регистрация запросов, 185
syslog, журнал
  запись статистики, 245
  поиск сообщений об ошибках от вторичного DNS-сервера, 119
  проверка наличия сообщений об ошибках (пример), 280
  проверка наличия сообщений об ошибках от первичного DNS-мастер-сервера, 110
syslog, сообщения, 218
  первичный DNS-сервер, 110
syslogd
  документация, 110
syslog-каналы, 212
syslog-сообщения
  регистрация в log-файле (BIND 8/9), 207
System Management Interface Tool, 166

T

TCP (Transmission Control Protocol)
  виртуальные соединения, 447, 553
  шлюзы приложений, 392
TCP/IP, протокол, 20
  nslookup и, 447
TCP-сокеты, посылка сообщений с помощью ndc, 181
telnet и список поиска, 152
timeout, параметр, 148
timeout, предписание, 662
TIS Firewall Toolkit, 393
TLA (Top-Level Aggregators), 355
TLDs, 40
top, инструмент, 244
trace, команда, 185
transfer-format, предписание, 346
transfers-in, предписание, 343, 647
transfer-source, предписание, 358
transfer-source-v6, предписание, 360
transfers-per-ns, предписание, 342, 647
Transmission Control Protocol/Internet Protocol, 20
Tru64 Unix, настройка DNS-клиентов, 166

Trusted Information Systems, 393
TRYAGAIN, условие, 162
TSIG (transaction signatures), 368
  настройка, 370
  применение, 372
  разрешение проблем, 530
TSIG-записи, 369
TSIG-подписи в динамических обновлениях, 309
  и Windows 2000, 605
TTL (time to live), 60
  значение TTL для отрицательных ответов, 121
  изменение значений, 259
  информация о корневых серверах, 97
  минимальное, 262
  некорректное, 352
  определение предельных значений, 352
  отрицательного кэширования, 121
  по умолчанию, 260, 263
  распределение нагрузки round robin и, 329
  стандартное значение для зон, 87
  умолчания в BIND 8/9, 529
TXT-записи, 196, 618
  secure_zone-записи и, 376
  статистика запросов, 233

U

UDP-пакеты ответных сообщений, 223, 256, 447
  выключенная проверка контрольных сумм, 528
  переполнение, 466
UNAVAIL, условие, 162
Unix, операционная система
  BSD версии, 21
  комплектация сетевыми программами, 62
Unix-сокеты, посылка сообщений с помощью ndc, 181
Unix-специальные случаи настройки DNS-клиентов, 166
update, категория, 215
update, команда (nsupdate), 306
update-policy, предписание оператора zone, 310
  TSIG и, 372
UPS (uninterruptible power system), система бесперебойного питания, 267
URL
  APNIC, регистр, 80
  ARIN, регистр, 80
  BIND, 64
  FTP-узлы, 16
  HP-UX, архив обновлений, 164
  ICANN, 40
  IRIX, архив обновлений, 168
  Modify Tool, 286
  Net::DNS, модуль, 573
  RIPE, регистр, 80
  us, домен, 74
    правила делегирования, 76
  WebMagic, 71
  Webmin, 290
  whois lookup, служба, 68, 71
  безопасность и уязвимость, 374
  глобальное ориентирование, 596
  регистраторы, список, 78
us, домен
  поиск в пространстве имен, 74
USAGE-сообщения, 220
/usr/etc/resolv.conf, 136

V

/var/run/ndc, сокет, 181
vc, настройка (nslookup), 447
view, оператор, 326

W

$=w, класс, 154
warning, приоритет, 207
WebMagic, 71
Webmin, инструмент, 290
Windows 2000
  и DNS, 602
  настройка DNS-клиентов, 175
Windows 95, настройка DNS-клиентов, 170
Windows 98, настройка DNS-клиентов, 172
Windows NT
  и ICMP Router Discovery Messages, 266
  настройка DNS-клиентов, 173
  обработка возврата каретки, 353
WINS (Windows Internet Name Service)
  использование вместо DNS, 31

WINS-записи, фирменные, 525
WINS-серверы, 600
Winsock 2.0, 172
WKS-записи, 619

X

X0.hosts, файл, 155
X.121, адреса, 594
X25-записи, 594, 622
X.400, 622
xfer-in, категория, 215, 216
xfer-out, категория, 215
xfrnets, инструкция, 377, 648
XSTATS-сообщения, 221

Y

Yellow Pages, 136, 157, 167
ypcat, команда, 495
ypmatch, команда, 495
ypserv, программа, 157

Z

zone, операторы, 99, 652, 658
  zone allow-transfer, 377
  добавление, 123

А

абсолютные доменные имена, 34, 138
авторитативность, раздел (DNS-сообщения), 627
авторитативные DNS-серверы, 44
  подключение по необходимости, 588
авторитативные ответы nslookup, 450
авторитативный ответ, флаг aa, 294
агрегаторы
  высшего уровня (TLA), 355
  следующего уровня (NLA), 355
администраторы
  как связаться по вопросам, связанным с поддоменами, 70, 75
  контактные адреса для зон, 88
  обновления файла корневых указателей, 97
  родительских зон
    установка контакта, 257
адрес loopback-интерфейса, 142
  и множественные инструкции nameserver, 143
адреса, 356
  X.121, 594
адреса
  круговая перестановка, 90
  разрешение проблем, 510
    отсутствие PTR-данных, 463
  сортировка, 90, 332
  электронной почты
    в качестве аргументов RP-записей, 196
адрес-имя
  отображение
    DNAME-записи, 364
    адресов IPv6, 360
    динамические обновления, 305
  преобразование, 55
адресные
  записи, 90
  запросы, 232
  типы, 38

Б

база данных, дамп, чтение, 499
бедствия, 263
безопасность, 367
  DNS-клиенты, 384
  DNS-серверы, 373
  брандмауэры, 389
  возможности BIND, 63
  выбор узла, 243
  данные зоны, важность создания резервных копий, 119
  запросы, ограничение, 375
  отвергнутые запросы, 464
  последние исправления в BIND, 62
  предотвращение несанкционированной передачи зоны, 377
  сообщение, 224
  уязвимости, 374
бесклассовая междоменная маршрутизация (CIDR), 79
бит-строковые метки, 364
Бичер, Брайан, 328
ближайшие известные DNS-серверы, 52
брандмауэры, 389
  отправка сообщений электронной почты, 404
братские узлы, и метки доменных имен, 34

В

веб-серфинг и резервирование мощностей, 243
ведение log-файла (BIND 8/9)
  необходимость в экспериментах, 209
версии, 393
взломщики, версии BIND, 374
видимость пространства имен, 408
виды, 326
  два DNS-сервера в одном, 388
  узел-бастион, 413
виртуальные соединения, 447, 553
внезональные записи ресурсов, 524
внешние имена
  поиск с помощью nslookup, 113
внутренние корневые DNS-серверы, 399
внутренние узлы, CNAME-записи, 577
возврат каретки
  обработка, 353
волшебный токен, 159
восьмибитные октеты, запись IP-адреса, 56
временная отметка изменения файла Unix (mtime), 313
временные корневые DNS-серверы, 269
время жизни, 60
время передачи сигнала (RTT), 54
время распространения, 251
время устаревания
  и перебои, 269
время, синхронизация и TSIG, 371
вспомогательные инструменты
  Socket.pm, 493
вторичные DNS-серверы, 47, 254
  syslog, поиск ошибок в журнале, 119
  данные не могут быть загружены, 508
  добавление, 250, 286
  загрузка зон со вторичных серверов, 251
  запуск, 115, 119
    автоматический, 119
  команда перезагрузки, 184
  необходимые предосторожности при добавлении, 252
  первичный DNS-мастер-сервер, 116
  перебои и, 269
вторичные DNS-серверы
  пример настройки, 282
  проверка зоны (пример отладки), 488
  резервные копии файлов данных, 119
  ретрансляторы и, 322
  тестирование с помощью nslookup, 119
  установка, 116
вторичный раздел (сообщения DNS), 456
выбор
  в домене us, 72
  доменных имен, 66
  доменов третьего уровня, 73
  поддоменов, 67
  регистратора, 78
  ретранслятора, 326
выключенная проверка контрольных сумм UDP, 528
вычислительно необратимые хэш-функции, 369

Г

географические обозначения, 41
главные файлы, 609
глобальное ориентирование, 596
города, производные доменные имена, 73

Д

делегирование, 42
  заглушки, 297
  и поддомены, 76, 517
  некорректное, 258
  поддоменов, 520
  проверка, 293, 518
  устаревшая информация, 536
делегированные DNS-серверы, 382
дефис (-) в именах узлов, 106
действительные порядковые номера против целочисленных, 193
действия при использовании источников информации, указание, 162
динамические обновления, 304
  DNSSEC, 434
  и Windows 2000, 602
  ограничение с помощью TSIG, 372

поддержка в BIND, 63
с GSS-TSIG подписями и Windows 2000, 605
директивы, 200
добавление
DNS-серверов, 249
вторичного DNS-сервера, 250, 286
зон, 123
оператора primary, 123
операторов zone, 123
первичного DNS-мастер-сервера, 250
узлов, 191, 510
доменные имена, 33, 34
абсолютные, 138
в качестве аргументов RP-записей, 196
выбор, 66
выбор в родовых доменах высшего уровня, 76
данные записей ресурсов, 629
добавление суффикса по умолчанию, 101
завершающая точка (.), 513, 610
конфликты имен, 23
локальные, 136
невозможность поиска для, 531
отсутствие локального, 531
отображение, 89
поиск для локальных и внешних с помощью nslookup, 111
полные (FQDN), 34
порядок перебора, 141
правила чтения, 41
псевдонимы, 27
регистрация, 80
серверы, 47
умолчания для настроек, 448
упаковка и распаковка, 546
установка для узла, 112
установка значения ndots, 147
хранение, 546
домены, 25, 34
arpa, 40
com, 39
edu, 39
gov, 39
in-addr.arpa, 56, 285
mil, 40
net, 40
org, 40
us
url-адрес, 74
второго уровня, 38
имена, 67
высшего уровня (TLDs), 38, 39
выбор в домене us, 72
выбор, международные домены, 67
изучение структуры с помощью nslookup, 68
имена поддоменов, 276
родовые, 76
делегирование, 42, 517, 520
устаревшая информация, 536
зоны, 44
и поддомены, 37
имена, 25
интернационализация, 40
международные домены, 67
по умолчанию, 522
отсутствие указания, 522
получение информации, 67
домены США, 72
семь доменов высшего уровня, 39
служба whois, 68, 71
третьего уровня, выбор, 73
уровни, 38
дополнительные зоны
команда выполнения служебных операций, 188
дополнительный раздел (DNS-сообщения), 546, 627
и достоверность, 503
достоверность, показатели, 502
доступ
rlogin и rsh, невозможно произвести проверку прав доступа, 534
к службам запрещен, разрешение проблем, 535
доступность, 130
выбор узла, 242
дублирующиеся запросы, 237

Ж

журналов, файлы (.jnl), 308
и инкрементальная передача зоны, 319

З

завершающая точка (.)
в доменных именах, 513

заглушки, зоны, 297
заголовок, формат (DNS-сообщения), 624
загрузочный файл (файл настройки BIND)
инструкции, 643
операторы BIND 8, 649
операторы BIND 9, 654
ошибки синтаксиса, 511, 513
редактирование, 114, 119
записи
для DNS-серверов, 89
канонических имен, 90
начала авторитативности, 70
почтового назначения (MD), 126
почтовой ретрансляции (MF), 126
ресурсов, см. RR-записи, 38, 609
записи-указатели, 92
запрос рекурсии (want recursion), 455
запросы
адресов, 232
время передачи сигнала, 54
дублирующиеся, 237
за секунду, 239, 245
и сообщения syslog, 226
и статистика, 227
имитация DNS-сервера с помощью nslookup, 456
итеративные (нерекурсивные), 540
DNS-серверы с разделяемой функциональностью, 382
команда регистрации, 185
неавторитативные ответы nslookup, 450
неизвестного типа, 232
нерекурсивные, 322
обратные, 58, 456, 464
ограничение, 375
ограничение для сервера, 342
отвергнутые, 463
отображение, 453, 456
переключение между DNS-серверами, 452
переключения (пример отладки), 486
регистрация, 503
рекурсивные, 322
системные, 236
типы, 52
успешные (пример отладки), 482
запуск
вторичных DNS-серверов, 119
первичного DNS-мастер-сервера, 109
автоматический, 114
затененное пространство имен, 407
защищенные сегменты, 425
звездочка (*) в качестве маски, 582
значения предпочтения
исключение MX-записей, 131
приращения, 134
зональные данные
добавление в файлы данных зон, 195
узла-бастиона, защита, 411
зоны, 26, 44
RP-записи, 196
SOA-записи, поиск, 70, 75
безопасность, 464
внезональные записи ресурсов, 524
данные не могут быть загружены, 510
добавление, 123
домены, 44
задание новых порядковых номеров, 194
и данные, важность создания резервных копии вторичными DNS-серверами, 119
как связаться с администратором, 88
механизм уведомления об изменениях, 63
невозможность загрузки данных, 508
подключение к сети Интернет, 30
подпись, 429
поиск имен с помощью nslookup, 114
пример, 84
проверка делегирования, 517, 520
проверяемые DNS-сервером (пример отладки), 488
прямого отображения, 126, 128
регистрация, 81
ретрансляции, 324
применение, 398
создание поддоменов, 276
стоимость регистрации, 81
технические контакты, 70
уведомление об изменении, 312
увеличение интервалов обновления, 251

И

идентификаторы процессов, 188
    команда отображения, 183
избыточность в доменных именах, 546
имена
    доменов, 25, 275
    конфликты
        ACL, 304
    разрешение, 49
    сетей, 590
    узлов
        правила именования, 106
        проверка корректности, 105
имя-адрес, отображение, 89
    A6-записи, 362
    адресов IPv6, 360
    динамические обновления, 305
инициализация DNS-серверов (пример отладки), 479
инкрементальная передача зон (IXFR), 317
    BIND 8/9, настройка, 318
инструкции, 136
    перечень, 643
инструменты
    bindgraph, 247
    bstat, 233
    dig, 200
        применение, 467
    h2n, 108, 197
    host, 293
        применение, 295
    Modify Tool, 286
    named-bootconf, 98
    named-xfer, 190, 496
        изменение места жительства, 205
    nslookup, 442
    nsupdate, 306
    rdist, 250
    rsync, 250
    top, 244
    Webmin, 290
интервалы
    обновления, 120, 261
        DNS NOTIFY, 312
        увеличение, 251
    ожидания
        nslookup, 443, 448
        и запросы DNS-клиента, 143
интервалы
    повторения, 121, 261
    сердцебиений, 589
    сканирования интерфейсов, 351
    служебных операций, 350
    создания статистики, 351
    устаревания, 262
    чистки, 350
интернационализация, 40
Интернет
    и DNS, 30
    и интернет-сети, 21, 22
    история, 20
    коммутируемые соединения, настройка DNS, 584
    пространство доменных имен, 39
    типы доступа, 83
интернет- и интранет-сети, 22
интернет-сети
    DNS, 30
    интранет, 22
    и Интернет, 21, 22
интерфейсы, сетевые
    интервал сканирования, 351
информация о корневых DNS-серверах, 97
исполняемый файл сервера, поиск с помощью страниц руководства, 109
источники информации
    BIND, 64
    Net, 373
    безопасность, уязвимости, 374
    брандмауэры, 390
    об узлах, 162
исходные тексты BIND
    получение, 61, 632
итеративное разрешение, 52, 53
итеративные (нерекурсивные) запросы, 52
    и SOA-записи, 540

К

каналы, 206, 211
    настройка, 207
канонизация, 91
    в программе sendmail, 153
    фильтр, 155
каноническое доменное имя, 28
    почтовые ретрансляторы, 132
каталоги, организация файлов данных зон, 200

категории, 206, 213
default, 209
классы, 38, 623
Chaosnet, 38
class, настройка (nslookup), 448
HS (Hesiod), 38
интернет-сетей, 38
сообщение, 225
ключи
командной строки инструмента h2n, 197
пара, 415
создание, 429
смена, 437
командная строка, отладка, 477
команды ndc (перечень), 182
комментарии
#, символ, 99
C++-стиль, 98
C-стиль, 98
в файлах данных зоны, 87
в файле resolv.conf, 149
в файле настройки BIND, 98
коммутируемые соединения, 584
подключение вручную, 586
конференции Usenet, посвященные пакету BIND, 64
информация о проблемах безопасности, 374
конфликты имен, 23
решение проблемы, 28
координаты, записи, 596
корневая зона, файл данных, 270
корневой домен, 33
корневой узел
точка (.), метка, 24
корневые DNS-серверы, 50, 339, 399
nslookup, 448
кэширование, 59
перебои и, 269
криптографические контрольные суммы, 369
круглые скобки ()
в SOA-записях, 89
группировка данных, 611
круговая перестановка адресов, 90
кэширование, 58
интервал чистки, 350
определение предельных значений TTL, 352
отрицательное, 491, 528
кэширование
отсутствие данных кэша, 514
ретрансляторы и, 321
устаревших данных, разрешение проблем, 536
кэш-файл, 97

Л

локальное доменное имя, 136
локальные
адреса, поиск
nslookup, 113
разрешение проблем, 531
имена, поиск
nslookup, 112
разрешение проблем, 531
локальный DNS-сервер, 150, 451

М

магистрали, 21
TLA, подключение, 355
маршрутизация почты, 125
петли, предотвращение появления, 127, 131
маски, 582
в MX-записях, 404
материнство, 272
поддомены
в домене in-addr.arpa, 287
переход, 298
создание, 276
советы, 273
места
производные доменные имена, 73
метки
в имени домена, 25
разделяющая точка (.), 25
корневого узла
. (точка), 24
метки доменных имен, 33, 106
данные записей ресурсов, 629
многосетевые узлы
сортировка адресов, 332

Н

настройка
DNS-клиентов, 135
примеры, 149, 280
TSIG, 370

настройка
инкременальной передачи зоны в BIND 9, 320
инкрементальной передачи зоны в BIND 8, 319
каналов, 207
предельные значения TTL, 352
синтаксис различных версий BIND, 98
транспорта IPv4, 356
транспорта IPv6, 359
узлов, 135
последствия, 151
неавторитативные ответы nslookup, 450
неверные ответы, разрешение проблем, 533
некорректное TTL, 352
некорректное делегирование, 226, 236, 258
диагностика, 520
непрерывное арифметическое пространство, 194
нерекурсивные
DNS-серверы, 339
запросы, 52, 322
и SOA-записи, 540
разрешение, 52
номера сетей, 590
нулевой
адрес, 142
порядковый номер, 194

О

обновление
HP-UX, 164
динамические, 304
файла корневых указателей, 199
файлов авторизации, 155
файлов данных зон, 190
обработка почты, 126
обратное отображение, 85
DNAME-записи, 364
адресов IPv6, 360
обратные запросы, 58, 456, 464
ограничение числа открытых файлов, 219
ограниченные DNS-серверы, 323
однородность
выбор узла, 243
округа
производные доменные имена, 73
октеты, 56, 611
данные записей ресурсов, 629
порядок передачи данных, 628
упаковка доменных имен, 547
формирование подсетей, 288
основной (первичный) DNS-сервер, 99, 122
вторичные DNS-серверы и, 116
запуск, 109
автоматический, 114
команда перезагрузки, 184
несколько экземпляров, 122
против вторичного, 123
сообщение, 221
тестирование с помощью nslookup, 111
файл настройки BIND (пример), 99
основной раздел (DNS-сообщения), 626
основные принципы разбиения пространства доменных имен, 274
отвергнутые запросы, 463
ответные сообщения
ограничение MX-записей, 583
отображение, 453, 456
разбор, 561
ответы
неверные или противоречивые, разрешение проблем, 533
от неизвестных источников, 523
отрицательные, и совместимость, 353
раздел (DNS-сообщения), 627
скорость получения, 534
открытые файлы, ограничение числа, 527
открытый ключ, шифрование, 415
отладка, 473
named-xfer, инструмент, 496
nslookup, 446
включение, 477
включение при исследовании перегруженных DNS-серверов, 249
команда выключения, 185
отладочный вывод, 473
уровни, 473
флаг для DNS-клиентов/серверов, 147
чтение диагностики, 478

отладочные сообщения
регистрация в log-файле (BIND 8/9), 207
отображение, 85
адреса в имя, 55, 85, 92
имен и номеров сетей, 590
имени в адрес, 85, 89
прямое, 85
отрицательное кэширование, 58, 491, 528
отсутствие
PTR-записей, 510
данных кэша, 514
делегирования поддоменов, 517
ошибки
rcodes, 455
в файле resolv.conf, 521
при делегировании поддоменов, 518
синтаксические, 511, 513
функция herror, 552

П

пакетные фильтры, 390
память
и ограничения ресурсов, 347
изменение размера сегмента данных, 347
использование, 244
хранение доменных имен, 546
первичный DNS-мастер-сервер, 47
большое число вторичных серверов, 250
в нескольких экземплярах, 122
добавление, 250
запуск, 109
автоматический, 114
и вторичные DNS-серверы, 116
и ретрансляторы, 322
команда перезагрузки, 184
не перезагружен, 507
против вторичного, 123
сообщение, 221
тестирование с помощью nslookup, 111
файл настройки BIND (пример), 99
перебои, 263
перевод, 85
передача зоны, 47, 116
DNS NOTIFY, 312
выполнение с помощью nslookup, 459
передача зоны
ограничение, 342
повышение эффективности, 346
порождаемые процессы, 189
предотвращение несанкционированной, 377
резервные копии файлов, 222
с помощью nslookup, 444
с помощью инструмента dig, 471
сообщение, 221
тестирование вручную, 497
тестирование с помощью named-xfer, 496
фирменные записи WINS, 525
период устаревания, 121
петли маршрутизации
предотвращение появления, 127, 131
пиковые периоды активности и наблюдение, 246
питание резервное, 266
ПО, BIND
получение, 632
повторение, 52, 53
запроса, 448
поддомены, 25, 37
in-addr.arpa, 287
выбор, 67
выбор имен, 274
делегирование, 47, 273, 276
границы октетов, 287
проверка, 518
домены и зоны, 44
и директивы, 200
определение числа создаваемых, 273
поиск SOA-записей, 75
поиск с помощью nslookup, 68
правила чтения, 41
размеры, 273
связь с администраторами, 70, 75
создание, 273, 276
стоимость регистрации, 74
подключение
коммутируемые соединения, 584
по необходимости, 588
подсети, 645, 661
сортировка адресов, 336
указание в инструкции sortlist, 146
формирование
на границе октетов, 288

подчеркивание (_), запрет на
использование в именах узлов, 106
подчиненные DNS-сервера, 47
поиск, 57
CNAME-записей, 580
IP-адресов, 65
алгоритм, 491
в пространстве имен домена us, 74
разрешение проблем, 534
поисковые анализаторы, 24
поле данных (RR-записи), 106
поле имени (RR-записи), 106
полные имена доменов (FQDN), 34
пользователи
уведомление об изменениях, 299, 300
порождаемые процессы, 189
порты, 356
порядковые номера, 123
nslookup, 444
данных зоны, 120
задание новых, 194
и динамические обновления, 307
прежние, 505, 507
сообщение, 225
увеличение, 192
порядок передачи, 628
потомки, 272
почта
обработка, 126
ретрансляция, 127
почтовые программы, 125
почтовые ретрансляторы, 126
A-записи, 132
доменные имена, 126
каноническое доменное имя, 132
пошаговая передача зоны (IXFR)
поддержка в BIND 8/9, 63
предпочтительные значения, 127
преобразование
h2n, программа, 108
префикс формата, 356
приложения, шлюзы, 392
примеры программ
получение, 16
примитивные анализаторы, 49
принципы создания поддоменов, 273
приоритеты, перечень, 206
пробел в качестве имени записи
ресурсов, 102
пробелы в конце строки
и DNS-клиенты BIND, 137
программирование
на языке командного
интерпретатора, 539
с использованием библиотеки DNS-клиента, 538
на языке C, 545
на языке Perl, 573
программное обеспечение
брандмауэры, 389
выбор узла, 242
комплектация Unix-систем
сетевыми программами, 62
распространение BIND, 61
собранные пакеты BIND, 65
программы
получение примеров, 16
производительность
DNSSEC, 428
и MX-записи, 583
и TTL, 60
наблюдение за DNS-серверами, 218
настройка системы, 342
обход фальшивых DNS-серверов, 340
передача зон и повышение
эффективности, 346
почтовых ретрансляторов, 130
разрешение проблем, 534
разрыв соединения, 515, 517
распределение нагрузки, round robin, 328
резервирование мощностей, 243
уровни отладки, 473
число запросов в секунду, 239
пространство имен
ip6.int, 361
доменных, 32
основные принципы разбиения, 274
расщепленное и затененное, 407
процессор, нагрузка, 23, 244
прямое отображение
A6-записи, 362
адресов IPv6, 360
псевдонимы, 89, 577
в MX-записях, 132
в записях ресурсов, 579
и DNS-клиенты, 555
использование, 577
множественные, 579

псевдонимы
определение, 581
переход к поддоменам, 299
родительской зоны, 300
создание, 156
таблицы узлов, 90
удаление, 300
пункты назначения почтовых сообщений, 126
пустые
ключи, 426
метки, 33
строки в файлах данных зоны, 87

Р

разбор ответных сообщений DNS, 561
разделы
авторитативности (сообщения DNS), 456, 546
и достоверность, 503
вопроса (сообщения DNS), 455, 546
объекты Perl, 574
заголовка (сообщения DNS), 455, 545
объекты Perl, 574
ответа (сообщения DNS), 455, 546
и достоверность, 503
размеры
сегмента данных, изменение, 347
стека, изменение, 348
файла образа, изменение, 348
разрешающие DNS-серверы, 384
разрешение, 49, 58
диаграмма процесса, 54
и корневые DNS-серверы, 50
и перебои связи, 267
ретрансляторы, 321
типы, 51
ускорение посредством кэширования, 58
разрешение проблем, 461, 494
BIND 8/9, 524
dig, инструмент, 468
NIS, 495
nslookup, 461
TSIG, ошибки, 530
инструменты и методы, 496
испорченный кэш, 536
потенциальные проблемы (перечень), 504
разрешение проблем
предотвращение и борьба с бедствиями, 263
проблемы взаимодействия версий, 525
проверка делегирования, 293
симптомы, 531
распаковка доменных имен, 560
распределение нагрузки, round robin, 90, 328
расщепленное пространство имен, 407
регистр
APNIC, 80
ARIN, 80
регистрация зон, 81
сетевые, 80
символов
файлы данных зоны, 86
чувствительность, 611
регистраторы, 67
выбор, 78
регистрация, 67
DNS-серверов, 255
запросов, 503
зон, 81
зоны in-addr.arpa, 82
стоимость, 74, 81
резервирование мощностей, 243
резервное питание, 266
резервные DNS-серверы, 143, 151
и распределение нагрузки round robin, 330
сообщение, 225
резервные копии
данных зоны, 119
передача зоны, 222
рекурсивные запросы, 52
и ретрансляторы, 322
получение ответов от DNS-серверов с разделяемой функциональностью, 384
ресурсы, ограничение, 347
ретранслирующие (специальные) DNS-серверы, 323
ретрансляторы, 321
брандмауэры, 394
отсутствие DNS-сервера, 324
ретрансляция обновлений, 305
решетка (#) , комментарии
файл resolv.conf, 149
родители и их жизнь, 301

родительские
домены, 43
регистрация, 80
связь с администрацией, 75
см. также домены, 82
зоны, установка контакта с администраторами, 257
родовые домены высшего уровня (gTLDs), 40, 67

С

сборка BIND, 632
связующие записи, 284
запрет поиска, 383
связь
перебои, 269
с родительской зоной, 272
сеанс управления, команда завершения, 185
сетевой информационный центр, 22
сетевые
маски, 378
регистры, 80
сети, 515
/24, формирование подсетей, 289, 293
ISDN, 594, 620
X.25, 594
класса A и B
формирование подсетей, 289
класса C
формирование подсетей, 289
отправка сообщений электронной почты через определенные сети, 406
подсети, 645, 661
сортировка адресов, 336
проверка регистрации, 78
указание в инструкции sortlist, 146
формирование подсетей, границы октетов, 288
сигналы, 180
и применение управляющих каналов, 181
применение, 188
символ табуляции в качестве имени записи ресурсов, 102
символьные строки
данные записей ресурсов, 629
синтаксис
инструкции domain, 137
инструкции search, 141
настройки в различных версиях BIND, 63
синтаксические ошибки, 110, 511, 513
в файле resolv.conf, 521
и ответы SERVFAIL, 235
системные
администраторы, 15
вызовы
getrlimit(), 219
sendto(), 237
setrlimit(), 219
запросы, 236
файлы, перемещение в BIND 8 и 9, 205
системы, настройка, 342
сквозная маршрутизация, записи, 595
служба whois
регистрация, 81
сетевая регистрация, проверка, 80
слэш (/), корень файловой системы Unix, 24, 33
советы, материнство, 273
совместимость
DNS-клиентов и DNS-серверов, 353
соединение, 508
разрыв, 515, 517
создание
видов, 326
поддоменов, 276
файла настройки BIND, 98
сокеты, дескрипторы, 480
сокращения в файле данных зоны, 101
примеры, 103
сообщения, 110
ведение log-файла (BIND 8/9), 206
об ошибках
DNS-клиенты, 143
неопределенные ошибки (nslookup), 466
ошибки синтаксиса, 110
первичного DNS-мастер-сервера
поиск ошибок в журнале syslog, 110, 119
объекты Perl, 574
предупреждающие об ошибках в именах узлов, 107
просмотр, 216
стартовые, 110

сообщения
упаковка
данные записей ресурсов, 629
управление форматом (BIND 8/9), 213
формат, 455, 545, 623
электронной почты
из-за брандмауэров, 404
отправка через определенные сети, 406
специальные кэширующие DNS-серверы, 252
и регистрация, 258
преимущества, 253
списки
отбора адресов, 303
allow-update, предписание, 309
поиска, 138
nslookup, 444
запрет на использование, 449
управления доступом (ACL), 303
динамические обновления, 309
справка по доступным командам, 183
статистика, 220
анализ данных перегруженного DNS-сервера, 247
наблюдение, 239
просмотр, 245
расшифровка значений, 227
стоимость регистрации поддоменов, 74
страны
суффиксы доменных имен, 41, 67
суффикс по умолчанию, добавление к доменным именам, 101
счетчик, нацеленный на увеличение порядковых номеров, 193
счетчики, таблица, 234
США
производные доменные имена, 73
суффиксы доменных имен, 41

Т

таблица счетчиков, 234
таблицы узлов, 23
вместо DNS, 31
и пункты назначения почтовых сообщений, 125
использование h2n для преобразования, 108
тасуемые адресные записи, 328
типы записей, поиск
nslookup, 449
топология и выбор DNS-серверов, 338
точка (.)
ndots, настройка, 661
в конце имени, 88
необходимость наличия, 102
завершающая доменные имена, 513, , 610
разрешение проблем, 513
указатель абсолютного имени домена, 34
и локальные доменные имена, 137
корневой узел, 24
отсутствие в локальном доменном имени, 137
последняя, 138
разделитель в именах доменов, 25
установка значения ndots, 147
точка с запятой (;)
начало комментария, 87, 98, 611, 661
транзакционные подписи, 368
трафик, 245
SRI-NIC, узел, 23
и корневые DNS-серверы, 50

У

увеличение порядкового номера, 505
удаление
CNAME-записей, 300
псевдонимов, 300
узлов, 192
удаленные имена
проблемы поиска, 532
удостоверенные (authenticated) DNS-серверы, 593
узел-бастион, 409
узлы, 33
RP-записи, 196
виды, 326
входящие в несколько сетей, 89
выбор, 242
добавление и удаление, 191, 510
доменные имена, 27
и загрузка процессора, 244
интервал сканирования сетевых интерфейсов, 351
информационные записи, 613
конфликты имен, 23

узлы
локальное доменное имя, установка, 112
многосетевые, 332
настройка, 135
настройка DNS-клиентов в различных системах, 157
неоднозначность имен, 155
определение псевдонимов, 581
последствия настройки, 151
сети, 34
статистика, 233
указание местоположения в TXT-записях, 196
умолчания
TTL, 260, 263
доменное имя узла, 112
доменное имя, настройка, 448
изменение суффикса для файла данных зоны, 204
приоритет, 207
список поиска, 138
упаковка доменных имен, 546
управляющие
каналы вместо сигналов, 181
операторы и ndc, 181
сообщения
и rndc, 185
изменение уровня отладки, 478
настройка DNS-серверов на прием, 182
усечение пакетов, 447
условия использования источников информации, создание, 162
успешный поиск (пример отладки), 482
установка
BIND, 632
вторичных DNS-серверов, 116
устаревание кэшированных данных, 97
и изменение значения TTL, 259
и распределение нагрузки round robin, 329

Ф

файловые
дескрипторы, 219, 480
каналы, 211
системы, в сравнении с DNS, 24, 32
домены, 34
файлы
db.ADDR, 85
в которых должна присутствовать SOA-запись, 88
db.cache, 95
db.DOMAIN, 85
в которых должна присутствовать SOA-запись, 88
/etc/hosts
и программа h2n, 108
/etc/named.boot, 86, 100
/etc/named.conf, 86, 100
авторизации, обновление, 155
данных (или базы данных), 48
корневых указателей, 95
обновление, 97, 199
ограничение числа открытых, 219
изменение, 348
файлы данных зоны, 48, 86
h2n, инструмент для создания, 108
генерация на основе таблиц узлов, 197
добавление данных, 195
и динамические обновления, 307
изменение суффикса по умолчанию, 204
обновление, 190
организация, 200
ошибки синтаксиса, 511
пример включения адресов DNS-серверов, 284
содержимое (примеры), 93
создание, 85
пример, 278
сокращения, 101
примеры, 103
указание расположения в файле настройки BIND, 99
формат, 609
Фер, Майкл, 373
Net::DNS, модуль, 573
фирменные записи WINS, 525
формат сообщений, 623
форматирование данных для каналов (BIND 8/9), 213

Х

хозяйственные работы, 350

Ц

целочисленные порядковые номера
против действительных чисел, 193
цепь доверия, 424

Ч

частично вторичные DNS-серверы, 254
и регистрация, 258
преимущества, 255

Ш

шифрование с открытым ключом, 415
шлюзы, 392

Э

электронная почта
DNS и, 153
адреса администраторов, 88
и резервирование мощностей, 243
электроэнергия, перебои, 263

По договору между издательством «Символ-Плюс» и Интернет-магазином «Books.Ru – Книги России» единственный легальный способ получения данного файла с книгой ISBN 5-93286-035-9, название «DNS и BIND, 4-е издание» – покупка в Интернет-магазине «Books.Ru – Книги России». Если Вы получили данный файл каким-либо другим образом, Вы нарушили международное законодательство и законодательство Российской Федерации об охране авторского права. Вам необходимо удалить данный файл, а также сообщить издательству «Символ-Плюс» (piracy@symbol.ru), где именно Вы получили данный файл.