Александр Филимонов

Протоколы Интернета

- ЛВС и удаленный доступ
- Внутренняя и внешняя маршрутизация
- Управление компонентами сети
- Формирование и интеграция потокового трафика

МАСТЕР_СИСТЕМ

Александр Филимонов

Протоколы Интернета

Санкт-Петербург

«БХВ-Петербург»

2003

УДК 681.3.06
ББК 32.973.202
Ф53

Филимонов А. Ю.

Ф53 Протоколы Интернета. — СПб.: БХВ-Петербург, 2003. — 528 с.: ил.

ISBN 5-94157-247-6

В книге подробно рассматриваются основные принципы построения и особенности применения наиболее популярных протоколов, используемых в современных сетях Интернета, а также направления их дальнейшего развития. Большое внимание уделено протоколам маршрутизации, управления компонентами сети и транспортным протоколам глобальной сети. Освещаются перспективные протоколы и технологии, предназначенные для обеспечения информационной безопасности современных вычислительных сетей, формирования и интеграции в них голосового и видеотрафика. Значительная часть книги посвящена рассмотрению протоколов таких популярных интернет-служб, как e-mail, WWW и др. Полнота представленного материала, подкрепленного практическими примерами, обеспечивает глубокое понимание принципов организации информационно-вычислительных сетей и успешную реализацию их на практике.

Для администраторов сетей,
сотрудников IT-отделов предприятий и студентов

УДК 681.3.06
ББК 32.973.202

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Анна Кузьмина*
Редактор	*Юрий Рожко*
Компьютерная верстка	*Татьяны Олоновой*
Корректор	*Виктория Голуб*
Оформление серии	*Via Design*
Дизайн обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 22.11.02.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 42,57.
Тираж 4000 экз. Заказ №
"БХВ-Петербург", 198005, Санкт-Петербург, Измайловский пр., 29.

Гигиеническое заключение на продукцию, товар № 77.99.02.953.Д.001537.03.02
от 13.03.2002 г. выдано Департаментом ГСЭН Минздрава России.

Отпечатано с готовых диапозитивов
в Академической типографии "Наука" РАН
199034, Санкт-Петербург, 9 линия, 12.

ISBN 5-94157-247-6

© Филимонов А. Ю., 2002
© Оформление, издательство "БХВ-Петербург", 2002

Содержание

Глава 1. Обзор протоколов Интернета 1

Рождение TCP/IP. История создания Интернета 1
Иерархическая система информационного взаимодействия 3
Уровень приложения 8
Уровень представления 8
Уровень сессии 8
Транспортный уровень 8
Сетевой уровень 9
Канальный уровень 9
Физический уровень 10
Взаимодействие на сетевом уровне Интернета 10
Адресация в IP-сетях 12
Сетевой IP-адрес (классы, структура) 12
Специфические адреса Интернета 14
Взаимодействие на сетевом уровне Интернета 19
Структура дейтаграммы IP 20
Формат заголовка дейтаграммы IP 20
Фрагментация и дефрагментация IP-дейтаграмм 22
Обзор протоколов, применяемых в Интернете 24
Прикладные сервисы TCP/IP 26

Глава 2. Протоколы канального уровня 33

Протокол SLIP 34
Протокол PPP 36
Служебный протокол LCP 40
Сообщение Configure-Request 41
Сообщение Configure-Ack 41
Сообщение Configure-Nak 42
Сообщение Configure-Reject 42

Сообщения Terminate-Request и Terminate-Ack 43
Сообщение Code-Reject 43
Сообщение Protocol-Reject 43
Сообщения Echo-Reply и Echo-Request 44
Сообщение Discard-Request 44
Протоколы аутентификации PAP и CHAP 44
Многоканальный протокол PPP 45
Организация связки каналов MLP 46
Структура фрагментов MLP 47
Обнаружение потери фрагмента MLP 49
Протоколы локальных вычислительных сетей 50
Сети CSMA/CD (Ethernet) 50
Сети с маркерным доступом 53
Сети Token Ring 54
Сети FDDI 57
Протокол ARP 58
ARP cache 62
Proxy ARP 62
Утилита ARP MS Windows и способы ее использования 62

Глава 3. Протоколы управления и контроля сетевых компонентов 65

Классы и задачи служебных протоколов 65
Протоколы мониторинга состояния сети 65
Протоколы управления компонентами сети 66
Протокол ICMP 66
Структура сообщения ICMP 67
Структура заголовка сообщений ICMP 68
Сообщение Destination Unreachable 69
Сообщение Time Exceeded 70
Сообщение Parameter Problem 71
Сообщение Source Quench 72
Сообщение Redirect 73
Сообщения Echo Request/Reply 75
Атаки с использованием ICMP Echo Request/Reply 76
Способы защиты от атак с использованием ICMP Echo Request/Reply 78
Утилита PING 79
Утилита TRACEROUTE 80
Утилита PATHPING 81
Сообщения Timestamp Request/Reply 82
Сообщения Address Mask Request/Reply 83
Протоколы RARP и DHCP 84
Протокол RARP 84
Организация RARP-взаимодействия компонентов сети 85

RARP-сервер ... 85
Протокол DHCP ... 87
Механизм динамического назначения адресов ... 88
Реализация протокола DHCP ... 89
Протоколы динамического резервирования ... 92
Назначение протокола HSRP ... 92
Состояния события и таймеры маршрутизаторов HSRP ... 94
Формат и типы пакетов HSRP ... 95
Взаимодействие HSRP с протоколами ARP и ICMP ... 96
Протокол VRRP ... 97
Протокол SNMP ... 98
Принципы построения системы управления сетью ... 99
Глобальное дерево имен ... 101
Взаимодействие компонентов протокола SNMP ... 104
Протокол RPC ... 107
Принципы построения протокола ... 108
Структуры сообщений протокола ... 110
Протокол TELNET ... 111
Принципы построения ... 112
Сетевые виртуальные терминалы — NVT ... 113
Описание протокола ... 116
TELNET — универсальный клиент ... 117

Глава 4. Маршрутизация в сетях TCP/IP ... 123

Принцип "Hop-by-Hop" ... 123
Статическое и динамическое определение маршрута ... 124
Домены маршрутизации и автономные системы ... 129
Протоколы внешней и внутренней маршрутизации ... 130
Протоколы RIP и IGRP ... 131
Формирование таблицы маршрутизации в алгоритмах "Distance-Vector" ... 131
Возникновение циклических маршрутов ... 133
Процедуры Split Horizon и Poison Reverse ... 135
Управляемое обновление маршрутов (Triggered Update) ... 137
Особенности алгоритма RIP II ... 138
Таймеры ... 138
Сообщения протокола RIP ... 139
Маршрутизация для внеклассовых сетей ... 142
Установление подлинности источника маршрутной информации ... 142
Протокол IGRP ... 143
Примеры практического применения протоколов RIP II и IGRP ... 145

Протокол маршрутизации OSPF ... 146
Формирование маршрутных таблиц алгоритмами "Link-State" ... 149
Иерархия областей и роли маршрутизаторов OSPF ... 150
Представление информации о состоянии компонентов сети (LSA) ... 151
Построение и обслуживание топологической базы маршрутов протокола OSPF ... 153
Сообщения протокола OSPF ... 155
Пример практического применения алгоритма протокола OSPF ... 161
Протокол маршрутизации EIGRP ... 162
Особенности гибридных протоколов маршрутизации ... 163
Алгоритм DUAL ... 164
Обеспечение быстрой сходимости и совместимости протоколов EIGRP и IGRP ... 168
Отношения между маршрутизаторами. Назначение и типы сообщений протокола EIGRP ... 170
Пример практического применения алгоритма протокола EIGRP ... 172
Протокол маршрутизации EGP ... 173
Назначение и особенности протоколов внешней маршрутизации ... 175
Взаимодействие маршрутизаторов протокола EGP, назначение и типы сообщений ... 177
Ограничения и недостатки протокола EGP ... 184
Протокол маршрутизации BGP ... 184
Способы распространения маршрутной информации протокола BGP ... 185
Атрибуты маршрутов ... 187
Понятие и способы реализации политики маршрутизации ... 190
Процедура определения оптимального маршрута ... 191
Особенности протокола BGP ... 192
Типы и форматы сообщений протокола BGP ... 194
Пример практического использования алгоритма протокола BGP ... 198

Глава 5. Транспортные протоколы Интернета ... 201

Протокол UDP ... 202
Протокол TCP ... 205
Способы обеспечения надежного информационного обмена ... 205
Взаимодействие с приложениями ... 206
Мультиплексирование информационных потоков, механизм гнезд (sockets) ... 208
Установление соединения и передача данных ... 209

Обеспечение гарантированной доставки данных ... 210
Управление темпом информационного обмена ... 213
Формат сегмента протокола TCP ... 214
Особенности практической реализации протокола TCP ... 215
Протокол RTP ... 217
Принципы построения протокола RTP ... 218
Сообщения протокола RTP ... 224
Сообщения протокола RTCP ... 226
Сообщения Sender Report ... 227
Сообщения Receiver Report ... 229
Сообщения Source Description ... 230
Сообщения BYE ... 231

Глава 6. Протоколы группового взаимодействия в сети Интернет ... 233

Особенности взаимодействия с использованием Multicast-адресов ... 233
Диапазоны групповых адресов TCP/IP ... 235
Передача группового трафика на канальном уровне ... 238
Определение состава группы ... 239
Маршрутизация и доставка группового трафика ... 240
Протокол IGMP ... 241
Формат сообщения протокола IGMP ... 244
Особенности использования и обеспечение совместимости протокола IGMP версий v1 и v2 ... 246
Дальнейшее развитие протокола IGMP ... 248
Практическая реализация протокола IGMP в современных сетях ... 248
Протокол CGMP (Cisco Systems) ... 250
Прослушивание сообщений протокола IGMP коммутаторами ЛВС ... 251
Протоколы групповой маршрутизации ... 254
Режимы доставки группового трафика ... 254
Протокол DVMRP ... 258
Принцип построения маршрута протоколом DVMRP ... 258
Сообщения протокола DVMRP ... 261
Протокол PIM ... 262
Особенности использования PIM в режиме DENSE ... 263
Режим SPARSE протокола PIM ... 264

Глава 7. Интеграция разнородного трафика в сетях TCP/IP ... 267

Принципы построения мультимедийных компонентов сети Интернет ... 270
Терминальные устройства H.323 ... 272
Компоненты и функции терминальных устройств H.323 ... 273
Компоненты и функции шлюза H.323 ... 278
Компоненты и функции привратника H.323 ... 279

Протокол RSVP ... 281
Механизмы управления трафиком протокола RSVP ... 282
Формы и методы резервирования ресурсов ... 284
Стиль Wildcard-Filter (WF) ... 285
Стиль Fixed-Filter (FF) ... 286
Стиль Shared Explicit (SE) ... 287
Сообщения протокола RSVP ... 288
Сообщения Reservation-Request ... 288
Сообщения Path ... 288
Сообщения Error ... 288
Сообщения Confirmation ... 289
Сообщения Teardown ... 289
Формат сообщений протокола RSVP ... 289
Принципы построения и компоненты сетей VoIP ... 291
Компоненты и алгоритмы систем VoIP ... 292
Терминальные устройства VoIP ... 292
Принципы построения и компоненты шлюзов VoIP ... 295
Функции привратника VoIP ... 297
Пример практического применения технологии VoIP ... 297

Глава 8. Обеспечение информационной безопасности в сетях TCP/IP ... 299

Комплекс протоколов IPsec ... 300
Схемы защиты информации ... 300
Соединения безопасности IPsec ... 303
Состав комплекса протоколов IPsec ... 304
Протокол заголовка аутентификации AH ... 304
Структура заголовка аутентификации протокола AH ... 304
Размещение заголовка аутентификации ... 305
Вычисление контрольного значения заголовка — ICV ... 306
Протокол защиты полезной нагрузки ESP ... 307
Формат блока данных протокола ESP ... 308
Размещение блока данных протокола ESP в IP-пакете ... 309
Процедура обработки порядковых номеров пакетов ... 310
Структура систем обеспечения безопасности IPsec ... 311
Процедуры аутентификации ... 311
Базы данных систем безопасности ... 312
Применение технологии MPLS в сети Интернет ... 315
Основные понятия и компоненты MPLS ... 316
Основные принципы построения MPLS-систем ... 318
Структура метки MPLS ... 319
Стеки меток MPLS ... 320
Принципы построения виртуальных частных сетей ... 321
Типы виртуальных частных сетей ... 322

Технологии виртуальных частных сетей ... 324
Виртуальные сети на основе технологий IPsec ... 324
Виртуальные сети на основе технологии MPLS ... 326

Глава 9. Система доменов и распределенная база данных DNS ... 329

История создания DNS ... 331
Решения DNS ... 331
Состав и основные компоненты DNS ... 332
Пространство имен домена и записи базы данных ... 336
Спецификация ... 336
Записи базы данных ... 337
Текстовое представление данных ... 342
Имена и псевдонимы ... 344
Запросы. Стандартные и инверсные запросы ... 344
Серверы имен ... 348
Понятие зоны. Деление DNS-пространства на зоны ... 348
Механизмы и алгоритмы обслуживания запросов ... 350
Инсталляция и перемещение зон ... 353
Программы разрешения имен ... 354
Интерфейс разрешения имен ... 354
Функциональность ... 355
Пример построения и инсталляции DNS ... 359
Примеры построения запросов и ответов серверов имен ... 362
Примеры работы программы разрешения имен ... 366
Утилита NSLOOKUP ... 368

Глава 10. Протоколы электронной почты — SMTP, POP3, IMAP4 ... 371

Протокол SMTP ... 371
Формат почтовых сообщений MIME ... 372
Взаимодействие компонентов SMTP ... 380
Организация информационного обмена ... 381
Ретрансляция сообщений ... 386
Примеры выполнения транзакций протокола SMTP ... 388
Использование протокола SMTP для передачи 8-битных данных ... 390
Взаимодействие SMTP с протоколом TCP ... 392
Протокол POP3 ... 393
Основные принципы протокола POP3 ... 393
Организация информационного обмена ... 395
Пример использования протокола POP3 ... 399
Протокол IMAP4 ... 400
Основные принципы протокола IMAP4 ... 401
Атрибуты сообщений IMAP-системы ... 402
Основные команды протокола IMAP4 ... 403
Пример взаимодействия IMAP4-клиента с сервером ... 408

Глава 11. Протокол передачи новостей — NNTP 411

Система новостей UseNet 411
Способы тиражирования сообщений 412
Серверы новостей 413
Формат статей и сообщений системы UseNet 414
Управляющие сообщения 418
Модели передачи статей 420
Протокол NNTP, описание команд 422
Пример передачи статьи по протоколу NNTP 426

Глава 12. Протокол HTTP 429

Сервисы сети WWW 429
Принципы построения HTTP-соединения 432
Описание протокола HTTP 434
Организация запроса протокола HTTP 435
Структура ответа протокола HTTP 441
Механизмы аутентификации 443
Взаимодействие протокола HTTP и CGI-сервиса 445
Расширение возможностей протокола HTTP 448

Глава 13. Протоколы передачи файлов 451

Основные принципы протокола FTP 451
Описание команд протокола 455
Сценарий работы протокола FTP 457
Протоколы TFTP и SFTP 458
Утилита FTP 461

Глава 14. Сетевая файловая система — NFS 465

Принципы построения протокола 466
Файловая система 466
Механизм аутентификации 467
Описание протокола 468

Глава 15. Направления дальнейшего развития протокола IP 471

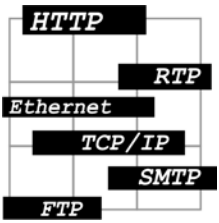
Предпосылки и направления модернизации TCP/IP 471
Технология вложенных заголовков 473
Основной заголовок протокола IPv6 475
Сетевые адреса протокола IPv6 478
Категории сетевых адресов 478
Формы представления сетевых адресов протокола IPv6 479
Специальные сетевые адреса протокола IPv6 480
Заголовки расширения протокола IPv6 481
Заголовок Hop-by-Hop Options 481

Заголовок маршрутизации Routing Header....483
Заголовок фрагментации Fragment Header....484
Заголовок Destination Options Header....485
Заголовки информационной безопасности протокола IPv6....486
Грандиозное переселение....486

Глоссарий....489

Предметный Указатель....507

Глава 1



Обзор протоколов Интернета

Рождение TCP/IP. История создания Интернета

В научном и техническом мире существует мнение, что чаще всего перспективные и передовые технологии рождаются при изобретении нового оружия или средств защиты от него. Это объясняется тем, что система, предназначенная для работы в экстремальных условиях, надежно и устойчиво функционирует в мирное время. Именно так и случилось с Интернетом и протоколами семейства TCP/IP (Transmission Control Protocol/Internet Protocol).

Идея создания Интернета была предложена RAND Corporation (США) в связи с необходимостью построения коммуникационной отказоустойчивой сети, которая могла бы функционировать даже в том случае, если большая часть ее узлов вышла из строя, например, в случае ядерной войны. Решение состояло в том, чтобы создать сеть, где информационные пакеты могли бы передаваться от одного узла к другому без какого-либо централизованного контроля. В случае, если бы основная часть сети не работала, пакеты самостоятельно передвигались бы по доступным узлам до тех пор, пока не попали бы в точку своего назначения. Кроме того, сеть должна была быть достаточно устойчива к возможным ошибкам при передаче пакетов, т. е. обладать механизмом контроля пакетов и обеспечивать контроль доставляемой информации. Хотя проект был по своей сути военным, к его разработке привлекались научные институты и университеты.

В начале 70-х годов созданием такой сети занялось Агентство Передовых Проектов Национальной Безопасности США (United States Defense Advanced Research Project Agency — DARPA). Точнее, история Интернета началась в 1969 году, когда в Калифорнийском университете был установлен первый узел сети, получивший название ARPAnet (по имени компании, финансировавшей проект), и были созданы основы для построения сети с коммутируемыми пакетами. К началу 1971 года ARPAnet насчитывала уже около 20 узлов, и более 30 университетов получили возможность подключения к сети в рамках проекта.

К середине 70-х были предприняты попытки объединения различных пакетных сетей. В начале 80-х к ARPAnet были подключены первые локальные

сети, и для использования в построенной сети (в дальнейшем именуемой Интернетом) был выбран, адаптирован и затем повсеместно принят для работы набор протоколов Transmission Control Protocol/Internet Protocol (TCP/IP). Он сменил более ранний протокол NCP (Network Communication Protocol). TCP/IP вполне удовлетворял всем тем требованиям, которые на него возлагались. По сути дела, это и послужило началом широкого распространения TCP/IP. Что из этого вышло — мы можем судить по той скорости, с которой внедряется в нашу жизнь Интернет.

Существует много причин, почему протоколы семейства TCP/IP были выбраны за основу в сети Интернет. Это, прежде всего, возможность работать с этими протоколами как в локальных LAN (Local Area Network), так и в глобальных WAN (Wide Area Network) сетях, способность протоколов управлять большим количеством стационарных и мобильных пользователей, удобство для использования частными лицами или организациями, желающими подсоединиться прямо к Интернету или через фирмы, предоставляющие этот сервис. Протоколы Интернета обеспечивают высокий уровень взаимодействия между совершенно различными операционными системами, предоставляют средства для разработки на их основе приложений, использующих современный программный интерфейс. Но главную роль, безусловно, сыграло провидение разработчиков ARPAnet, когда они выбирали этот почти никому не известный протокол для построения своей тогда еще крохотной сети.

В 1986 году на базе существующей опорной сети ARPAnet, которая обладала производительностью 56 Кбит/с и объединяла шесть суперкомпьютерных центров США, было начато строительство новой сети. Решение о модернизации было принято вследствие загруженности существующей инфраструктуры. Заказ на проведение проекта получил консорциум Merit, MCI и IBM. При реализации этого проекта самой сложной проблемой был поиск специалистов, знающих технологию сетей и, в частности, TCP/IP. Однако уже через 8 месяцев система на базе каналов T1 (1,54 Мбит/с) была сдана в эксплуатацию. Она состояла из 13 коммутационных узлов, каждый из которых составляли 9 параллельно работающих компьютеров IBM, работающих под управлением Berkeley UNIX и соединенных локальной сетью. Узлы занимались маршрутизацией пакетов и сбором сетевой информации. Причем каждый был построен таким образом, что при выходе его из строя выполняемые им функции принимали на себя оставшиеся узлы.

Дальнейшее развитие Интернета шло подобно снежному кому. Ядро обрастало все большим и большим количеством узлов, соединений и подключенных сетей. И уже в 1989 году загрузка опорных линий достигла предельных значений. Поэтому в 1991 году была выполнена модернизация и сеть была переведена на линии T3 (45 Мбит/с), причем коммутационные компьютеры были заменены на ЭВМ IBM RS/6000. К тому времени сеть уже насчитывала 16 узлов и более 3500 подключенных сетей.

В настоящее время скорости информационного обмена сети Интернет и темпы ее роста продолжают увеличиваться.

Иерархическая система информационного взаимодействия

Информационное взаимодействие в сети Интернет строится в соответствии с правилами и требованиями общего международного стандарта ISO 7498 (ISO — International Organization for Standardization).

Этот стандарт имеет тройной заголовок "Информационно-вычислительные системы — Взаимодействие открытых систем — Эталонная модель". Обычно его называют короче — "Эталонная модель взаимодействия открытых систем". Публикация этого стандарта в 1983 году подвела итог многолетней работы многих известных телекоммуникационных компаний и стандартизующих организаций.

Основной идеей, которая положена в основу этого документа, является разбиение процесса информационного взаимодействия между системами на уровни с четко разграниченными функциями.

Идея такого разбиения не была революционной. Можно вспомнить, что слоистую архитектуру имели информационные взаимодействия в сетях SNA (Systems Network Architecture) и DNA (Digital Network Architecture).

В качестве прообраза модели взаимодействия OSI (Open System Interconnection) была использована структура, предложенная ANSI (American National Standards Institute). Основные работы по созданию текста документа были выполнены CCITT (Consultative Committee for International Telegraphy), а итоговый документ появился в виде стандарта ISO. Статус стандарта ISO важен для данного документа, поскольку ISO 7498 является стандартом стандартов в области телекоммуникаций.

Преимущества слоистой организации взаимодействия заключаются в том, что она обеспечивает независимую разработку уровневых стандартов, модульность аппаратуры и программного обеспечения информационно-вычислительных систем и способствует тем самым техническому прогрессу в данной области.

При использовании многоуровневой модели проблема перемещения информации между узлами сети разбивается на более мелкие и, следовательно, более легко разрешимые проблемы.

Многоуровневая модель четко описывает, каким образом информация проделывает путь через среду сети от одной прикладной программы, к примеру, обработки таблиц, до иной прикладной программы обработки тех же таблиц, находящейся на другом компьютере сети.

Предположим, например, что система А, изображенная на рис. 1.1, имеет информацию для отправки в систему В. Прикладная программа системы А начинает взаимодействовать с уровнем 4 системы А (верхний уровень), который, в свою очередь, начинает взаимодействовать с уровнем 3 системы

А, и т. д. — до уровня 1 системы А. Задача уровня 1 — отдавать, а потом забирать информацию из физической среды сети.

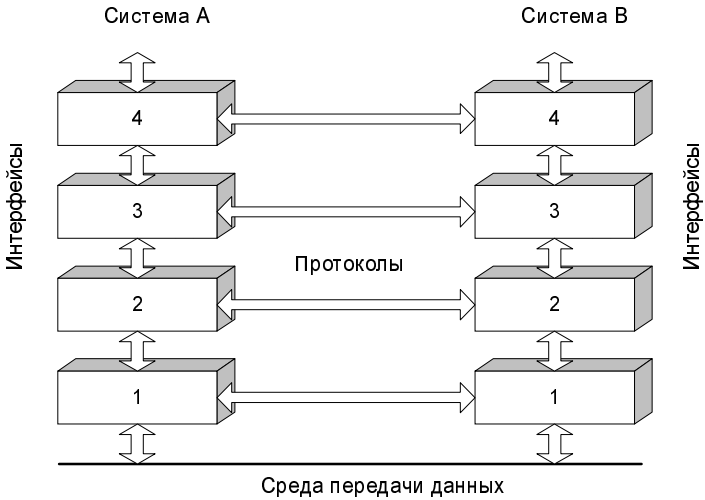


Рис. 1.1. Организация информационного взаимодействия двух систем

Примечание

Поскольку информация, которая должна быть отослана, проходит вниз через уровни системы, по мере этого продвижения она становится все меньше похожей на человеческий язык и все больше похожей на ту информацию, которую понимают компьютеры, а именно "единицы" и "нули".

После того как информация проходит через физическую среду сети и поступает в систему В, она последовательно обрабатывается на каждом уровне системы В в обратном порядке — сначала на уровне 1, затем на уровне 2 и т. д., пока, наконец, не достигнет прикладной программы системы В.

Многоуровневая модель не предполагает наличия непосредственной связи между одноименными уровнями взаимодействующих систем. Следовательно, каждый уровень системы А должен полагаться на услуги, предоставляемые ему смежными уровнями системы А, чтобы помочь осуществить связь с соответствующим уровнем системы В. Предположим, что уровень 4 системы А должен связаться с уровнем 4 системы В. Для того чтобы выполнить эту задачу, уровень 4 системы А должен воспользоваться услугами уровня 3 системы А, тогда уровень 4 будет называться "пользователем услуг", а уровень 3 — "источником услуг".

Информация по оказываемым услугам передается между уровнями в специальном информационном блоке, который называется *заголовком*. Заголовок обычно предшествует передаваемой прикладной информации.

Предположим, что система А хочет отправить в систему В какой-либо текст, называемый "данные" или "информация". Этот текст передается из прикладной программы системы А в верхний уровень этой системы. Прикладной уровень системы А должен передать определенную информацию в прикладной уровень системы В, поэтому он помещает управляющую информацию своего уровня в виде заголовка перед фактическим текстом, который должен быть передан. Построенный таким образом информационный блок передается в уровень 3 системы А, который может предварить его своей собственной управляющей информацией, и т. д.

Размеры сообщения увеличиваются по мере того, как оно проходит вниз через уровни до тех пор, пока не достигнет сети, где оригинальный текст и вся связанная с ним управляющая информация перемещаются в систему В и поглощаются уровнем 1 системы В. Уровень 1 системы В отделяет от поступившей информации и обрабатывает заголовок уровня 1, после чего он определяет, как обрабатывать поступивший информационный блок. Слегка уменьшенный в размерах информационный блок передается в уровень 2, который отделяет заголовок этого же уровня, анализирует его, чтобы узнать о действиях, которые он должен выполнить и т. д. Когда информационный блок наконец доходит до прикладной программы системы В, он должен содержать только оригинальный текст.

Структура заголовка и собственно данных относительна и зависит от уровня, который в данный момент анализирует информационный блок. Например, на уровне 2 информационный блок состоит из заголовка этого же уровня и следующих за ним данных. Однако данные уровня 2 могут содержать заголовки уровней 3 и 4. Кроме того, заголовок уровня 2 является просто данными для уровня 1. Помимо заголовка на каждом уровне системы информационный блок завершается соответствующей контрольной суммой КонтСум. Эта концепция иллюстрируется на рис. 1.2.

Данная модель напоминает собой вложенные друг в друга матрешки. Самая маленькая из них — это и есть пользовательские данные, а все остальные служат для доставки данных в точку назначения.

Иными словами, в результате работы этого механизма каждый пакет более высокого уровня вкладывается в "конверт" протокола нижнего уровня. Здесь уместно провести аналогию с обычными почтовыми отправлениями. Так, например, если вы пишете обычное письмо и вкладываете его в конверт с адресом, то текст письма будет информационным сообщением, которое вы хотите отправить, а конверт — заголовком "почтового" протокола. На почте ваше письмо перекладывают в мешок (протокол более низкого уровня) с письмами того же или близкого назначения и т. п. Электронные протоколы работают по той же схеме, только доставку и целостность обычных писем обеспечивает добросовестность служащих отделений связи, а электронным протоколам приходится следить за этим самостоятельно.

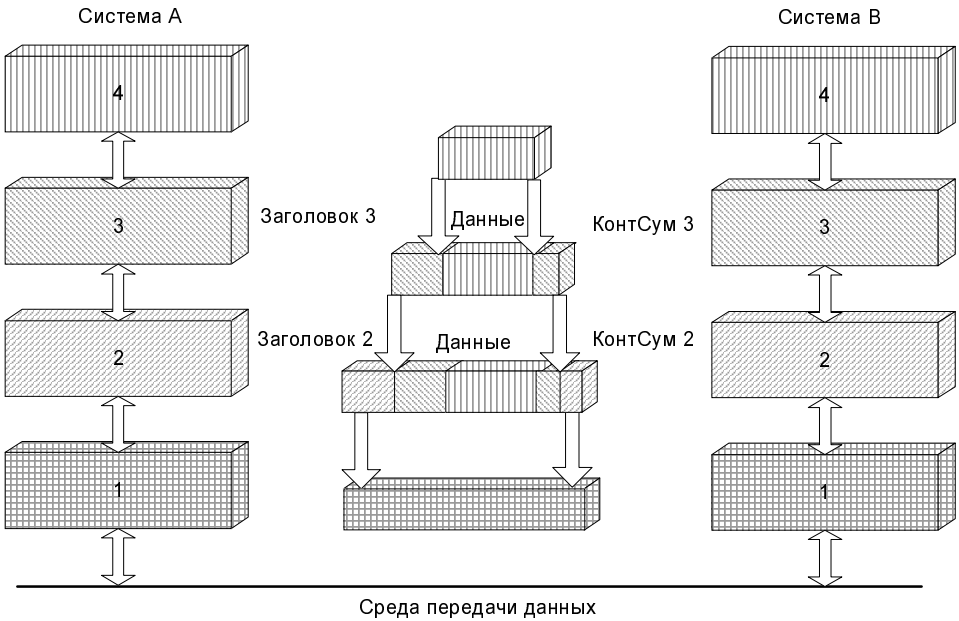


Рис. 1.2. Инкапсуляция блоков данных различных уровней

В соответствии с ISO 7498 выделяются семь уровней (слоев) информационного взаимодействия:

7. Уровень приложения (Application Layer)
6. Уровень представления (Presentation Layer)
5. Уровень сессии (Session Layer)
4. Транспортный уровень (Transport Layer)
3. Сетевой уровень (Network Layer)
2. Канальный уровень (DataLink Layer)
1. Физический уровень (Physical Layer)

Информационное взаимодействие двух или более систем, таким образом, представляет собой совокупность информационных взаимодействий уровневых подсистем, причем каждый слой локальной информационной системы взаимодействует только с соответствующим слоем удаленной системы.

Определение

Протоколом мы будем называть набор алгоритмов (правил) взаимодействия объектов одноименных уровней.

Слои (уровни) одной информационной системы также взаимодействуют друг с другом, причем в непосредственном взаимодействии участвуют только соседние уровни. Как правило, средний уровень пользуется услугами, которые ему предоставляет нижний уровень, а сам, в свою очередь, предоставляет услуги для верхнего уровня.

Определение

Интерфейсом мы будем называть совокупность правил, в соответствии с которыми осуществляется взаимодействие с объектом данного уровня.

Иерархическая организация сетевого взаимодействия позволяет обеспечивать преемственность разработанных структур и их быструю адаптацию к изменениям, происходящим в технологиях передачи данных. Например, при переходе на новый способ передачи данных по физическому носителю, изменения коснутся только нижних уровней и совсем не затронут верхние в том случае, если система протоколов организована в соответствии с требованиями ISO 7498. На практике требования данного стандарта реализуются в виде стека протоколов.

Определение

Стеком мы будем называть иерархически организованную группу взаимодействующих протоколов

Протоколы, которые входят в стек, имеют специализированный интерфейс и предназначены для взаимодействия только с протоколами соответствующих уровней данного стека. В качестве примеров таких стеков можно привести стек TCP/IP и протоколы X.25.

Уровни 7—5 считаются верхними и, как правило, не отражают специфики конкретной сети. Блок данных пользователя (сообщение) этими уровнями рассматривается как единое целое. Изменения могут испытывать только сами данные.

Уровни 1—3 и иногда 4 считаются нижними уровнями OSI. На каждом из этих уровней определяется свой формат представления данных. При прохождении по стеку с 4-го уровня до первого сообщение пользователя последовательно фрагментируется и преобразуется в последовательность блоков данных соответствующего уровня.

Определение

Процесс помещения фрагментированных блоков данных одного уровня в блоки данных другого уровня называют *инкапсуляцией*.

Обычно инкапсулируются данные протоколов верхних уровней в блоки данных протоколов нижних уровней (сетевой — канальный), но также может выполняться инкапсуляция для протоколов одноименных уровней (IP-X.25).

Уровень приложения

Протоколы, оперирующие на седьмом уровне OSI, предназначены для обеспечения доступа к ресурсам сети программ-приложений пользователя. На данном уровне определяется интерфейс с коммуникационной частью приложения. В качестве примера протоколов прикладного уровня можно привести протокол TELNET, который обеспечивает доступ пользователя к узлу сети в режиме удаленного терминала.

Уровень представления

На этом уровне обычно выполняется преобразование форматов данных, алгоритмы шифрования и компрессии данных.

Уровень сессии

На данном уровне устанавливаются, обслуживаются и разрываются сессии между представительными объектами приложений. В качестве примера протокола сеансового уровня можно рассмотреть протокол RPC (Remote Procedure Call). Как следует из названия, данный протокол предназначен для отображения результатов выполнения процедуры на удаленном узле. В процессе выполнения этой процедуры между приложениями устанавливается сеансовое соединение. Назначением этого соединения является обслуживание запросов, возникающих при взаимодействии приложения-клиент с приложением-сервер. Протоколы сеансового уровня непосредственно взаимодействуют с протоколами транспортного уровня. Организация сеансового соединения может вызвать установление транспортного соединения. Возможен, однако, вариант, когда несколько сеансовых соединений используют одно транспортное.

Транспортный уровень

Можно выделить два типа протоколов транспортного уровня — сегментирующие протоколы и дейтаграммные протоколы.

Сегментирующие протоколы транспортного уровня разбивают исходное сообщение на блоки данных транспортного уровня — сегменты. Основной функцией таких протоколов является обеспечение гарантированной доставки этих сегментов до объекта назначения и восстановление исходного сообщения. Для выполнения этой задачи протокол транспортного уровня устанавливает и разрывает соединение между узлами, управляет скоростью передачи сегментов и обеспечивает достоверность передачи и восстановления сообщения специальными контрольными процедурами. Функционирование протокола транспортного уровня состоит из трех фаз: фазы установления соединения, фазы передачи сегментов и фазы разрыва соединения.

Дейтаграммные протоколы не сегментируют сообщение и отправляют его одним куском, который называется *дейтаграмма*. При этом функции установления и разрыва соединения, управления потоком не нужны. Дейтаграммные протоколы просты для реализации, однако не обеспечивают достоверной доставки сообщений.

Поскольку транспортное соединение может быть использовано несколькими приложениями, для идентификации транспортных блоков этих приложений используются номера портов получателя и источника данных. Таким образом, сетевое взаимодействие на транспортном уровне осуществляется между портами — интерфейсами данного уровня.

Сетевой уровень

Задачей протоколов сетевого уровня является построение сетевого адреса и определение маршрута доставки информационных блоков протоколов верхних уровней (маршрутизация).

Для того чтобы блок данных был доставлен до какого — либо узла, этому узлу должен быть поставлен в соответствие известный передатчику сетевой адрес. Группы узлов, объединенные по территориальному принципу, образуют сеть. Для упрощения решения задачи маршрутизации сетевой адрес узла составляется из двух частей -адреса сети и адреса узла. Таким образом, задача маршрутизации распадается на две подзадачи — поиск сети и поиск узла в этой сети.

Например, сетевой адрес у протокола IP состоит из четырех байтов. Адрес сети может занимать в нем от 1 до 3 байтов. Маршрут может быть определен двумя способами — статическим способом и динамическим способом. Использование статической маршрутизации целесообразно в тех случаях, когда сеть имеет простую и стабильную архитектуру. Примером такой сети может быть локальная частная сеть небольшой организации, которая пользуется услугами только одного провайдера Интернета. Динамический способ является более предпочтительным для использования в сетях крупных организаций, имеющих в своем составе множество региональных офисов и нескольких провайдеров. Использование динамической маршрутизации в данном случае позволяет учитывать изменения, происходящие в сети. Динамический способ реализуется при использовании алгоритмов маршрутизации. Современные алгоритмы маршрутизации обеспечивают не только быстрое переключение маршрута в обход неисправного участка, но динамический анализ качества используемого канала передачи данных и перераспределение загрузки между параллельными каналами.

Канальный уровень

Назначением протоколов канального уровня является обеспечение передачи данных по физическому носителю — среде передачи. На канальном уровне данные передаются в виде блоков, которые называются кадрами. Тип используемой среды передачи и ее топология во многом определяют вид кадра

протокола транспортного уровня, который должен быть использован. При использовании топологий "общая шина" и "point-to-multipoint" средствами протокола канального уровня должны быть определены физические адреса, с помощью которых будет производиться обмен данными по разделяемой среде передачи и процедура доступа к этой среде. Примерами таких протоколов являются протоколы Ethernet (в соответствующей части) и HDLC (High-level Data Link Control). Протоколы транспортного уровня, которые предназначены для работы в среде типа "точка-точка", не определяют физических адресов и имеют упрощенную процедуру доступа. Примером протокола такого типа является протокол PPP (Point-to-Point Protocol).

Физический уровень

Протоколы физического уровня обеспечивают непосредственный доступ к среде передачи данных для протоколов канального и последующих уровней. Данные передаются протоколами данного уровня в виде битов (для последовательных протоколов) или групп бит (для параллельных протоколов). На данном уровне определяются набор сигналов, которыми обмениваются системы, параметры этих сигналов — временные и электрические, и последовательность формирования этих сигналов при выполнении процедуры передачи данных. Кроме того, на данном уровне формулируются требования к электрическим, физическим и механическим характеристикам среды передачи. Наиболее распространенным протоколом транспортного уровня до недавних пор был V.24, который, в частности, обеспечивал интерфейс последовательного обмена IBM PC.

Взаимодействие на сетевом уровне Интернета

Если быть наиболее точным, то данному набору протоколов больше подходит название "Комплекс протоколов Интернета". Этот комплекс охватывает целое семейство протоколов, прикладные программы и даже саму сеть. В его состав входят протоколы UDP (User Datagram Protocol), ARP (Address Resolution Protocol), ICMP (Internet Control Message Protocol), TELNET, FTP (File Transfer Protocol) и многие другие. Но, поскольку TCP и IP — наиболее известные протоколы этого комплекса, часто, ссылаясь на данный набор протоколов, используют термин TCP/IP. По сути, TCP/IP — это технология межсетевого взаимодействия.

TCP/IP был разработан совместно членами объединения, использующего процесс экспертной оценки документации, называемый Request for Comments (RFC). Несколько человек предложили структурировать и опубликовать описание RFC. Оно было изучено другими членами команды, в него были внесены изменения, и затем эти изменения отразились на структуре RFC.

Часть изменений были сделаны позднее. Со временем RFC превратился в развивающийся набор стандартов и стал применяться в построении продуктов, подчиняющихся одному или нескольким стандартам RFC. Этот процесс стал достаточно эффективно развиваться и со временем (первые RFC опубликованы в 1969 году) имеющиеся в протоколах ошибки были устранены.

Существующие RFC доступны для массового использования. В основном они предназначены для тех разработчиков и организаций, кто проектирует продукты и сервисы для использования их в сети Интернет, а также для тех, кто хочет лучше понимать идеи и принципы, заложенные в основу современного киберпространства.

Как бы мы не называли TCP/IP — это семейство протоколов. Часть из них обеспечивает выполнение "низкоуровневых" сетевых функций для множества приложений, таких, как работа с аппаратными протоколами, поддержка механизма доставки пакета по адресу назначения через множество сетей и узлов, обеспечение достоверности и надежности соединения и др. Другая часть протоколов предназначена для выполнения прикладных задач, таких, как передача файлов между компьютерами, отправка электронной почты или чтение гипертекстовой страницы WWW-сервера.

Рассмотрим, например, процесс отправки почты. Прежде всего, здесь используется протокол работы с почтой. Он определяет систему команд, которые один компьютер посылает другому. Эти команды обозначают, кто будет отправителем сообщения, кто получателем, в какой форме будет отправлено само сообщение. Этот протокол, как и другие прикладные протоколы, подразумевает, что между компьютерами уже установлено надежное соединение. Именно за установление и обеспечение этого соединения и отвечают "низкоуровневые" протоколы TCP и IP.

Задачей TCP является доставка всей информации компьютеру получателя, контроль последовательности передаваемой информации, повторная отправка недоставленных пакетов в случае сбоев работы сети. Кроме того, если сообщение достаточно большое, чтобы отправить его в одном пакете, TCP делит и отправляет его несколькими блоками. TCP также осуществляет контроль за составлением первоначального сообщения из этих блоков на компьютере получателя. Поскольку эти функции требуется выполнять для многих приложений, их объединили и выделили в единый протокол, который мог бы использоваться всеми прикладными протоколами. TCP можно рассматривать как библиотеку процедур, которую используют прикладные протоколы, если необходимо установить надежное и достоверное соединение между компьютерами.

Подобно тому как почтовый протокол использует TCP, сам TCP использует протокол IP, который обеспечивает доставку пакета по адресу, т. е. адресацию и маршрутизацию. Функции, которые предоставляет TCP, необходимы

для работы множества приложений, однако существуют приложения, для работы которых эти функции не требуются. Эти приложения используют вместо TCP свой протокол, обеспечивающий взаимодействие приложений, например UDP, которому для работы также необходим механизм, который бы осуществлял доставку пакета по адресу (т. е. уровня IP).

Адресация в IP-сетях

IP-адресация компьютеров в сети Интернет построена на концепции сети, состоящей из узлов и других сетей. Узел представляет собой объект сети, который может передавать и принимать IP-пакеты, например, компьютер рабочей станции или маршрутизатор. Обычно ошибочно понимают под узлом какой-либо сервер, однако в рамках концепции IP-сети и рабочие станции, и серверы — все являются узлами.

Узлы соединены друг с другом через одну или несколько сетей. IP-адрес любого из узлов состоит из адреса сети и адреса узла в этой сети. IP-адресация, в отличие, например, от IPX-адресации (IPX — Internetwork Packet Exchange), использует один идентификатор, объединяющий и адрес сети, и адрес узла. В соответствии с соглашением, адрес представляется четырьмя десятичными числами, разделенными точками. Каждое из этих чисел не может превышать 255 и представляет один байт 4-байтного IP-адреса.

Длина адреса IP составляет 32 бита, разделенных на две или три части. Первая часть обозначает адрес сети, вторая (если она имеется) — адрес подсети, и третья — адрес главной вычислительной машины. Адреса подсети присутствуют только в том случае, если администратор сети принял решение о разделении сети на подсети. Длина полей адреса сети, подсети и главной вычислительной машины являются переменными величинами.

Менеджер сети (администратор или программа) присваивает IP-адреса машинам в соответствии с тем, к каким IP-сетям они подключены. Старшие биты 4-байтного IP-адреса определяют номер IP-сети. Оставшаяся часть IP-адреса — номер узла. Напомним, что IP-адрес узла идентифицирует точку доступа модуля IP к сетевому интерфейсу, а не всю машину.

Сетевой IP-адрес (классы, структура)

Различают пять классов сетевых адресов Интернета. Все классы имеют различные структуры сетевого адреса. Крайние левые биты сетевого адреса Интернета обозначают класс сети. Перечислим эти пять классов сетевых адресов, имеющихся в сетях IP.

- **Класс A.** Сети класса A предназначены главным образом для использования крупными организациями, так как они обеспечивают всего 7 бит для поля адреса сети.

- **Класс B.** Сети класса B выделяют 14 бит для поля адреса сети и 16 бит для поля адреса главной вычислительной машины. Этот класс адресов обеспечивает хороший компромисс между адресным пространством сети и главной вычислительной машиной.
- **Класс C.** Сети класса C выделяют 22 бита для поля адреса сети. Однако сети класса C обеспечивают только 8 бит для поля адреса главной вычислительной машины, поэтому число главных вычислительных машин, приходящихся на сеть, может стать ограничивающим фактором.
- **Класс D.** Адреса класса D резервируются для групповой адресации в соответствии с официальным документом RFC-1112. В адресах класса D четыре бита наивысшего порядка устанавливаются на значения 1, 1, 1 и 0.
- **Класс E.** Адреса класса E также определены IP, но зарезервированы для использования в будущем. В адресах класса E все четыре бита наивысшего порядка устанавливаются в 1.

На рис. 1.3 изображена структура адресов сетей классов A—E.

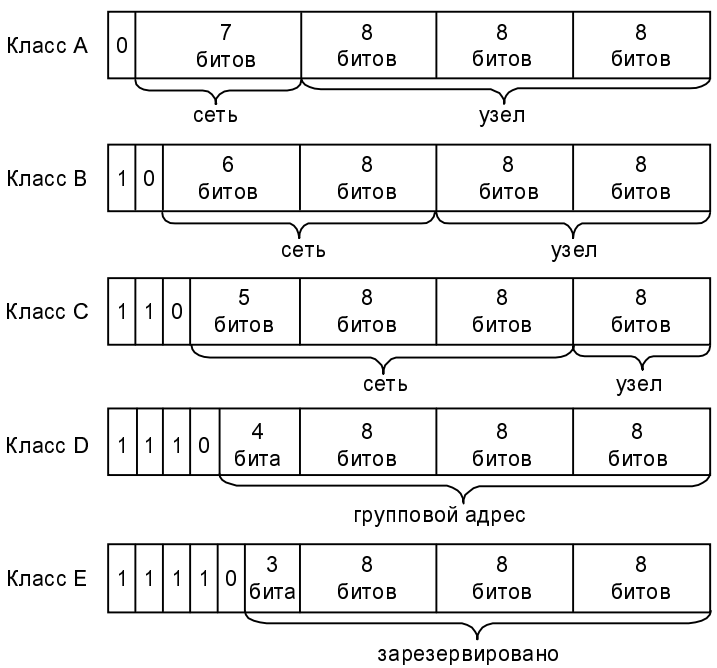


Рис. 1.3. Структура адресов сетей классов A–E

В табл. 1.1 приведены диапазоны адресов сетей и узлов сетей классов A, B, C.

Таблица 1.1. *Диапазоны адресов сетей и узлов сетей классов A, B, C*

Класс	Диапазон номера сети	Диапазон адресов узла
A	1—126	0.0.1—255.255.254
B	128.0—191.255	0.1—255.254
C	192.0.0—223.255.255	1—254

Адреса класса A предназначены для использования в больших сетях общего пользования. Они допускают большое количество номеров узлов. Адреса класса B используются в сетях среднего размера, например, сетях университетов и крупных компаний. Адреса класса C используются в сетях с небольшим числом компьютеров. Адреса класса D используются при обращениях к группам машин, а адреса класса E зарезервированы на будущее.

Специфические адреса Интернета

Для написания адреса IP обычно используется так называемая десятичная — точечная (dotted — decimal) нотация. При использовании этой формы представления адрес записывается в виде четырех десятичных чисел — по одному числу на байт, — разделенных точками:

14.0.0.1

Некоторые адреса и адресные диапазоны сети Интернет используются для реализации специфических режимов адресации. К ним относятся:

- ❒ адрес сети;
- ❒ локальный адрес узла;
- ❒ адреса типа Broadcast;
- ❒ адреса типа Multicast;
- ❒ адреса типа loop back;
- ❒ адреса частных сетей.

Адрес сети

Для обозначения адреса сети используется специальный адрес, который состоит из номера сети и использует в качестве идентификатора узла значение "все нули". Для приведенного выше примера адрес сети должен быть представлен следующим образом:

14.0.0.0

Локальный адрес

Для обозначения локального узла используется адрес, состоящий из нулевых байтов:

0.0.0.0

Broadcast-адреса протокола IP

Адреса данного типа используются в тех случаях, когда содержимое пакета адресовано группе абонентов сети. В сетях IP применяются два типа Broadcast-адресов:

- адреса типа Limited Broadcast;
- адреса типа Directed Broadcast.

Адреса первого из указанных типов используются в том случае, когда пакет адресован всей сети в целом. В этом случае все 32 разряда адреса формируются равными "1":

255.255.255.255

В тех случаях, когда пакет адресован всем узлам конкретной сети, должен быть использован адрес типа направленный (directed) Broadcast. У адресов данного типа единицами формируется только значение адреса узла. Поле адреса сети в данном случае соответствует номеру сети, в которую передается пакет. Ниже приведен пример адреса Directed Broadcast для сети 14.0.0.0.

14.255.255.255

Адрес loop back

Адрес типа loop back (петля) представляет собой виртуальный адрес, который присваивается любому интерфейсу, сконфигурированному для обработки пакетов протокола IP. Этот адрес имеет только локальное значение в пределах данного узла и может быть использован для проверки функционирования программных компонентов, используемых для реализации в данном устройстве стека протоколов TCP/IP. Для адресов данного типа используется одна из сеток класса A, в качестве номера узла обычно используется значение "1". Ниже приведен пример адреса типа loop back.

127.0.0.1

Адреса частных сетей

Как уже было отмечено выше, размер сети определяется максимальным числом активных узлов, которое, в свою очередь, ограничивается длиной поля узла сети в сетевом адресе.

Очень быстро выяснилось, что длина сетевого адреса IP (32 бита) является недостаточной для того, чтобы обеспечить одновременное информационное взаимодействие всех узлов в мире, поддерживающих протоколы TCP/IP.

Дело в том, что в тот момент, когда определялась структура этого сетевого адреса, невозможно было предположить, что сеть IP покроет весь мир и станет, по сути, безальтернативной.

Для того чтобы обеспечить уникальность используемых сетевых адресов IP, был создан специальный комитет, который называется IANA (Internet Assigned Number Authority) и предназначен для контроля над соблюдением правил назначения сетевых адресов. Непосредственное выделение уникальных сетевых адресов в регионах координируется центрами INTERNIC (Internet Network Information Center). Например, в Северной Америке эти функции наряду с некоторыми другими могут выполняться организацией ARIN (American Registry for Internet Numbers). В городах и регионах выделение адресного пространства осуществляется коммерческими организациями — провайдерами Интернета.

Однако потребности в адресном пространстве, которые существуют в настоящий момент, уже существенно превышают те возможности, которые могут быть реализованы при использовании существующей адресной схемы IP. Это объясняется появлением и широким распространением значительного числа приложений, которые функционируют в стеке TCP/IP HTTP (Hypertext Transfer Protocol), FTP и многие другие.

Помимо того что сам адрес IP не является достаточно длинным для того, чтобы использоваться для создания достаточного количества уникальных сетевых узлов, размер адресного пространства ограничивается также многочисленными "мертвыми" зонами. Действительно, если какой-либо организации (например, IBM) выделена сеть адресов IP класса A (9.0.0.0), размер адресуемого пространства сокращается приблизительно на 2^{24} независимо от того, может эта корпорация использовать это адресное пространство по назначению или нет. Для решения данной проблемы был разработан специальный механизм, который позволял использовать внеклассовое разбиение сетей. В этом случае сеть класса A может быть интерпретирована в качестве совокупности сетей класса B (например, 255 сетей 9.1.0.0—9.255.0.0). Однако этот метод не обеспечивает коренного решения проблемы и, кроме того, порождает дополнительные трудности (доработка протоколов маршрутизации и появление дополнительных "мертвых" зон). В рассмотренном выше примере адреса 9.1.0.0 и 9.1.255.255 уже не могут быть использованы для назначения узлам сети, поскольку являются адресом сети и адресом типа Directed Broadcast для данной сети, соответственно.

Вместе с тем было отмечено, что обеспечение глобальной уникальности используемого сетевого адреса не является актуальным для ряда классов корпоративных сетей, которые используют сервисы Интернета только для внутренних нужд. Для обеспечения возможности более широкого использования сервисов Интернета в корпоративных сетях комитет IANA зарезервировал специальное адресное пространство. Концепция частных сетей в Ин-

тернете и адресные диапазоны IP, которые могут быть использованы для построения данных сетей, приведены в RFC-1918.

Этот документ содержит определение нескольких категорий сетей, и относит к понятию "частная сеть" сети образованные из узлов, для обеспечения информационного взаимодействия между которыми не требуется обращения к глобальным ресурсам Интернета. К категории "частная сеть" этот документ относит также сети, узлы которых для обращения к информационным ресурсам глобальной сети используют специальные шлюзы.

В соответствии с RFC-1918, для построения сетей, которые попадают под определение "частная сеть", могут быть использованы диапазоны адресов Интернета, представленные в табл. 1.2.

Таблица 1.2. *Адреса "частных" сетей Интернет*

Класс	Начальный адрес	Конечный адрес	Число сетей
A	10.0.0.0	10.255.255.255	1
B	172.16.0.0	172.31.255.255	16
C	192.168.0.0	192.168.255.255	255

Диапазоны адресов, которые могут быть реально использованы для назначения узлам сети Интернет, представлены в табл. 1.3.

Таблица 1.3. *Адресное пространство Интернета*

Класс	Начальный адрес	Конечный адрес
A	0.1.0.0	126.0.0.0
B	128.0.0.0	191.255.0.0
C	192.0.1.0	223.255.255.0
D	224.0.0.0	239.255.255.255
E	240.0.0.0	247.255.255.255

Групповые адреса

Групповые IP-адреса предназначены для обращения к группе узлов по одному IP-адресу, а именно, адресу группы. Например, узлам P, Q и R назначен один групповой адрес — X. Тогда для того, чтобы отправить дейтаграмму всем трем узлам (P, Q и R), достаточно отправить ее по адресу X.

Групповые адреса принадлежат классу D и лежат в промежутке от 224.0.0.1 до 239.255.255.244. Из этого промежутка адресов группе узлов с определен-

ными IP-адресами присваивается дополнительный групповой адрес. Назначение этого адреса некоторому узлу означает, что этот узел становится приемником сетевого трафика, адресованного членам данной группы.

Внеклассовые сети Интернета

Для решения проблемы адресного дефицита в сети Интернет было предложено использовать внеклассовые сети. В сетевом адресе внеклассовых сетей граница между адресом сети и адресом узла определяется с точностью до бита. Таким образом, может быть создана сеть, у которой адрес сети занимает 30 разрядов, а адрес узла — 2 разряда сетевого адреса. Благодаря использованию таких внеклассовых сетей, адресное пространство сети Интернет может быть разделено на более мелкие непересекающиеся подпространства ("подсети") — subnets, с каждой из которых можно работать как с обычной сетью TCP/IP. Как правило, подсеть соответствует одной физической сети, например, одной ветви сети Ethernet.

Использование внеклассовых сетей позволяет экономить сетевые адреса и обеспечивает дополнительные удобства для администратора сети.

Допустим, ваша организация получила один сетевой адрес, например, адрес класса B. Предположим, что третий байт будет определять адрес подсети, а четвертый байт — адрес узла в ней, т. е. часть битов адреса сети заимствуется для использования их в качестве адреса подсети. На рис. 1.4 показан вариант организации подсети класса B.

Если администратор сети решил использовать восемь битов для организации подсети, то третий байт адреса IP-класса B обозначает адрес этой подсети. В приведенном примере адрес 128.10.1.0. относится к сети 128.10, подсети 1; адрес 128.10.2.0 относится к сети 128.10, подсети 2 и т. д.

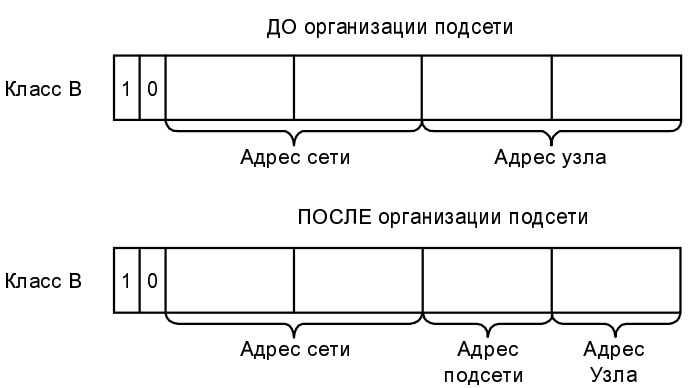


Рис. 1.4. Организация подсети сети класса B

Принцип разбиения сети на подсети довольно прост. Как правило, каждой физической сети, например, Ethernet или Token Ring, назначается отдельный адрес подсети или адрес сети. В некоторых случаях имеет смысл назначать одной физической сети несколько адресов подсетей.

Например, имеется сеть Ethernet, охватывающая три здания. Ясно, что при увеличении числа машин, подключенных к этой сети, придется ее разделить на несколько отдельных сетей Ethernet. Для того чтобы избежать необходимости менять IP-адреса, когда это произойдет, можно заранее выделить для этой сети три адреса подсети — по одному на каждое здание. Кстати, это полезно и в случае, когда не планируется физическое деление сети. Просто такая адресация позволяет сразу определить, где находится та или иная машина.

Структура сетевого адреса внеклассовой сети определяется маской этой сети. Для написания маски обычно используется тот же формат и техника представления адреса, что и для сетевого адреса IP. Обычно маска подсети указывается в файле стартовой конфигурации сетевого программного обеспечения вместе с адресом сети.

Биты маски, соответствующие адресу сети в сетевом адресе, должны быть равны 1, а биты, соответствующие адресу узла, должны быть равными 0. Так, в вышеописанном примере маска подсети будет задана числом 255.255.255.0. Таким образом, внеклассовая сеть в данном случае будет определена парой чисел: 128.10.0.0. и 255.255.255.0. В некоторых случаях для определения внеклассовой сети может быть использована укороченная нотация, в которой вместо самой маски задается только количество бит в адресе сети: 128.10.0.0/24.

Иногда, наоборот, полезно объединить несколько сетей одного класса в суперсеть. Например, четыре сети класса B 128.0.0.0, 128.1.0.0, 128.2.0.0 и 128.3.0.0 могут быть объединены в одну внеклассовую сеть 128.0.0.0/14.

Взаимодействие на сетевом уровне Интернета

Протокол Интернета является главным на сетевом уровне семейства протоколов TCP/IP и предназначен для передачи пакетов данных от одного узла сети к другому. Возможности Интернета по обеспечению надежности передачи данных в точку назначения весьма ограничены. Собственно протокол IP не предназначен для поддержания достоверного, стабильного соединения. У него также нет ни механизмов подтверждения получения информационных пакетов, ни средств обеспечения достоверности передаваемых данных, ни возможностей контроля потока данных от одного узла к другому. Этот протокол выполняет две основные функции: адресация и фрагментация дейтаграмм.

Протокол IP взаимодействует непосредственно с протоколами канального уровня. Канальный уровень, в соответствии со спецификацией многоуровневой системы протоколов, обеспечивает доступ данным к физическому носителю и играет для них роль "направляющего" при переходах через межсегментные или межсетевые шлюзы.

Протокол IP определяет маршрутизацию данных по сети до точки назначения или промежуточного шлюза. На шлюзах между различными сетями Интернет-дейтаграммы "разворачиваются" из "конвертов" локальных сетей и перенаправляются в тот канал, который соответствует маршруту пакета. После этого дейтаграммы опять "заворачиваются" в "конверт", подходящий для маршрута до следующего шлюза или точки назначения. Шлюз может разбить Интернет-дейтаграмму на фрагменты, если это необходимо для работы в следующей сети, и т. д.

На каждом компьютере, включенном в Интернет-сообщество, который играет роль отправителя, получателя, или шлюза сетевых пакетов, действуют модули обработки IP-дейтаграмм, которые функционируют по одним и тем же схемам идентификации адресов, фрагментации и дефрагментации пакетов и их маршрутизации через модули шлюзов сетей.

Модуль обработки IP обрабатывает каждую дейтаграмму как независимую от других дейтаграмм единицу данных, поскольку протокол не использует ни логических соединений, ни каких-либо других виртуальных каналов связи.

Структура дейтаграммы IP

Дейтаграмма протокола Интернета состоит из двух частей:

- заголовок дейтаграммы (Datagram Header);
- поле данных дейтаграммы (Datagram Data Area).

Информация пользователя размещается в поле данных дейтаграммы. В это поле могут быть помещены, в частности, блоки данных протоколов верхних уровней. В заголовке дейтаграммы размещается информация, которая необходима для того, чтобы доставить содержимое полезной нагрузки в требуемый узел назначения сети.

Формат заголовка дейтаграммы IP

Заголовок дейтаграммы IP не имеет фиксированной длины. Его длина определяется значением одного из полей. Структура дейтаграммы IP приведена на рис. 1.5.

В поле **VERS** заголовка дейтаграммы IP размещается номер версии протокола IP, который был использован для формирования данной дейтаграммы. Появление этого поля было вызвано желанием обеспечить совместимость по принципу "снизу вверх" старых версий с более новыми версиями прото-

кола IP. В данном поле современных дейтаграмм размещается значение, которое соответствует четвертой версии протокола IP. В поле **HLEN**, которое занимает 4 бита в заголовке дейтаграммы IP, размещается длина заголовка дейтаграммы, выраженная в 32-битовых словах.

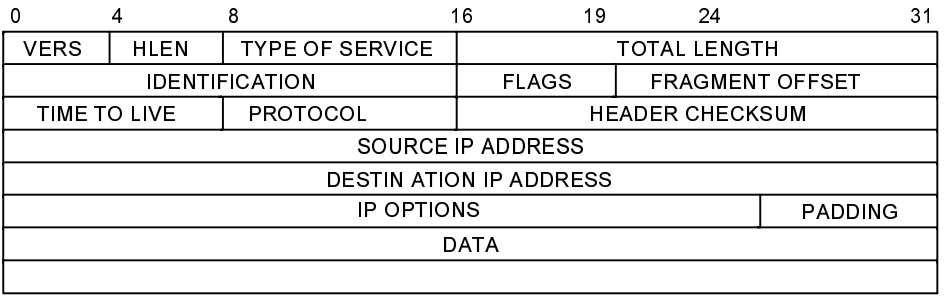


Рис. 1.5. Структура дейтаграммы IP

Поле **TYPE OF SERVICE** иначе обозначаемое **TOS**, занимает один байт в заголовке дейтаграммы и, в свою очередь, подразделяется на несколько полей:

- поле **PRECEDENCE**;
- бит **D**;
- бит **T**;
- бит **R**.

Поле **PRECEDENCE** (превосходство) занимает три бита в поле **TOS** и предназначено для явного указания степени важности информации, которая размещается в поле полезной нагрузки данной дейтаграммы. Наличие данного поля обеспечивает возможность использования механизмов, приоритетного разрешения конфликтов при доступе к транспортным ресурсам со стороны различных информационных потоков. Биты **D**, **T**, **R** используются для того, чтобы указать, какой тип каналов передачи данных должен быть использован для обеспечения доставки полезной нагрузки данной дейтаграммы. Бит **D** указывает на то, что при передаче данной дейтаграммы должно отдаваться предпочтение каналам с минимальной задержкой. Наличие бита **T** указывает на то, что при передаче данной дейтаграммы должно отдаваться предпочтение каналам с максимальной пропускной способностью. Бит **R** устанавливается в том случае, если для передачи дейтаграммы должны использоваться преимущественно надежные каналы. Значения этих битов могут быть использованы протоколами маршрутизации, для того, чтобы обеспечить одновременное выполнение требований надежности и эффективности доставки данных пользователя. Так, например, для передачи дейтаграммы, для которой несущественно наличие задержек, может быть использован более экономичный спутниковый канал. Для успешного функ-

ционирования этого аппарата, однако, необходимо, чтобы соответствующие возможности по анализу требований к характеристикам используемых каналов были реализованы во всех маршрутизаторах сети.

Поле **TOTAL LENGTH** занимает 16 бит в заголовке дейтаграммы и предназначено для указания размера поля данных, выраженного в байтах. Максимальный размер поля полезной нагрузки составляет, таким образом, 65535 байт. Однако при передаче такой дейтаграммы через сеть IEEE 802.3 (IEEE — Institute of Electrical and Electronic Engineers) неминуемо возникнет необходимость в ее преобразовании в совокупность дейтаграмм, каждая из которых не превышает 1518 байт.

Фрагментация и дефрагментация IP-дейтаграмм

Одной из важных характеристик протокола канального уровня, является параметр MTU (Maximum Transfer Unit) — максимальная длина передаваемого блока данных. Ограничение этого размера, как правило, определяется типом используемой процедуры доступа к каналу и не может быть изменено. В процессе передачи дейтаграммы по сети она может проходить через каналы с различными значениями MTU. Очевидно, что для успешной передачи дейтаграммы в данном случае необходимо, чтобы она была фрагментирована на части, размер которых соответствует минимальному значению MTU среди всех используемых в данном случае каналов передачи данных. Каждая из полученных таким образом дейтаграмм называется фрагментом.

Определение

Процесс преобразования одной крупной дейтаграммы в несколько мелких дейтаграмм называется *фрагментацией*.

Определение

Процесс восстановления исходного вида передаваемого сообщения путем соединения переданных фрагментов называется *дефрагментацией*.

Значение поля **TOTAL LENGTH** у всех фрагментов, кроме последнего, для одного и того же сообщения должно быть одинаковым. Каждый фрагмент снабжается своим заголовком и передается по сети независимо от других фрагментов этого же сообщения. При поступлении первого фрагмента на станцию назначения включается специальный таймер фрагментации, определяющий максимальную величину временного интервала, в течение которого на данную станцию должны поступить все фрагменты передаваемого сообщения. В том случае, если по истечении установленного интервала на станцию поступят не все фрагменты сообщения, исходная дейтаграмма не будет считаться принятой данной станцией.

Ниже приведено описание полей заголовка дейтаграммы, которые предназначены для управления процессом ее фрагментации и обеспечения возможности восстановления исходного вида переданного сообщения.

Поле **IDENTIFICATION** занимает 2 байта в заголовке дейтаграммы. В данном поле должен быть расположен уникальный идентификатор, формируемый передающей станцией для того, чтобы обеспечить отличие фрагментов одной дейтаграммы от фрагментов другой такой же дейтаграммы. На самом деле, уникальным должен быть не сам этот идентификатор, а совокупность **IDENTIFICATION + SOURCE IP ADDRESS**. Таким образом, все фрагменты одной дейтаграммы должны иметь уникальное сочетание указанных полей, что может быть использовано в качестве признака при дефрагментации.

Поле **FLAGS** занимает 3 бита в заголовке дейтаграммы. Два бита (младший и средний) данного поля предназначены для указания отношения данной дейтаграммы к процессу фрагментации. Третий бит в данном случае не используется. Значение младшего бита определяет, возможно или нет выполнение фрагментации данной дейтаграммы. Средний бит имеет название **MF** (More Fragments). Значение этого бита определяет порядок данного фрагмента в общей цепочке фрагментов данного сообщения. У всех, кроме последнего, фрагментов одного сообщения бит **MF** должен быть установлен в 1. Очевидно, что значение данного бита может быть также использовано в качестве общего признака фрагментации сообщения.

Содержимое поля **FRAGMENT OFFSET** определяет смещение, выраженное в 8-байтовых отрезках. Это смещение соответствует месту данного фрагмента в исходном сообщении. Для первого фрагмента сообщения содержимое данного поля должно быть равно 0. Значение поля **FRAGMENT OFFSET** последующих фрагментов определяется следующим образом:

$$\text{FRAGMENT OFFSET } (n) = \text{TOTAL LENGTH} \times (n - 1),$$

где n — порядковый номер фрагмента.

Для повышения эффективности использования ресурсов сети желательно, по возможности, избегать использования процедур фрагментации и дефрагментации, поскольку это приводит к появлению дополнительных задержек в передаче сообщений. Более того, недостаточно аккуратная реализация этих процедур может привести к возникновению брешей в системе обеспечения безопасности.

Содержимое поля **TIME TO LIVE** используется для того, чтобы ограничить возможность бесконечного перемещения в сети ошибочно адресованных дейтаграмм. Значение данного поля формируется при первой передаче каждой дейтаграммы. При перемещении этой дейтаграммы по сети значение поля **TIME TO LIVE** уменьшается на время, которое было затрачено на ее обработку в каждом активном узле. В том случае, если значение этого поля

становится равным 0, дейтаграмма считается просроченной и уничтожается. Маршрутизатор, который уничтожил дейтаграмму, должен послать соответствующее сообщение в адрес ее источника.

Содержимое поля **PROTOCOL** определяет тип протокола, который был использован при формировании полезной нагрузки данной дейтаграммы.

В поле **HEADER CHECKSUM** размещается проверочная контрольная сумма заголовка кадра.

Обзор протоколов, применяемых в Интернете

Для организации информационного взаимодействия в сети Интернет используются протоколы различных уровней. Ниже приведены назначение и краткие характеристики некоторых из этих протоколов.

- *Уровень сопряжения с физической средой (канальный)* обеспечивает надежный транзит данных через физический канал. Этот уровень решает задачи физической адресации (в противоположность сетевой или логической адресации), топологии сети, линейной дисциплины (каким образом конечной системе использовать сетевой канал), уведомления о неисправностях, упорядоченной доставки блоков данных и управления потоком информации.

  Ниже этого уровня расположен только аппаратный уровень, который определяет электротехнические, механические, процедурные и функциональные характеристики активизации, поддержания и деактивизации физического канала между конечными системами (уровни напряжений, синхронизацию изменения напряжений, скорость передачи физической информации, максимальные расстояния передачи информации, физические соединители и др.).

  Протоколы, обеспечивающие функции канального уровня, тесно связаны с физической (аппаратурной) средой, в которой они работают, например, Ethernet, Token Ring, FDDI (Fiber Distributed Data Interface), PPP, ISDN (Integrated Services Digital Network) и др. В семействе TCP/IP нет протоколов, принадлежащих этому уровню, за счет этого и достигается аппаратная независимость семейства TCP/IP. Однако в состав семейства входят протоколы ARP (Address Resolution Protocol) и RARP (Reverse Address Resolution Protocol), обеспечивающие взаимодействие между данным — канальным уровнем и следующим — сетевым уровнем TCP/IP, а именно, обеспечивающие трансляцию сетевых адресов в адреса локальной сети.

- *Сетевой уровень* — это комплексный уровень. Он обеспечивает возможность соединения и выбор маршрута между двумя конечными системами, подключенными к разным "подсетям", которые могут находиться в

разных географических пунктах. Поскольку две конечные системы, желающие организовать взаимодействие, могут быть разделены значительным географическим расстоянием и множеством других подсетей, сетевой уровень обеспечивает маршрутизацию пакетов от одного удаленного узла к другому.

К этому уровню в TCP/IP относится межсетевой протокол IP, который является базовым в структуре TCP/IP и обеспечивает доставку пакета по месту назначения — маршрутизацию, фрагментацию и сборку поступивших пакетов на узле получателя. Кроме того, к этому уровню принадлежит протокол ICMP (Internet Control Message Protocol), в функции которого входят, в основном, сообщения об ошибках и сбор информации о работе сети.

Оптимальные маршруты через последовательность соединенных между собой подсетей выбирают протоколы маршрутизации. К ним относятся такие протоколы как RIP (Routing Information Protocol), EGP (Exterior Geteway Protocol), BGP (Border Gateway Protocol), OSPF (Open Shortest Path First) и др.

- *Транспортный уровень* предоставляет услуги по транспортировке данных. Эти услуги избавляют механизмы передачи данных прикладного уровня от необходимости вникать в детали транспортировки данных. В частности, заботой транспортного уровня является решение таких вопросов, как надежная и достоверная транспортировка данных через сеть. Транспортный уровень реализует механизмы установки, поддержания и упорядоченного закрытия каналов соединения, механизмы систем обнаружения и устранения неисправностей транспортировки, управления информационным потоком.

  Транспортный уровень семейства TCP/IP представлен протоколами TCP и UDP. TCP обеспечивает транспортировку данных с установлением соединения, в то время как UDP работает без установления соединения. Оба эти протокола имеют дело с конкретными процессами (приложениями) на компьютере и могут обеспечивать связь процессов на различных компьютерах сети, хотя в их компетенцию еще не входит управление сеансом работы. Если TCP обеспечивает полный сервис транспортного уровня — надежность, достоверность и контроль соединения, то UDP может только отправлять пакеты от одного процесса к другому без какого-либо дополнительного сервиса, за исключением, разве что проверки контрольной суммы переданных данных.

- *Прикладной уровень* идентифицирует и устанавливает наличие предполагаемых партнеров для связи, синхронизирует совместно работающие прикладные программы, устанавливает соглашение по процедурам устранения ошибок и управления целостностью информации. Кроме того, протоколы прикладного уровня определяют, имеется ли в наличии дос-

таточно ресурсов для предполагаемой связи. Прикладной уровень отвечает также за то, чтобы информация, посылаемая из прикладного уровня одной системы, была читаемой на прикладном уровне другой системы. При необходимости он осуществляет трансляцию между множеством форматов представления информации путем использования общего формата и структур данных, а также согласует синтаксис передачи данных для прикладного уровня. Прикладной уровень устанавливает и завершает сеансы взаимодействия между прикладными задачами, управляет этими сеансами, синхронизирует диалог между объектами и управляет обменом информации между ними. Кроме того, прикладной уровень предоставляет средства для отправки информации и уведомления об исключительных ситуациях передачи данных.

- Комплект протоколов Интернета включает в себя большое число протоколов высших уровней, имеющих самые разнообразные применения, в том числе управление сети, передача файлов, распределенные услуги пользования файлами, эмуляция терминалов и электронная почта. Например, протокол передачи файлов FTP обеспечивает способ перемещения файлов между компьютерными системами. Telnet обеспечивает виртуальную терминальную эмуляцию. Протокол управления простой сетью SNMP (Simple Network Management Protocol) является протоколом управления сетью, используемым для сообщения об аномальных условиях в сети и установления значений допустимых порогов в сети. Telnet является популярным протоколом, который позволяет терминалу с интеллектом связываться с отдаленными компьютерами таким образом, как если бы они были непосредственно подключенными мониторами. Простой протокол передачи почты SMTP (Simple Mail Transfer Protocol) обеспечивает механизм передачи электронной почты между пользователями сети. А комбинация протоколов NFS (Network File System), XDP (External Data Representation) и RPC (Remote Procedure Call) обеспечивает прозрачный доступ к ресурсам отдаленной сети.

Примечание

Здесь необходимо заметить, что каждый из трех вышеупомянутых протоколов — NFS, XDR и RPC реализует свой набор функций, которые составляют подуровни прикладного уровня. Так, открытием, закрытием и управлением сессией обмена, синхронизацией передачи данных управляет RPC, согласованием форматов данных занимается XDP, а NFS управляет синхронизацией всей работы.

Прикладные сервисы TCP/IP

Изучение протоколов сопряжения с физической средой выходит за рамки данной книги и будет нас интересовать лишь постольку, поскольку это связано с работой IP-протокола.

Структура IP и протоколов уровня TCP, их функциональность и принцип работы изложены далее с точки зрения тех возможностей, которые они содержат, и их роли в Интернет-пространстве. К настоящему моменту эти протоколы в целом завершили процесс формирования и составляют область киберпространства, которую уже никто старается не изменять.

Пользователей и разработчиков программ более интересуют протоколы или сервисы верхнего уровня. По сути, с ними мы имеем дело, работая за компьютером. Они осуществляют и контролируют те функции, ради которых мы пользуемся Интернетом. И именно благодаря им Интернет стал удобным, дружественным и многофункциональным средством общения. Ниже перечислены наиболее широко распространенные и в настоящее время ставшие "традиционными" сервисы работы в Интернете.

- *Сервис разрешения имен* предоставляет механизм именования ресурсов сети, чтобы любой компьютер сети мог осуществить доступ к определенному ресурсу Интернет-пространства по имени этого ресурса. О каждом компьютере, подсоединенном к Интернету и предоставляющем свои ресурсы для использования другими, мы должны иметь возможность знать не только его адрес, но и вид предоставляемых услуг. Это приводит к одной из центральных проблем, которую необходимо разрешить, — проблеме хранения и управления именами ресурсов сети и их адресами. Проблема соотнесения имен и адресов гораздо более сложная, чем это может показаться на первый взгляд. Поскольку в больших, географически разделенных системах нереально хранить и обновлять на каждом компьютере сети базу данных, необходимую для хранения имен и адресов, то это, по сути, задача построения распределенной базы данных. Сервис разрешения имен DNS (Domain Name System), который будет подробно описан далее, и обеспечивает управление такой базой.
- *Компьютерная почта* позволяет обмениваться сообщениями с пользователями на других компьютерах. Первоначально схема работы электронной почты выглядела следующим образом. Пользователь, который желает получать почту, регистрирует свой почтовый ящик (файл) на почтовом сервере, в который будет складываться вся поступающая на него корреспонденция. При поступлении нового сообщения данному пользователю система электронной почты просто добавляет это сообщение к его почтовому файлу. Для того чтобы отправить сообщение, вам нужно присоединиться к компьютеру с почтовым ящиком получателя и... "дописать" в его почтовый файл ваше сообщение. Это было приемлемо, когда компьютеров было не слишком много и большинство из них представляли собой mainframe, на которых зарегистрированы их пользователи. *Mainframe* — большие ЭВМ, подключаемые к нескольким терминалам, каждый из которых обслуживал отдельного пользователя. Когда количество компьютеров стало расти, начали появляться почтовые серверы, которые служили "пересылочными пунктами" сообщений. Так,

чтобы отправить сообщение своему приятелю, связь с сервером которого напрямую не очень устойчива, вы можете отправить это сообщение на один из почтовых серверов, который и займется переправкой сообщения самостоятельно. С появлением персональных компьютеров это стало неудобно, так как при отправке почты нужно быть уверенным, что, как минимум, компьютер адресата включен и почтовый сервер сможет до него "достучаться". Поэтому были созданы почтовые серверы, которые хранят сообщения для своих пользователей. Чтобы прочитать полученные сообщения, вам нужно присоединиться к такому серверу и работать со своими сообщениями как с отдельными файлами. Работу с электронной почтой осуществляют разные протоколы. Мы подробно остановимся на SMTP, POP3 (Post Office Protocol) и IMAP4 (Internet Message Access Protocol).

- *Система новостей UseNet.* UseNet — это распределенная система ведения дискуссий, которая состоит из нескольких групп, называемых группами новостей. Группы новостей организованы в определенном порядке, основанном на распределении дискуссий по темам (например, отдых, спорт, новости, информация, религия и др.). Внутри каждой из этих групп может быть от нескольких до тысяч подгрупп, которые обладают своей структурой. Принцип этой организации подобен структуре размещения каталогов и подкаталогов на жестком диске. Группы новостей позволяют пользователям с общими интересами обмениваться интересными сообщениями, отправлять свои статьи и отвечать на заметки других пользователей. После того как статья отправлена в группу новостей UseNet, она рассылается через сервис на другие компьютеры, присоединенные к Интернету, на которых работает сервис UseNet. Структура групп UseNet отличается от структуры почтовых сервисов тем, что требует централизованного хранения статей на сервере UseNet, поскольку к банку статей должны иметь доступ все пользователи сети, подсоединенные к этому серверу. Система UseNet позволяет вести дискуссии по выбранной теме, осуществлять фильтрацию статей по ключевым словам или просматривать специальные файлы, включенные в статью (например, картинки). Протокол, используемый UseNet, называется NNTP (Network News Transport Protocol). NNTP — это прикладной протокол высокого уровня, который используется для обеспечения связи между различными серверами, работающими с программным обеспечением UseNet.
- *World Wide Web* (WWW) состоит из компьютеров, объединенных в единую сеть, которые предоставляют определенный тип доступа к хранящейся на них информации. На таких компьютерах работает программное обеспечение, предоставляющее возможность пользователям Интернета подсоединяться и пользоваться его средствами для поиска и выбора информации. В основном WWW предназначена для тиражирования рек-

ламной, образовательной или другой информации, которую компания, владеющая сервером WWW, считает целесообразным выставить как свою визитную карточку. WWW предоставляет высокоуровневый протокол работы со своим сервисом — HTTP. Этот протокол обеспечивает навигацию по миру WWW, аутентификацию пользователя, если этого требует сервер WWW, формирует информационные запросы и передает запрошенную информацию пользователю. По сравнению с другими "традиционными" высокоуровневыми протоколами Интернета, HTTP — самый молодой, но самый широко используемый протокол.

- *Передача файлов (FTP)* — это один из первых широко используемых сервисов Интернета. В основе этого сервиса и его функций лежит протокол передачи файлов. Он предоставляет возможность пользователю, работающему на каком-либо компьютере, считывать файлы с другого компьютера или передавать файлы на другой компьютер, на котором работает FTP-сервис. Протокол передачи файлов FTP имеет двух "младших братьев" — протоколы SFTP (Simple File Transfer Protocol) и TFTP (Trivial File Transfer Protocol). Эти протоколы выполняют те же задачи, что и FTP, но пользуются для этого несколько упрощенными и облегченными механизмами передачи.

- *Удаленный терминал и терминальные серверы.* Протокол удаленного терминала (TELNET) предоставляет возможность работать на удаленном компьютере сети, поддерживающим сессии TELNET. В начале сессии вы должны подсоединиться к удаленному компьютеру и, если необходимо, ввести свой идентификатор пользователя и пароль. После этого и до окончания сессии все введенные данные попадут в удаленный компьютер. Физически вы вводите данные в свой компьютер, но TELNET делает ваш компьютер как бы невидимым, пока вы работаете с удаленным узлом. Во время работы с TELNET ваш компьютер становится только средством, обеспечивающим сеанс связи. Этот сервис в прошлом составлял основу работы Интернета. Сегодня TELNET используется в основном для удаленного администрирования компьютерных систем и для доступа к узлам Интернета.

  Если компьютерная система не может долго находиться в терминальном соединении с удаленным компьютером, на помощь приходят *терминальные серверы.* Такой сервер представляет собой компьютер, который умеет только запускать TELNET или использовать какой-либо другой протокол для входа в удаленную систему. Если ваш компьютер подсоединен к такому серверу, то для того, чтобы открыть соединение, вам достаточно набрать имя компьютера, с которым вы хотите соединиться. Терминальный сервер позволяет вам держать одновременно открытыми несколько TELNET-соединений, быстро переключаться между ними и перенаправлять потоки данных. Кроме того, терминальные серверы поддерживают сервис разрешения имен DNS и другие сетевые протоколы.

- *Сетевая файловая система (NFS — Network File System)* позволяет одному компьютеру работать с файлами (или устройствами) другого компьютера. Название сервиса берет начало от UNIX, где все устройства компьютера представлены в виде файлов dev/*. Сервис сетевой файловой системы создает иллюзию, что устройства (диски, порты и т. п.) удаленной системы напрямую подсоединены к вашему компьютеру. Для того чтобы получить доступ к файлу удаленного компьютера, вам не нужно использовать какую-либо специальную утилиту. Ваш компьютер просто будет "думать", что ему добавили еще одно устройство, например, диск или порт.

  Использование сетевой файловой системы может быть полезно или даже просто необходимо во многих случаях. Например, при использовании больших по объему справочников или документов несколькими сотрудниками с разных рабочих мест. Разумнее разместить справочники и документы на одном компьютере, который предоставлял бы доступ к ним через сетевую файловую систему всем остальным пользователям. Это, во-первых, позволит каждому пользователю сэкономить дисковое пространство на своем компьютере, а во-вторых, позволит просто и быстро проводить обновление и резервирование данных справочников или документов. В некоторых случаях компьютеры пользователя могут вообще не иметь своих дисков, а работать через сервис удаленной файловой системы (необходимо заметить, что этот вариант не имеет ничего общего с TELNET, поскольку задача выполняется в памяти и на процессоре компьютера пользователя, а не на удаленной машине, как при работе с TELNET).

- *Мониторинг и управление сетью (SNMP)* — простой протокол управления сетью. Он является протоколом прикладного уровня, предназначенным для облегчения обмена информацией управления между сетевыми устройствами. Пользуясь информацией SNMP (такой, как показатель числа пакетов в секунду или коэффициент сетевых ошибок), сетевые администраторы могут более просто управлять производительностью сети и обнаруживать и решать сетевые проблемы. Как правило, протокол работает на базе UDP. Основная функциональная единица по сбору сетевой информации протокола, агент SNMP, — это программный модуль, работающий в управляемых устройствах. Агенты собирают информацию об управляемых устройствах, в которых они работают, и делают эту информацию доступной для систем управления сетями — NMS (Network Management Systems). SNMP позволяет как проводить мониторинг сети, так и управлять устройствами исходя из полномочий и возможностей агентов.

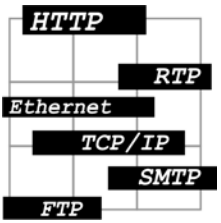
- *Удаленное выполнение процедур (RPC — Remote Procedure Call)* позволяет по запросу запускать на удаленном компьютере определенные программы. Например, если вы делаете основную работу на своем компьютере,

но для выполнения некоторых задач требуются ресурсы более производительного компьютера, то решением этой проблемы будет использование сервиса удаленного выполнения процедур. Необходимо заметить, что в настоящее время этот сервис имеет множество реализаций и далек от завершения. Полностью свои возможности он, вероятно, сможет раскрыть в распределенных вычислительных средах, создание которых в настоящее время только начинается.

- *Сетевая печать*. Этот сервис позволяет вам печатать на принтерах, подсоединенных к удаленным компьютерам, как будто они подключены к вашему. Можно с большой долей уверенности сказать, что если в организации есть компьютерная сеть, то из всех остальных сервисов этот используется чаще всего.

Безусловно, Интернет не ограничивается описанными выше сервисами, но, тем не менее, они используются в подавляющем большинстве основных приложений, работающих сегодня с Интернетом.

Глава 2



Протоколы канального уровня

IP-адрес представляет собой логическое выражение, которое никак не связано с конкретной физической реализацией сети, по которой передается дейтаграмма. *IP-адресация* — это адресация сетевого уровня многоуровневой системы протоколов. Для рассмотрения взаимодействия протокола IP с протоколами более низкого — канального уровня, необходимо обратиться к конкретной реализации той или иной сети.

Назначением протоколов канального уровня является предоставление протоколу сетевого уровня услуги по передаче блока данных сетевого уровня через физический канал. Тип протокола канального уровня однозначно соответствует типу используемого канала передачи данных и освобождает протокол сетевого уровня от необходимости учитывать специфику этого канала. При построении вычислительных сетей и систем передачи данных используются каналы двух основных типов:

- двухточечные (Point-To-Point);
- многоточечные (Point-To-Multipoint).

Каналы первого типа соединяют, как это видно из их названия, только две станции и используются каждой из них в монопольном режиме. Каналы такого типа применяются для обеспечения доступа клиента к удаленным ресурсам сети, а также для построения глобальных вычислительных сетей путем объединения групп локальных сетей через маршрутизирующие устройства.

Каналы второго типа обычно организуются для обеспечения множественного разделяемого доступа группы станций к единой среде передачи данных. Такой тип доступа положен в основу большинства современных локальных вычислительных сетей.

Таким образом, в качестве канала передачи для IP дейтаграммы может быть использован сегмент локальной вычислительной сети или канал передачи данных, соединяющий два узла сети.

Непосредственная доставка дейтаграмм протокола IP между узлами сети производится с использованием протокола канального уровня. В сетях с множественным доступом к среде передачи, в дополнение к этой функции, протоколы канального уровня выполняют также отображение сетевых

IP адресов на адреса канального уровня, называемые обычно MAC–адресами (Media Access Control).

Канальный уровень передачи данных, формально называемый также уровнем звена передачи данных, обеспечивает транзит данных через физический канал. Выполняя эту задачу, протокол канального уровня, при необходимости, решает вопросы физической адресации, обеспечения дисциплины доступа к каналу передачи данных, фиксирования и передачи уведомлений о неисправностях, упорядочивания доставки блоков данных канального уровня и управления темпом информационного потока.

Благодаря четкому разделению функций протоколов сетевого и канального уровня, стек протоколов TCP/IP одинаково успешно функционирует в различных сетевых средах: Ethernet, Token Ring, X.25, спутниковых каналах, коммутируемых каналах последовательного доступа. Эта возможность обеспечивается наличием соответствующих протоколов канального уровня и правил инкапсуляции дейтаграмм IP-протокола для всевозможных типов используемых каналов. Одним из наиболее известных протоколов, обеспечивающих построение сетей Интернет на каналах последовательного доступа — Serial Line является протокол SLIP (Serial Line IP).

Протокол SLIP

В конце 80-х сеть Интернет начала испытывать резкий рост числа главных вычислительных машин, обеспечивающих работу TCP/IP. Большая часть этих машин была подсоединена к локальным сетям — LAN различных типов, причем наиболее популярной была Ethernet. Другие вычислительные машины соединялись через глобальные сети — WAN, такие, как сети передачи данных общего пользования X.25. На раннем этапе развития сети сравнительно небольшое число компьютеров были подключены к Интернету посредством выделенных линий связи с непосредственным двухточечным соединением. Однако, по мере роста сети Интернет, когда к сети стали подключаться индивидуальные пользователи, популярность этого способа подключения начала резко увеличиваться. Во многом это объяснялось популярностью последовательных интерфейсов и развитием сетей, способных предоставить услуги по аренде выделенных линий и последовательных каналов.

Каналы связи с непосредственным соединением принадлежат к числу старейших методов передачи информации, и почти каждый сетевой узел может поддерживать непосредственные соединения. Например, последовательные асинхронные интерфейсы RS-232C можно встретить практически на любом сетевом устройстве.

SLIP (Serial Line Internet Protocol, Протокол подключения к Интернету по последовательной линии) выполняет функцию обрамления дейтаграмм IP

для обеспечения их передачи по линии последовательного доступа. Этот протокол не обеспечивает выполнение многих функций, свойственных другим протоколам канального уровня (адресация, идентификация, проверка ошибок), однако благодаря своей простоте он нашел достаточно широкое применение. В настоящее время протокол SLIP используется на коммутируемых и выделенных линиях последовательного доступа на скоростях от 1200 бит/с до 19,2 Кбит/с.

Для обозначения границ передаваемых дейтаграмм IP в протоколе SLIP используются специальные символы END (0xc0) и ESC (0xdb). Символ END предназначен для непосредственного обрамления дейтаграммы, а символ ESC используется для обеспечения символьной прозрачности протокола SLIP. Передача дейтаграммы по протоколу начинается и завершается символами END. В том случае, если в теле передаваемой дейтаграммы встречается байт со значением символа END, драйвер протокола заменяет этот символ комбинацией из двух символов: ESC и ESC_END (0xdc), то есть на 2-байтную строку db dc, для того чтобы избежать некорректного завершения пакета. Подобным образом производится замена байта со значением символа ESC на пару символов ESC ESC_ESC (0xdd), то есть на 2-байтную строку (0xdbdd).

Примечание

Здесь и далее последовательность символов "0x" используется для обозначения числа, представленного в шестнадцатеричной системе исчисления.

Символ ESC, определяемый протоколом SLIP, не является эквивалентом символа ESC из таблицы ASCII (American Standard Code for Information Interchange).

К достоинствам протокола SLIP можно отнести простоту его реализации и экономичность вследствие относительно небольшого объема служебной информации, передаваемой в кадре.

Размер кадра протокола SLIP формально не ограничен, однако, следует учитывать, что некоторые реализации драйвера SLIP для UNIX построены исходя из предположения, что максимальный размер кадра SLIP не превышает 1006 байт.

SLIP проектировался много лет назад, поэтому некоторые недостатки, которые он имел тогда и которые в то время казались несущественными, в настоящее время представляют серьезные неудобства при его использовании.

- Из-за того, что кадр протокола SLIP не содержит поля "Тип Протокола", протокол SLIP не может быть использован для одновременного обслуживания нескольких протоколов сетевого уровня.
- Поскольку протокол SLIP не поддерживает механизмов автоматического определения адресов сетевого уровня, абонент SLIP должен явно указать адрес удаленного партнера до начала информационного взаимодействия.

- Из-за того, что протокол SLIP не обеспечивает обнаружение и исправление ошибок в передаваемых данных на приемной стороне. Предполагается, что эти функции выполняются протоколами последующих уровней.
- Потому, что протокол SLIP не имеет механизмов установления подлинности абонента, что ограничивает возможности его применения в ответственных сетях.

К недостаткам протокола можно также отнести отсутствие встроенных алгоритмов сжатия передаваемых данных.

Несмотря на описанные недостатки, применение протокола SLIP может быть оправдано в тех случаях, когда эти недостатки не имеют решающего значения.

Протокол SLIP, например, может быть эффективно использован для обеспечения управления сетевыми компонентами по протоколу SNMP через последовательный интерфейс. В этом случае решающее значение имеет простота реализации, а такие функции, как аутентификация и коррекция ошибок, просто не требуются, поскольку соединение узлов тогда выполняется в пределах одного помещения.

В то же время не представляется перспективным использование протокола SLIP для организации удаленного доступа к сетевым ресурсам. Общепризнанно, что для реализации подобных приложений гораздо более эффективным является использование более сложного и эффективного протокола канального уровня — PPP.

Протокол PPP

Одной из причин относительно невысоких темпов роста числа подключений к Интернету с использованием последовательных каналов передачи данных долгое время являлось отсутствие эффективного стандартного протокола канального уровня.

В то время было разработано достаточно много нестандартных протоколов, обеспечивающих передачу блоков данных сетевого уровня через последовательную линию, но ни один из них не удовлетворял в полной мере растущим требованиям пользователей ни сети Интернет, ни каких-либо других сетей.

Для решения этой проблемы был разработан протокол PPP (Point-to-Point Protocol, Протокол двухточечного соединения).

PPP является универсальным протоколом канального уровня, который обеспечивает инкапсуляцию блоков данных протокола сетевого уровня в кадры для их последующей передачи через синхронные или асинхронные, выделенные или коммутируемые каналы в режиме полного дуплекса.

Универсальность структуры кадра PPP, тщательно проработанной для сохранения совместимости с наиболее часто используемыми аппаратными средствами, обеспечивает возможность мультиплексирования различных протоколов сетевого уровня одновременно в одном канале.

В то же время, этот протокол является достаточно экономичным: всего 8 дополнительных байтов требуется для оформления инкапсулированного блока данных канального уровня в том случае, когда используется заданный по умолчанию формат HDLC. При применении приложений, для которых важно наиболее полное использование пропускной способности канала, оформление кадра может быть сокращено до 2 или 4 байтов.

Примечание

Для обозначения каждой из сторон, участвующих в информационном взаимодействии с использованием протокола PPP будут использоваться термины "PPP-партнер" или "партнер".

Функциями протокола PPP являются:

- присвоение и управление адресами протокола сетевого уровня (IP);
- асинхронное и синхронное формирование пакета данных;
- мультиплексирование протоколов сетевого уровня;
- конфигурирование канала связи;
- проверка качества канала связи;
- обнаружение ошибок при передаче данных и согласование параметров сжатия информации.

Для обеспечения эффективного выполнения этих функций в протоколе PPP применяются экономичный алгоритм формирования кадра на основе протокола HDLC и два типа вспомогательных протоколов: протокол Управления Каналом — LCP (Link Control Protocol) и семейство Протоколов Управления Сетью — NCP (Network Control Protocols).

Протокол LCP используется для автоматического согласования формата инкапсуляции, управления размерами кадров, установления подлинности партнера по соединению, обнаружения замыкания канала — короче говоря, обеспечивает непосредственное управление функционированием канала.

Протоколы NCP предназначены для организации и выбора конфигурации PPP для взаимодействия с различными протоколами сетевого уровня. Для взаимодействия с протоколом IP, в частности, используется IPCP (IP Control Protocol) и NCP (Network Control Protocol) протокола PPP. Протокол IPCP обеспечивает настройку, активизацию и блокирование модулей IP на обоих концах PPP-соединения, присвоение и управление IP-адресами, что является особенно актуальной и сложной задачей при использовании коммутируемых каналов.

При разработке протокола PPP предполагалось, что он должен быть максимально прост для конфигурирования. Задаваемые по умолчанию значения изменяемых параметров протокола PPP, соответствуют его наиболее используемой конфигурации. Заданная по умолчанию конфигурация может быть изменена по инициативе любого PPP-партнера, при этом произведенные изменения автоматически будут сообщены другому партнеру. Это автоматическое конфигурирование реализовано с использованием дополнительного механизма согласования опций, в котором каждый из партнеров сообщает свои возможности и требования. Примечательно, что в протоколе PPP один и тот же механизм согласования опций используется для протокола LCP и для семейства протоколов NCP.

Процедура информационного взаимодействия с использованием протокола PPP реализуется следующим образом.

- Для того чтобы организовать связь через канал связи с непосредственным соединением, партнер, инициирующий PPP соединение отправляет пакеты LCP для выбора конфигурации и (при необходимости) проверки качества канала передачи данных.
- После того как соединение установлено и проведено необходимое согласование параметров канала, инициирующий узел отправляет пакеты NCP для того, чтобы определить конфигурацию одного или нескольких протоколов сетевого уровня.
- Как только конфигурация каждого выбранного протокола определена, дейтаграммы сетевого уровня могут быть отправлены через данный канал.
- Канал сохраняет свою конфигурацию до тех пор, пока она не будет изменена протоколами LCP или NCP, или пока не произойдет какое-нибудь внешнее событие. Таким внешним событием может быть истечение допустимого интервала времени бездействия или вмешательство администратора системы.

Прежде чем по установленному соединению между узлами сети начнут передаваться IP-пакеты, протоколы LCP и IPCP должны, каждый на своем уровне, открыть канал. В информационное поле протокола PPP инкапсулируется только одна IP-дейтаграмма, а максимальная длина IP-пакета, который можно передавать через PPP-соединение, равна максимальной длине информационного поля пакета PPP. Если длина IP-дейтаграммы превышает эту величину — она должна быть подвергнута фрагментации.

Протокол PPP может быть использован на любом последовательном интерфейсе DTE/DCE (Data Terminal Equipment/Data Circuit Terminal Equipment), например, EIA RS-232C, EIA RS-422, EIA RS-423 и CCITT V.35. Протокол PPP не предъявляет каких-либо дополнительных ограничений, касающихся скорости передачи информации, кроме тех, которые определяются конкретным примененным интерфейсом DTE/DCE. Формат кадра протокола PPP показан на рис. 2.1.

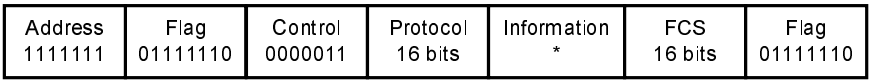


Рис. 2.1. Формат кадра протокола PPP

Flag (1 байт). Поле флага. Это поле указывает на начало или конец блока данных. Поле **Flag** содержит бинарную последовательность 01111110 (0x7e) и является разделителем кадров. Кадр PPP, состоящий из двух последовательных байтов 01111110, игнорируется.

Примечание

Для обеспечения кодовой прозрачности протокола PPP все комбинации из шести последовательных единиц, встречающиеся в управляющих и информационных полях передаваемого кадра PPP, заменяются на комбинацию бит — 01111101. Восстановление исходного вида кода производится после приема кадра при помощи процедуры обратной замены.

Address (1 байт). Поле адреса. Это поле является рудиментом HDLC и содержит бинарную последовательность 11111111, представляющую собой стандартный широковещательный адрес. Поскольку протокол PPP используется исключительно на линиях с двумя абонентами, присвоение им индивидуальных адресов канального уровня просто не имеет смысла.

Control (1 байт). Поле управления. Это поле содержит бинарную последовательность 00000011, которая требует от пользователя передачи информации непоследовательным кадром. Кадры с другим значением этого поля не обрабатываются.

Protocol (2 байта). Поле протокола. Его значение определяет тип протокола, использованного при формировании информационного поля данного кадра. Например, для протокола IP значение этого поля равно 0x0021 (33_{10}).

Information. Поле данных. Длина этого поля может составлять нуль или более байтов. В этом поле размещается блок данных протокола сетевого уровня, тип которого определен в поле **Protocol**. Граница поля данных определяется положением завершающей последовательности состоящей из полей **Flag** и **FCS**.

Установленное по умолчанию значение максимальной длины поля **Information** в кадре PPP равно 1500 байтов. Другие значения максимальной длины могут быть установлены при использовании стандартной процедуры согласования параметров соединения PPP.

FCS (2 байта). Поле контрольной суммы. Это поле содержит значение контрольной суммы всех полей кадра. При необходимости PPP-партнеры могут согласовать использование 32-битового (четырехбайтного) поля **FCS**, для того, чтобы улучшить процесс контроля ошибок.

Служебный протокол LCP

Как было выше отмечено, протокол LCP предназначен для обеспечения динамического согласования параметров канала протокола PPP. Протокол LCP представляет собой внутренний протокол PPP, поэтому блоки данных протокола LCP передаются в информационном поле полезной нагрузки кадра PPP. Только один блок данных протокола LCP одновременно может быть инкапсулирован в информационное поле кадра PPP, при этом поле **Protocol** кадра PPP должно иметь значение (0xc021) (Протокол управления каналом LCP).

На рис. 2.2 представлен формат блока данных протокола управления каналом. Поля передаются слева направо.

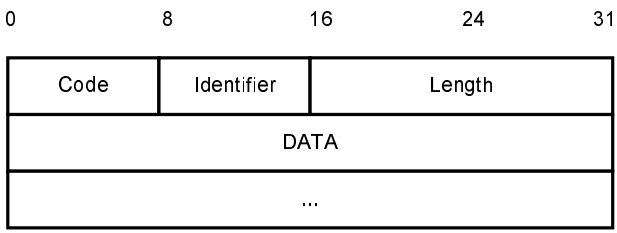


Рис 2.2. Формат блока данных протокола LCP

Поле **Code** (Код) занимает один байт в заголовке блока данных протокола LCP. Код, размещенный в данном поле, определяет тип сообщения LCP. В том случае, если партнер PPP принимает блок данных с неизвестным ему значением поля **Code**, он передает сообщение Code-Reject (Код отклонен), означающее, что переданный код не был принят. Перечень кодов сообщений протокола LCP и их назначения приведены в табл. 2.1.

Таблица 2.1. *Сообщения протокола LCP*

Код	Сообщение	Назначение
1	Configure-Request	Запрос выбора конфигурации
2	Configure-Ack	Выбор конфигурации подтвержден
3	Configure-Nak	Выбор конфигурации не подтвержден
4	Configure-Reject	Выбор конфигурации отклонен
5	Terminate-Request	Запрос завершения
6	Terminate-Ack	Подтверждение завершения
7	Code-Reject	Код отклонен

Таблица 2.1 (окончание)

Код	Сообщение	Назначение
8	Protocol-Reject	Протокол отклонен
9	Echo-Request	Запрос Эхо
10	Echo-Reply	Ответ Эхо
11	Discard-Request	Запрос отброшен

Поле **Identifier** (Идентификатор) занимает один байт в заголовке блока данных протокола LCP. В этом поле размещается уникальный идентификатор, формируемый источником PPP-сообщения. Анализ значения этого идентификатора помогает установить соответствие переданных запросов и полученных ответов. В том случае, если PPP-партнер принимает блок данных с недопустимым значением данного поля, он должен игнорировать этот блок данных.

Поле **Length** (Длина) занимает два байта в заголовке блока данных протокола LCP. Значение этого поля определяет длину блока данных протокола LCP, включая поля **Code**, **Identifier**, **Length** и **Data**. Значение **Length** не должно превышать максимальную величину принимаемого блока MRU (Maximum Receive Unit), установленную для данного канала. Байты, выходящие за диапазон, определенный полем **Length**, игнорируются на приеме. Если получен блок данных с недопустимым значением поля **Length**, этот блок данных игнорируется.

Поле **Data** (Данные) занимает нуль или более байтов, в строгом соответствии со значением поля длины. Формат и содержание поля **Data** определяется типом передаваемого сообщения.

Сообщение Configure-Request

Сообщение Configure-Request (Запрос выбора конфигурации) передается PPP-партнером для инициирования согласования изменения параметров канала с удаленным партнером. В поле опций помещается информация о желательных изменениях значений параметров канала, заданных по умолчанию. Опции конфигурации не должны включать значения, задаваемые по умолчанию.

В ответ на полученное сообщение запроса выбора конфигурации Configure-Request PPP-партнер должен передать соответствующее ответное сообщение.

Сообщение Configure-Ack

Сообщение Configure-Ack (Выбор конфигурации подтвержден) передается PPP-партнером в том случае, если все опции конфигурации, получен-

ные в запросе, идентифицированы, и их новые значения, предлагаемые инициирующей стороной, являются для него приемлемыми. В сообщение Configure-Ack отвечающая сторона помещает подтвержденные опции конфигурации, которые не должны быть переупорядочены или изменены каким-либо другим способом.

Содержимое поля **Identifier** принятого пакета выбора конфигурации Configure-Ack должно точно соответствовать содержимому поля **Identifier** последнего переданного запроса выбора конфигурации Configure-Request. Кроме того, опции конфигурации в Configure-Ack должны точно соответствовать перечню опций конфигурации последнего передаваемого запроса Configure-Request. Блоки данных несовпадающего формата должны быть проигнорированы протоколом LCP.

Сообщение Configure-Nak

В том случае, если все полученные опции конфигурации были распознаны, но некоторые из них неприемлемы для партнера, он должен сформировать сообщение Configure-Nak (Выбор конфигурации не подтвержден).

В поле опций этого сообщения должны быть помещены только несогласованные опции конфигурации принятого пакета Configure-Request. Все опции конфигурации, принятые PPP-партнером, исключаются из сообщения Configure-Nak. Обязательным условием является то, что принятые и не принятые опции конфигурации не должны быть переупорядочены в сообщении Configure-Nak.

В том случае, когда в запросе Configure-Request некоторая опция конфигурации перечислена более одного раза с различными значениями, пакет Configure-Nak должен включить список всех величин для этой опции, которые являются приемлемыми для отправителя Configure-Nak. В этот список включаются также приемлемые значения, которые были представлены в запросе выбора конфигурации.

Сообщение Configure-Reject

В том случае, если некоторые из опций конфигурации, полученных в запросе выбора конфигурации Configure-Request, не идентифицированы PPP-партнером или по требованию администратора сети не приемлемы для согласования, PPP партнер должен сформировать сообщение Configure-Reject (Выбор конфигурации отклонен). В поле опций этого сообщения вносятся только недопустимые опции конфигурации запроса выбора конфигурации. Все идентифицированные и согласуемые опции конфигурации должны быть исключены из поля опций пакета Configure-Reject. Однако, в любом случае, опции конфигурации, представленные в сообщении Configure-Reject, не должны быть переупорядочены или изменены.

Содержимое поля **Identifier** сообщения Configure-Reject должно точно соответствовать содержимому поля **Identifier** последнего полученного запроса выбора конфигурации.

Сообщения Terminate-Request и Terminate-Ack

Для экстренного завершения соединения протокол LCP использует сообщения Terminate-Request (Запрос завершения) и Terminate-Ack (Подтверждение завершения).

В том случае, если PPP-партнер желает закрыть соединение, он должен передать сообщение Terminate-Request. Передача пакетов Terminate-Request должна продолжаться до тех пор, пока от удаленной стороны не будет получено сообщение Terminate-Ack. Получение этого сообщения сигнализирует о том, что канальный уровень PPP-партнера перешел в пассивное состояние.

Передача сообщения Terminate-Request должна подтверждаться приемом сообщения Terminate-Ack. Прием непарного сообщения Terminate-Ack указывает на то, что PPP-партнер находится в состоянии Closed (закрыт) или Stopped (остановлен), или нуждается в том, чтобы провести повторное согласование параметров соединения.

Сообщение Code-Reject

Прием пакета LCP с неопознанным значением поля **Code** может указывать на то, что PPP-партнеры используют несовместимые версии протокола PPP. О возникновении подобной ситуации PPP-партнер должен сообщить отправителю неизвестного кода путем передачи сообщения Code-Reject (Код отклонен).

При получении сообщения Code-Reject на код, который является основным для данной версии протокола, драйвер протокола PPP должен передать информацию о возникшей проблеме протоколам сетевого и последующих уровней. В подобном случае PPP-соединение должно быть разорвано, поскольку маловероятно, что данная ситуация может быть исправлена автоматически.

Сообщение Protocol-Reject

Прием PPP пакета с неизвестным значением поля **Protocol** указывает на то, что PPP-партнеры пытаются использовать несовместимые версии протокола. Обычно это происходит в том случае, когда один из PPP-партнеров пытается применить новую версию протокола. В том случае, если протокол LCP находится в состоянии Opened (открыт), он должен сообщить о возникновении данной проблемы своему PPP-партнеру передачей пакета Protocol-Reject (Протокол отклонен).

При приеме сообщения Protocol-Reject, партнер должен прекратить передачу пакетов обозначенного протокола. Пакеты Protocol-Reject могут быть по-

сланы только в состоянии Opened LCP. Пакеты Protocol-Reject, принятые в каком-либо другом состоянии, LCP должны игнорироваться.

Сообщения Echo-Reply и Echo-Request

Сообщения Echo-Reply (Ответ Эхо) и Echo-Request (Запрос Эхо) используются для обеспечения проверки функционирования канала передачи данных. Эта проверка выполняется путем организации цифровой петли на канальном уровне. Диагностическая информация, полученная при помощи этого механизма, помогает локализовать неисправности при отладке, определении качества связи и при проверке пропускной способности канала.

Используемое в данном сообщении поле **Magic-Number** (волшебное число) предназначено для обнаружения каналов, находящихся в замкнутом состоянии. Пока опция конфигурации Magic-Number не будет согласована, значение этого поля должно быть равно 0.

Сообщение Discard-Request

Сообщение Discard-Request (Запрос отключения) применяется для того, чтобы сформировать запрос на освобождение канала. Данные сообщения используются при отладке, проверке эффективности функционирования протокола, а также для выполнения некоторых других функций.

Пакеты Discard-Request должны передаваться только в состоянии Opened протокола LCP.

Протоколы аутентификации PAP и CHAP

После согласования параметров PPP соединения с использованием протокола LCP при необходимости может быть использована процедура аутентификации абонента.

Определение

Процедура аутентификации представляет собой совокупность действий, выполняемых сторонами, участвующими в информационном обмене, для обеспечения взаимного опознавания и подтверждения своей подлинности.

Поскольку эта процедура не является обязательной, PPP-партнер должен явно указать на необходимость ее использования и определить тип протокола, который будет применен для установления подлинности партнера.

Для установления подлинности партнера могут быть использованы следующие стандартные протоколы:

- PAP (Password Authentication Protocol);
- CHAP (Challenge Handshake Authentication Protocol).

Протокол PAP предполагает применение простой схемы установления подлинности абонента и не может быть использован для реализации реальных систем обеспечения информационной безопасности, поскольку пароли этого протокола передаются в открытом виде, вследствие чего они легко могут быть перехвачены и воспроизведены нарушителем.

Протокол CHAP использует более сложную схему информационного взаимодействия, которая называется 3-этапным алгоритмом опознавания.

Первый этап алгоритма CHAP реализуется в тот момент, когда PPP-партнеры завершили стадию согласования параметров канала. В этот момент партнер, выполняющий процедуру аутентификации, передает специальное разведывательное сообщение протокола CHAP — challenge (обращение). В ответ на это сообщение его партнер должен передать ответное одноразовое кодовое сообщение, вычисленное с использованием известного для приемника алгоритма и кодового слова (пароля). В том случае, если это сообщение нормально дешифровано ответной стороной, процедура аутентификации будет успешно завершена. В противном случае PPP соединение должно быть разорвано.

Для обеспечения информационной безопасности повторение выполнения процедуры CHAP производится по истечению установленного временного интервала.

К несомненным достоинствам этого протокола следует отнести тот факт, что период повторного выполнения процедуры аутентификации после неудачной попытки определяется отвечающей стороной, что повышает уровень информационной безопасности и позволяет уменьшить нагрузку на сервер аутентификации.

Многоканальный протокол PPP

Многоканальный протокол PPP (MLP — Multilink PPP) представляет собой расширение стандартного протокола и обеспечивает возможность увеличения скорости передачи данных при одновременном использовании нескольких параллельных каналов. Первоначально предполагалось, что протокол MLP будет использоваться в сетях ISDN (Integrated Services Digital Network). Известно, что первичный PRI (Primary Rate Interface) и базовый BRI (Basic Rate Interface) интерфейсы ISDN с точки зрения пользователя представляются группой каналов, количество которых по его требованию может изменяться в некоторых пределах. Применение протокола MLP позволяет создавать виртуальный канал, объединяющий в себе несколько физических каналов. Группа физических каналов в данном случае называется связкой (bundle). Увеличение пропускной способности достигается за счет динамического перераспределения общего трафика виртуального канала между каналами связки. Для обеспечения параллельной передачи через группу каналов блоки данных сетевого уровня разбиваются на фрагменты,

количество которых определяется числом активных на текущий момент каналов в связке. Фрагменты получают сквозную нумерацию, что обеспечивает возможность восстановления исходного вида переданного блока данных сетевого уровня на приемной стороне. Дополнительным преимуществом, которое обеспечивается при использовании протокола MLP, является то, что все функции по созданию и обслуживанию виртуального канала выполняются исключительно на канальном уровне. Таким образом, протокол сетевого уровня имеет дело с единым виртуальным каналом, что существенно упрощает конфигурирование и обеспечивает экономию адресов сетевого уровня.

Организация связки каналов MLP

Возможность использования режима MLP протокола PPP определяется на этапе согласовании опции протокола LCP, в течение которого PPP-партнеры устанавливают возможность каждого из них объединять несколько соединяющих физических каналов в связку. В исключительных условиях для объединения двух партнеров может использоваться более одной соединяющей связки.

Возможность использования и параметры MLP согласуются при согласовании начальных опций протокола LCP. При этом локальный партнер сообщает удаленному PPP-партнеру о своем желании создать связку MLP путем посылки опции MLP в составе начальных опций протокола LCP.

Передача подобного сообщения означает следующее:

- партнер, предлагающий опцию MLP способен к объединению нескольких физических каналов в один логический канал;
- партнер способен получать фрагментированные блоки данных протокола верхнего уровня PDU (Protocol Data Unit) и собирать фрагменты обратно в исходные блоки PDU для последующей обработки;
- партнер способен получать блоки PDU размером N байтов, значение N согласуется партнерами с использованием стандартных процедур и в общем случае может превышать величину MRU, установленную на используемых физических каналах.

После успешного завершения согласования параметров протокола MLP обоими партнерами передающая сторона может начинать передачу инкапсулированных и/или фрагментированных блоков PDU с использованием заголовка MLP.

Примечание

Следует учесть, что непосредственно на связке MLP согласование параметров канала PPP не разрешено. Сообщения LCP Configure-Request, -Reject, -Ack, -Nak, Terminate-Request или –Ack не должны передаваться через связку MLP.

Структура фрагментов MLP

В соответствии с алгоритмом формирования кадров PPP блоки данных протокола сетевого уровня (например, пакеты протокола IP) инкапсулируются в кадры этого протокола. При использовании протокола MLP кадры PPP разбиваются на множество фрагментов, в соответствии с числом используемых физических каналов.

Каждым фрагмент при этом снабжается новым заголовком протокола PPP, состоящим из идентификатора протокола и заголовка MLP. Таким образом, первый фрагмент MLP PPP будет иметь два заголовка — заголовок фрагмента MLP и заголовок кадра PPP.

Выполнение процедуры MLP не требует обязательного выполнения фрагментации, например, небольших пакетов. Не существует также требований, говорящих о том, чтобы фрагменты имели равные размеры, или о том, что пакеты должны быть обязательно фрагментированы.

Более того, в качестве одной из возможных стратегий компенсации различий скоростей передачи в групповых каналах может использоваться разделение пакетов на фрагменты с длинами, пропорциональными скоростям передачи. Другой стратегией может быть разделение пакетов на равные фрагменты и установка коэффициентов относительного распределения фрагментов по каналам пропорциональных информационным скоростям каналов.

Протоколу MLP соответствует идентификатор протокола PPP имеющий значение 0x003d. Последующие четыре байта заголовка содержат трехбайтовый порядковый номер фрагмента и два признака того, что данный фрагмент содержит начало или конец блока данных сетевого уровня.

Для сокращения накладных расходов протокола MLP после согласования дополнительных опций PPP LCP четырехбайтовый заголовок может быть заменен укороченным заголовком, состоящим из двух байтов, позволяющим определить 12-разрядный последовательный номер фрагмента. Таким же образом может быть согласовано выполнение сжатия полей **Address**, **Control** и **PID**.

Структура фрагмента протокола MLP, имеющего длинный и короткий формат порядкового номера, представлены соответственно на рис. 2.3 и рис. 2.4.

Примечание

Идентификаторы PID (H) и PID (L) применяются здесь для обозначения старшего и младшего байта поля **Protocol** заголовка кадра протокола PPP, соответственно.

Бит стартового фрагмента (**B**) — одноразрядное поле, устанавливаемое в 1 у первого фрагмента, кадра PPP и 0 для всех остальных фрагментов этого же кадра.

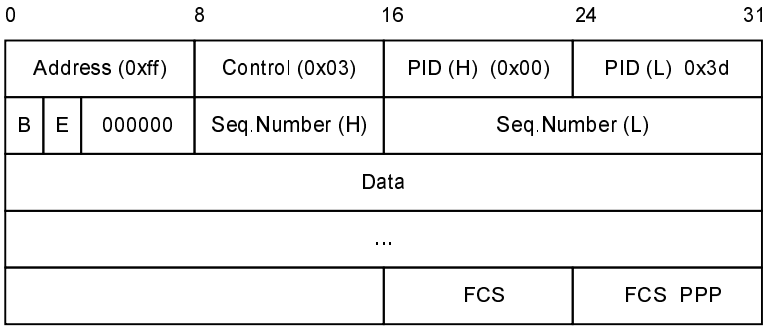


Рис. 2.3. Структура фрагмента протокола MLP с длинным форматом порядкового номера

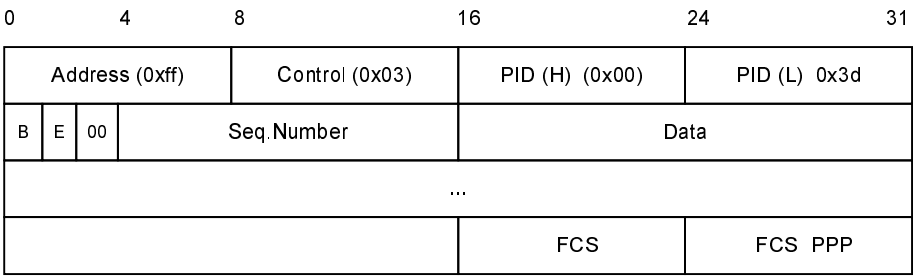


Рис. 2.4. Структура фрагмента протокола MLP с коротким форматом порядкового номера

Бит завершающего фрагмента (**E**) — одноразрядное поле, устанавливаемое в 1 у последнего фрагмента кадра PPP и 0 для всех остальных фрагментов этого же кадра.

Примечание

Фрагмент MLP, переносящий короткий кадр PPP, может содержать биты B и E установленными в 1.

В поле **Seq.Number** (порядковый номер) размещается 24- или 12-разрядное число, значение которого увеличивается на 1 после формирования очередного фрагмента. По умолчанию, поле последовательности имеет длину 24 бита, но с помощью описанной ниже опции конфигурации LCP может быть согласована 12-битная длина данного поля.

В заголовке фрагмента MLP между полем **E** и полем порядкового номера находится зарезервированное поле, использование которого в настоящее время не определено, которое должно быть установлено равным нулю. Длина зарезервированного поля составляет 2 бита в том случае, когда используются короткое поле порядкового номера, и 6 бит в противном случае.

Поле контрольной суммы кадра FCS формируется в соответствии с правилами формирования этого поля в кадре PPP. Протокол MLP не формирует дополнительных контрольных сумм для каждого из фрагментов исходного кадра PPP.

Обнаружение потери фрагмента MLP

Как уже было ранее отмечено, для обеспечения восстановления исходного кадра PPP протокол MLP снабжает последовательными номерами все фрагменты этого кадра, передаваемые через виртуальный канал. Порядковый номер устанавливается равным 0 и при организации связки и увеличивается на 1 после передачи очередного фрагмента, даже в том случае, если переданный фрагмент содержал весь кадр целиком.

По каждому из активных каналов связки PPP-партнер должен передать фрагменты со строго увеличивающимся порядковым номером. Выполнение этого требования обеспечивает функционирование на приемной стороне механизма обнаружения потери фрагментов, основанного на сравнении порядковых номеров.

Приемник просматривает порядковые номера фрагментов, приходящих по каждому из каналов связки, и хранит текущий минимум M среди порядковых номеров фрагментов, полученных по каналам связки. Обнаружение конца кадра происходит при получении фрагмента с установленным значением бита E. Сборка (восстановление, дефрагментация) исходного кадра может быть выполнена, если к этому моменту уже получены фрагменты, имеющие все последовательные предыдущие порядковые номера вплоть до последнего. В том случае, если значение M превышает порядковый номер фрагмента, несущего бит E, не собранного на текущий момент фрагментированного кадра PPP фиксируется потеря одного (как минимум) из фрагментов.

Таким образом, потеря фрагмента обнаруживается тогда, когда получены не все фрагменты, имеющие порядковые номера, значения которых больше номера фрагмента с битом B и меньше номера фрагмента с установленным битом E. Для обнаружения потери "крайних" фрагментов должны быть использованы дополнительные процедуры.

При нормальном функционировании каналов фрагмент с установленным битом B (первый) должен следовать непосредственно после фрагмента с установленным битом E (последним). Следовательно, при потере последнего фрагмента первым окажется фрагмент с номером M. В этом случае PPP-партнер должен уничтожить все фрагменты несобранного кадра и начать прием фрагментов нового.

Чтобы обеспечить скорейшее обнаружение потерянных фрагментов приемником, PPP-партнеры должны избегать наличия незадействованных групповых каналов в связке, поскольку у таких каналов значение M не увеличива-

ется. Для того чтобы избежать появления незадействованных каналов, PPP-партнеры должны перераспределять трафик между групповыми каналами связки даже при отсутствии трафика, превышающего пропускную способность одного канала.

Примечание

Потеря заключительного фрагмента может вызывать остановку приемника. Вероятность возникновения этой нежелательной ситуации может быть уменьшена путем посылки пустого фрагмента, состоящего только из заголовка MLP с установленными признаками первого и последнего фрагментов сразу после передачи последнего фрагмента.

Протоколы локальных вычислительных сетей

Локальные вычислительные сети, как правило, построены с использованием принципов разделения доступа к среде передачи данных. Использование этого принципа обеспечивает организацию информационного взаимодействия в режимах:

- Unicast (один — одному);
- Multicast (один — группе);
- Broadcast (один — всем).

Для обеспечения этих режимов взаимодействия на канальном уровне каждый объект, подключенный к локальной вычислительной сети должен обладать уникальным адресом канального уровня.

Адреса канального уровня, называемые также физическими, машинными, аппаратными или MAC-адресами, как правило, уникальны для каждого сетевого соединения. Адреса канального уровня компонентов современных локальных сетей размещаются в схеме интерфейса фирмой-производителем сетевого оборудования или назначаются той организацией, которая определяет стандарт протокола, представленный этим интерфейсом, т. е. "зашиты в железо" как, например, принято в сетях Ethernet.

MAC-адреса представляют собой так называемые "плоские" адреса. Часть этого адреса ничего вам не скажет о структуре сети, в которой работает машина с таким адресом. Так как для большинства компьютерных систем возможно только одно физическое сетевое соединение, они имеют только один адрес канального уровня. Маршрутизаторы и другие системы, соединенные с несколькими физическими сетями, могут иметь несколько адресов канального уровня.

Сети CSMA/CD (Ethernet)

Технология Ethernet была разработана Исследовательским Центром в Пало Альто (PARC) корпорации Xerox в 1970 году. Предполагалось, что Ethernet

должен был заполнить нишу между глобальными сетями, низкоскоростными сетями и специализированными сетями компьютерных центров, которые работали с высокой скоростью, но на очень ограниченном расстоянии.

На сегодняшний день Ethernet (IEEE 802.3) является наиболее распространенным протоколом построения локальных вычислительных сетей (ЛВС). Сегодня термин "Ethernet" используется для обозначения всех ЛВС, работающих по принципу множественного доступа с анализом несущей и обнаружением коллизий — CSMA/CD (Carrier Sense Multiple Access/Collision Detection).

Метод CSMA/CD предполагает, что все устройства взаимодействуют в одной среде и в каждый момент времени может передавать только одно устройство, а принимать могут все одновременно. Если два устройства пытаются передавать одновременно, то происходит столкновение передач, и оба устройства после случайного краткого периода ожидания пытаются вновь выполнить передачу.

Аналогией взаимодействиям в среде Ethernet может служить разговор группы вежливых людей в небольшой темной комнате. Каждый человек слышит речь других людей — анализ несущей. Все люди в комнате имеют одинаковые возможности вести разговор — множественный доступ, но никто не говорит слишком долго, так как все вежливы. Все молчат, пока кто-то говорит. Если два человека начинают говорить одновременно, то они сразу обнаруживают это, поскольку слышат друг друга — обнаружение коллизий. В этом случае они замолкают и ждут некоторое время, после чего один из них вновь начинает разговор. Другие люди слышат, что ведется разговор, и ждут, пока он закончится, а затем могут начать говорить сами. Каждый человек имеет собственное имя — аналог уникального Ethernet-адреса. Каждый раз, когда кто-нибудь начинает говорить, он называет по имени того, к кому обращается, и свое имя, например, "Гена, это Иван, идем пить кофе". Если кто-то хочет обратиться ко всем, то он говорит: "Слушайте все, это Ларри, кончай работу !" — широковещательная передача.

Каждый адаптер, предназначенный для подключения к сети Ethernet, имеет уникальный MAC-адрес. Как было отмечено выше, MAC-адрес присваивается сетевому устройству на заводе изготовителе. Этот адрес имеет длину 48 бит, что вполне достаточно для обеспечения его глобальной уникальности.

Блок данных канального уровня — кадр Ethernet имеет 14-байтный заголовок, в котором размещаются MAC-адреса отправителя и получателя, поле **Type/Code**, данные и поле контрольной суммы кадра (рис. 2.5). Каждая машина в Ethernet-сети обращает внимание только на те кадры, которые адресованы ей лично, например 80:12:CB:34:56:45, либо адресованы всем машинам сети — адрес назначения кадра содержит широковещательный адрес FF:FF:FF:FF:FF:FF, состоящий из единиц во всех 48 разрядах.

Примечание

Напомним, что Ethernet-адреса сами по себе не имеют никакого отношения к IP-адресам, и для того, чтобы зафиксировать, что данная машина с таким-то Ethernet-адресом имеет такой-то IP-адрес, необходимо построить дополнительное соответствие.

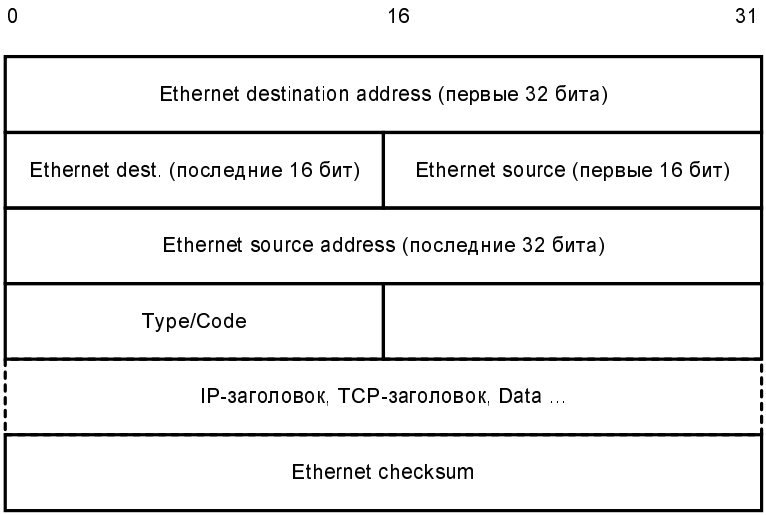


Рис. 2.5. Формат кадра Etnernet

Ethernet destination address. Адрес получателя.

Ethernet source address. Адрес отправителя.

Некоторые Ethernet-адреса предназначены для использования в качестве групповых multicast-адресов. Групповые Ethernet-адреса занимают область адресов с 01-00-5E-00-00-00 по 01-00-5E-7F-FF-FF.

Групповая адресация Ethernet — это, пожалуй, самый естественный способ построения multicast-IP-адресации при реализации маршрутизации на уровне локальных сетей.

IP-адрес группы узлов ассоциируется, с Ethernet multicast-адресом следующим образом: 23 младших бита IP-адреса группы помещаются в 23 младших бита группового Ethernet-адреса.

Например, IP-адрес 224.1.10.111 соотносится с адресом 01-00-5E-01-0A-6F. Поскольку у группового IP-адреса значащими являются 28 бит, одному multicast Ethernet-адресу можно поставить в соответствие 31 IP-группу.

Type/Code. Поле типа Ethernet-кадра. Это поле позволяет использовать в одной сети Ethernet одновременно несколько протоколов сетевого уровня (IP, IPX и т. д.). Каждому из этих протоколов соответствует свое значение кода поля. Например, для протокола IP значение этого поля равно 0800.

Ethernet Checksum. Поле контрольной суммы данных всего кадра. Это поле позволяет проверять правильность передачи данных через физический канал. Если контрольная сумма полученного кадра не совпадает с вновь вычисленной контрольной суммой данных — кадр уничтожается.

Максимальная длина кадра, передаваемая по сети Ethernet, составляет немногим более 1500 байт. Отсутствие этого ограничения привело бы к увеличению неравномерности доступа к среде передачи. Минимальная длина кадра Ethernet также ограничена значением 64 байта для обеспечения достаточных физических размеров сети.

Примечание

Минимальная длина кадра некоторых современных высокоскоростных технологий построения локальных вычислительных сетей-преемников Ethernet существенно превышает 64 байта.

Сети с маркерным доступом

В последующих разделах данной главы будет приведено краткое описание схемы взаимодействия протоколов IP и Ethernet, механизм разрешения адресов и правила построения работы IP через локальную сеть Ethernet. В настоящем разделе будут кратко рассмотрены механизмы взаимодействия протокола IP с другим типом локальных сетей, которые по популярности стоят на втором месте после сетей Ethernet — это сети Token Ring и FDDI (Fiber Distributed Data Interface).

По принципу построения сети Token Ring и FDDI относятся к сетям с передачей маркера, называемым также *сетями с маркерным доступом*. Компоненты этих сетей по очереди передают по кольцевой сети небольшой блок данных, называемый *маркером*. Владение этим маркером гарантирует право передачи. Если узел, принимающий маркер, не имеет информации для отправки, он просто переправляет маркер следующему узлу по кольцу (ring). Каждый узел может удерживать маркер в течение ограниченного интервала времени.

Если у станции, владеющей маркером, имеется информация для передачи, она захватывает маркер, изменяет у него один бит (в результате чего маркер становится началом блока данных), дополняет его информацией, которую он хочет передать, и отсылает эту информацию к следующей станции кольцевой сети.

Когда информационный блок перемещается по кольцу, маркер в сети отсутствует, поэтому другие станции, желающие передать информацию, вынуждены ожидать. Это делает невозможным возникновение коллизий в сетях с маркерным доступом. Информационный блок перемещается по кольцу, пока не достигнет предполагаемой станции назначения, которая копирует из него информацию себе в буфер для дальнейшей обработки.

Информационный блок продолжает тем временем свой путь по кольцу; он окончательно удаляется только после достижения станции, отославшей этот блок. При этом станция может проверить вернувшийся блок, чтобы убедиться, что он был просмотрен и скопирован станцией назначения. Таким образом, в сетях с маркерным доступом на канальном уровне могут быть реализованы процедуры обеспечения гарантированной доставки.

В отличие от сетей CSMA/CD (например, Ethernet), сети с передачей маркера являются детерминистическими сетями. Это означает, что можно вычислить максимальное время, которое пройдет, прежде чем любая конечная станция сможет передать свои данные. Эта особенность, а также некоторые характеристики надежности, которые будут рассмотрены дальше, делают сети Token Ring и FDDI идеальными для применения в тех системах, где задержка должна быть предсказуема и важна устойчивость функционирования сети — например, на автоматизированных конвейерах заводов.

Сети Token Ring

В сети Token Ring используются два типа кадров: кадр маркера и кадры данных/команд. Структура кадра данных/команд Token Ring приведена на рис. 2.6.

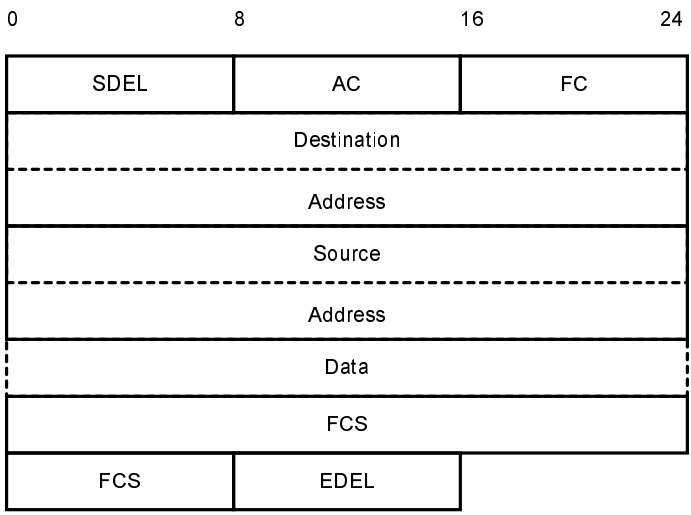


Рис. 2.6. Кадр данных/команд Token Ring

SDEL (Start Delimiter) (1 байт). Ограничитель начала. Служит для предупреждения каждой станции о прибытии маркера (или блока данных / блока команд).

AC (Access Control) (1 байт). Байт управления доступом. Содержит поля приоритета и резервирования, а также бит маркера, используемый для диф-

ференциации маркера и блока данных/блока команд. Кроме того, AC содержит бит монитора, используемый активным монитором, чтобы определить, циркулирует какой-либо блок в кольце непрерывно или нет.

Примечание

Функция станции, выбранной активным монитором, заключается в том, чтобы удалять из кольца Token Ring кадры, не удаленные по какой-то причине станциями-отправителями.

EDEL (End Delimiter) (1 байт). Ограничитель конца. Сигнализирует о конце маркера или блока данных / блока команд. В нем также имеются биты для индикации поврежденного блока, а также блока, являющегося последним в логической последовательности.

FC (Frame Control) (1 байт). Поле управления кадром. В этом поле помещается признак типа кадра — управляющий или информационный.

Следом за полем **FC** в кадре Token Ring размещены два адресных поля, которые идентифицируют станции пункта назначения и источника. Длина адресов равна 6 байтам.

В Token Ring так же, как и в Ethernet, используются 48-разрядные (6-байтные) универсальные адреса. Однако форматы адресов источника и получателя имеют ряд отличий по отношению к структуре кадра Ethernet. На рис. 2.7 приведен формат адреса получателя, где отмечены особенно важные разряды адреса. Различные комбинации этих разрядов определяют отличные друг от друга типы адресов получателя.

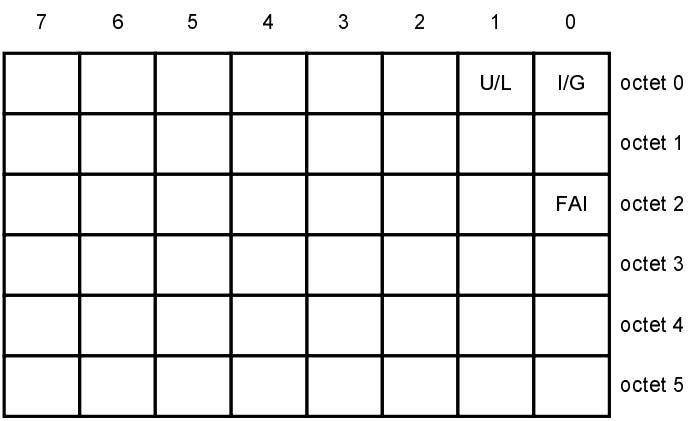


Рис. 2.7. Формат адреса Token Ring
(кадр передается справа налево, сверху вниз)

Адрес получателя соответствует какой-либо конкретной станции либо некоторой группе устройств. Если значение бита **I/G** (Individual/Group) равно 1,

адрес получателя будет групповым адресом, т. е. адресованным какой-либо группе устройств. В Token Ring можно задавать универсальный широковещательный адрес FF-FF-FF-FF-FF-FF. Такой адрес используется для рассылки данных пользователями через мосты и другие кольца сети. В сетях Token Ring может использоваться и другой тип широковещательного адреса: C0-00-FF-FF-FF-FF. Этот адрес является широковещательным адресом управляющих сообщений и перемещается только внутри одного кольца.

Адрес, назначенный производителем адаптера и записанный в ПЗУ, называется универсальным адресом. Такому адресу соответствует значение 0 бита **U/L** (Universal/Local). Если вы решили подавить универсальный адрес на плате, задав свой собственный (по соглашению), этот бит необходимо установить в 1.

Бит **FAI** (Functional Address Indicator) используется для указания принадлежности адреса к классу функциональных адресов. Если бит **FAI** установлен в 0, это означает, что устройству присвоен функциональный адрес. В этом случае функциональный адрес определяется значениями младших разрядов адреса Token Ring (с 0 по 30).

Для соотнесения адресов IP и адресов Token Ring используется протокол ARP (Address Resolution Protocol).

Для передачи группового трафика IP в сетях Token Ring существует три способа:

- широковещательное сообщение по всем кольцам сети;
- назначение функциональных адресов приемникам группового трафика;
- назначение групповых адресов приемникам группового трафика.

Но, как правило, в сетях Token Ring преимущество отдается второму способу, т. е. группа IP ассоциируется с одним функциональным адресом.

Непосредственно после адресных полей в кадре Token Ring размещается поле данных. Размер этого поля ограничен максимальным временем удержания маркера кольца.

Поле данных составляют команды управления доступом к среде LAN — кадры команд или данные пользователя — кадры данных. Кадры данных циркулируют исключительно внутри одного кольца и не передаются через маршрутизаторы или мосты в другие кольца. Данные пользователя могут включать в себя информацию о маршруте или дейтаграмму протокола более высокого уровня, например, IP.

За полем данных в кадре Token Ring расположено поле контрольной суммы кадра FCS (Frame Check Sequence). Станция-отправитель заполняет это поле вычисленной величиной, зависящей от содержания блока данных. Станция назначения повторно вычисляет эту величину, чтобы определить, не был ли блок поврежден при прохождении. Если блок был поврежден — он отбрасывается.

Примечание

Кадр маркера кольца состоит только из байтов SDEL, AC и EDEL.

Сети FDDI

Стандарт FDDI появился достаточно давно, в середине 80-х. Хотя технология FDDI во многом похожа на Token Ring, FDDI она изначально была предназначена для обеспечения более высоких скоростей передачи данных. Обе эти технологии имеют похожие характеристики, включая топологию (кольцевая сеть), технику доступа к носителю (передача маркера), характеристики надежности и др.

Одной из наиболее важных особенностей технологии FDDI является то, что, она ориентирована на использование в качестве передающей среды волоконно-оптического кабеля (ВОК). Применение волоконно-оптического кабеля обеспечивает ряд преимуществ по сравнению с традиционной медной проводкой:

- защита данных, поскольку ВОК не излучает электрические сигналы, которые можно перехватывать;
- надежность, так как ВОК устойчив к электрическим помехам;
- скорость — потенциальная пропускная способность ВОК, намного выше, чем у медного кабеля.

Благодаря применению оптоволоконной технологии, стандарт FDDI обеспечивает передачу данных со скоростью 100 Мбит/с на значительные расстояния.

Протокол FDDI отличается от Token Ring большей надежностью, за счет использования технологии двойного кольца. Кроме того, отличается механизмами управления распределением полосы пропускания в режиме реального времени, поскольку FDDI обеспечивает это путем обозначения двух типов трафика: синхронного и асинхронного.

Синхронный трафик может использовать выделенную часть общей полосы пропускания сети FDDI; остальную часть может потреблять асинхронный трафик. Синхронная полоса пропускания выделяется тем станциям, которым необходима постоянная возможность передачи. Например, наличие такой возможности помогает при передаче голоса и видеоинформации. Другие станции используют остальную часть полосы пропускания асинхронно.

Форматы блока данных FDDI аналогичны форматам Token Ring за некоторыми исключениями.

1. Перед каждым кадром FDDI передаются 2 байта преамбулы (preamble), которая используется для обеспечения надежной взаимной синхронизации генераторов источника и приемника данных.

2. Завершает каждый кадр FDDI 2 байта статуса, которые позволяют станции источника определять, не появилась ли ошибка и был ли блок данных признан и скопирован принимающей станцией.
3. Расширилась спецификация управляющего поля FC блоком данных. Оно может содержать информацию о размере адресных полей, вида данных, содержащихся в блоке (синхронная или асинхронная информация), и другую управляющую информацию.

Протокол FDDI также поддерживает механизм сопоставления IP-адресов и адресов узлов локальной сети с использованием протокола ARP. Механизм использования широковещательных и групповых адресов локальной сети аналогичен подобным механизмам, применяемым в технологии Ethernet, с учетом специфики архитектуры сети. Так, широковещательному IP-адресу (оповещение всех узлов в данной сети) соответствует FDDI-адрес FF-FF-FF-FF-FF-FF, а, например, групповому адресу 224.255.0.2 будет соответствовать групповой FDDI-адрес 01-00-5E-7F-00-02.

Протокол ARP

В этом разделе мы рассмотрим, каким образом при посылке IP-кадра определяется Ethernet-адрес узла назначения, и как по Ethernet-адресу кадра определяется IP-адрес узла.

Особенность сети Интернет заключается в том, что IP-адреса и Ethernet-адреса выбираются независимо, и, следовательно, не существует алгоритма для однозначного преобразования одного из них в другой.

IP-адрес определяется менеджером сети с учетом положения машины в сети Интернет. Если машину перемещают в другую часть сети Интернет, то ее IP-адрес должен быть изменен.

Ethernet-адрес определяется производителем сетевого интерфейсного оборудования из выделенного для него по лицензии адресного пространства. Когда у машины заменяется плата сетевого адаптера, то меняется и ее Ethernet-адрес.

Примечание

В некоторых сетях (например, IPX/SPX Nowell Netware) адрес канального уровня является частью адреса сетевого уровня. В таких случаях специальный протокол для сопоставления канального и физического адресов просто не нужен.

Для отображения IP-адресов в Ethernet-адреса используется протокол ARP (Address Resolution Protocol, Протокол разрешения адресов). Это отображение выполняется только для отправляемых IP-пакетов, так как только в момент отправки создаются заголовки IP и Ethernet. Для обратного отображения Ethernet-адресов в IP-адреса используется протокол RARP (Reverse Address Resolution Protocol, Протокол обратного разрешения адресов).

Прежде всего, отметим, что ни ARP-, ни RARP-протоколы не используют сервисы протокола IP. Они являются самостоятельными и независимыми друг от друга протоколами канального уровня.

Основным инструментом работы протокола ARP является таблица разрешения адресов. Прямое преобразование адресов выполняется путем поиска подходящей записи в этой таблице. Эта таблица, называемая ARP-таблицей, хранится в памяти каждого узла сети и содержит записи о взаимном соответствии адресов канального и физического уровней. В двух столбцах таблицы содержатся IP- и Ethernet-адреса узлов. Если для приложения требуется по IP-адресу узла определить его Ethernet-адрес, ищется запись с соответствующим IP-адресом, по которой и определяется требуемый Ethernet-адрес узла. Пример соответствия адресов сетевого уровня и канального уровня приведен в табл. 2.2.

Таблица 2.2. *Соответствие адресов сетевого и канального уровней*

Адрес сетевого уровня	Адрес канального уровня (Ethernet-адрес)
220.20.2.1	08:00:39:00:2F:C3
220.20.2.3	08:00:5A:21:AB:22
220.20.2.4	08:00:10:91:AC:54

ARP-таблица заполняется автоматически модулем ARP по мере необходимости. Когда с помощью существующей ARP-таблицы не удается преобразовать IP-адрес, выполняются следующие действия.

1. По сети передается широковещательный запрос протокола ARP.
2. Исходящий IP-кадр ставится в очередь.
3. Каждый сетевой адаптер принимает широковещательный кадр ARP-запроса.
4. Драйверы протокола Ethernet проверяют поле типа в принятом Ethernet-кадре и передают ARP-кадры модулю ARP.

ARP-запрос может быть интерпретирован следующим образом: "Если ваш IP-адрес совпадает с указанным в этом запросе, то сообщите мне ваш Ethernet-адрес". Кадр ARP-запроса содержит Ethernet и IP-адреса отправителя, IP-адрес узла, Ethernet-адрес которого необходимо определить, и поле типа запроса, указывающее, что необходимо выяснить именно Ethernet-адрес по данному IP-адресу.

Каждый модуль ARP проверяет поле искомого IP-адреса в полученном ARP-кадре и, если адрес совпадает с его собственным IP-адресом, посылает ответ прямо по Ethernet-адресу отправителя запроса. ARP-ответ можно интерпретировать так: "Да, это мой IP-адрес, ему соответствует такой-то

Ethernet-адрес". Кадр с ARP-ответом в этом случае будет содержать IP- и Ethernet-адреса как отправителя, так и получателя — составителя запроса. После этого данный кадр передается ARP-модулю, который дополняет ARP-таблицу новой записью.

Таким образом, новая запись в ARP-таблице появляется автоматически спустя всего несколько миллисекунд после того, как она потребовалась. Ранее, на шаге 2, исходящий IP-кадр был поставлен в очередь. Теперь, с использованием обновленной ARP-таблицы, выполняется преобразование IP-адреса в Ethernet-адрес, после чего кадр Ethernet передается по сети. Следующие шаги выглядят так.

1. Обладатель IP-адреса формирует ARP-ответ, содержащий информацию о соответствии IP- и Ethernet-адреса. Эта информация заносится в ARP-таблицу станции, сформировавшей запрос.
2. Для преобразования IP-адреса в Ethernet-адрес у IP-кадра, поставленного в очередь, используется ARP-таблица.
3. Ethernet-кадр передается по сети Ethernet.

Если в сети нет машины с искомым IP-адресом, то ARP-ответ не будет сформирован, следовательно, в ARP-таблице не появится соответствующая запись. В этом случае протокол IP будет уничтожать IP-кадры, направляемые по несуществующему адресу.

Примечание

Протоколы верхнего уровня не могут отличить случай повреждения сети Ethernet от случая отсутствия машины с искомым IP-адресом.

Некоторые реализации протоколов IP и ARP не ставят в очередь IP-кадры на то время, пока они ждут ARP-ответов. Вместо этого IP-кадр просто уничтожается, а его восстановление возлагается на модуль TCP или прикладной процесс, работающий через UDP. Такое восстановление выполняется с помощью тайм-аутов и повторных передач. Повторная передача сообщения проходит успешно, так как первая попытка уже вызвала заполнение ARP-таблицы.

Следует отметить, что узел сети должен иметь отдельную ARP-таблицу для каждого из своих сетевых интерфейсов. Формат ARP-кадра приведен на рис. 2.8.

В поле **Hardware Type** помещается признак типа используемого протокола канального уровня. Протоколу Ethernet соответствует значение 1 в данном поле.

В поле **Protocol Type** помещается признак типа используемого протокола сетевого уровня. Протоколу IP соответствует значение 0800 в данном поле. Содержимое полей **Length**, **HLEN** (Hadware Length) и **PLEN** (Protocol определяет размер адреса канального и сетевого уровня соответственно. Наличие данных полей обеспечивает возможность использования протокола ARP для определения физического адреса в различных сетях второго и третьего

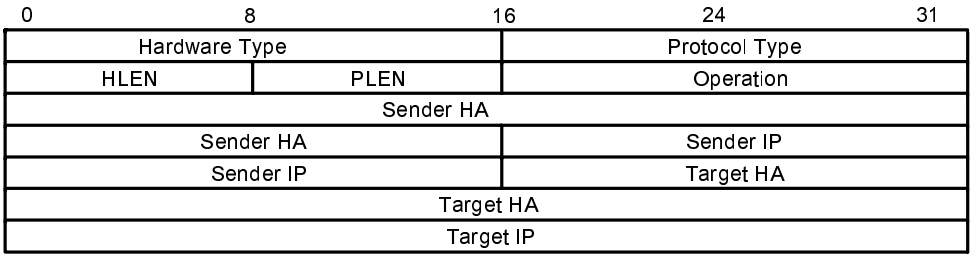


Рис. 2.8. Формат ARP-кадра

уровня. В поле **Operation** размещается признак типа информационного кадра ARP. В табл. 2.3 приведены коды и назначения сообщений протоколов ARP и RARP.

Таблица 2.3. *Коды и назначения сообщений протоколов ARP и RARP*

Код	Значение	Назначение сообщения
1	ARP Request	Запрос протокола ARP
2	ARP Response	Ответ протокола ARP
3	RARP Request	Запрос протокола RARP
4	RARP Response	Ответ протокола RARP

В полях **Target IP** и **Sender IP** кадров ARP и RARP размещаются сетевые адреса станции назначения и передающей станции, соответственно. Обычно в современных сетях в качестве логических адресов фигурируют 32-разрядные адреса IP, однако протоколы ARP и RARP могут быть использованы и для обслуживания других протоколов сетевого уровня. Такая возможность, в частности обеспечивается использованием гибкой структуры кадра ARP/RARP. В зависимости от типа протокола, информация, размещаемая в полях **Target IP** и **Sender IP,** может принимать различное значение. Например, в поле **Target IP** кадра типа ARP Request передающая станция помещает сетевой адрес узла, для которого она хочет определить адрес канального уровня. В полях **Target HA** и **Sender HA** кадров ARP и RARP размещаются физические адреса станции назначения и передающей станции, соответственно. Обычно в современных сетях в качестве физического адреса фигурируют 48-разрядные MAC-адреса. В зависимости от типа протокола (ARP или RARP) информация, которая размещается в данных полях, может принимать различное значение. В кадре типа ARP Response, который формируется в ответ на кадр ARP Request, в поле **Target HA** отвечающая станция помещает собственный MAC-адрес.

ARP cache

Для промежуточного хранения результатов определения физического адреса может быть использовано аппаратно- или программно-организованное буферное оперативное запоминающее устройство (ОЗУ). Каждой записи, которая заносится в данное ОЗУ, присваивается значение относительного момента времени, который соответствовал ее формированию. Таким образом, запись в ARP cache содержит следующие поля:

- сетевой адрес;
- физический адрес;
- возраст записи.

Возраст записи учитывается для того, чтобы обеспечить возможность повторного выполнения процедуры ARP при изменении каких-либо параметров установленного соответствия. Важно также отметить, что при формировании ARP-запроса станция должна сохранять копию данного запроса в специальной очереди для того, чтобы установить соответствие полученного ответа реальному состоянию сети в текущий момент. Для того чтобы обеспечить возможность организации информационного взаимодействия на сетевом уровне с абонентом, который по каким-либо причинам не может самостоятельно ответить на ARP-запрос, используется специальная функция, которая называется Proxy ARP.

Proxy ARP

Функция Proxy ARP может быть использована тогда, когда абонент данной сети не может ответить на ARP-запрос, потому что не поддерживает протокол ARP. Кроме того, запрашиваемый абонент может находиться за пределами данной сети и, следовательно, данный запрос до него просто не дойдет. В подобных случаях формирование ответов на запросы выполняется одной из станций данной сети, которая поддерживает функцию Proxy ARP. По умолчанию данная функция реализуется маршрутизаторами.

Примечание

Протокол ARP может быть использован для построения закрытых фрагментов локальных сетей. В качестве устройства, которое используется для разделения открытого и закрытого фрагментов локальной сети, в данном случае может быть применен двухпортовый сервер Proxy ARP.

Утилита ARP MS Windows и способы ее использования

С помощью утилиты ARP пользователь или администратор системы может формировать содержание буферного ОЗУ ARP cache или проверять его содержимое. Необходимость в подобных операциях может возникнуть в процессе поиска неисправности в сегменте локальной сети. Формирование ста-

тических записей в таблице ARP необходимо в том случае, если один или несколько узлов сети не поддерживают протокол ARP и одновременно отсутствует возможность включения функции Proxy ARP в этой сети. В ряде случаев применение статического принципа формирования ARP cache для некоторых узлов сети может повысить общий уровень информационной безопасности. В то же время следует учесть, что неправильное формирование таблицы ARP для некоторого узла может привести к полному нарушению его информационного взаимодействия с остальными компонентами сети. В табл. 2.4 приведен перечень режимов использования утилиты ARP.

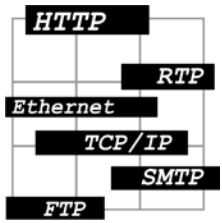
Таблица 2.4. *Перечень режимов использования утилиты ARP*

Ключ	Режим
-a	Отображение текущего содержимого ARP cache
-s	Формирование статической записи в ARP cache
-d	Удаление записи из ARP cache

Примечание

При использовании утилиты ARP MS Windows следует учитывать, что максимальное значение времени жизни динамической записи в ARP cache обычно составляет от 5 до 15 минут.

Глава 3



Протоколы управления и контроля сетевых компонентов

Классы и задачи служебных протоколов

В этой главе будут рассмотрены служебные протоколы сети Интернет, предназначенные для обеспечения и поддержания ее работоспособности. Служебные протоколы решают задачи сбора и предоставления информации о состоянии компонентов сети, сигнализируют о возникновении аварийных ситуаций, одним словом представляют собой комплекс разного рода инструментов, облегчающих администрирование и обслуживание сети. Функции служебных протоколов можно разделить на две большие группы:

- функции мониторинга состояния сети;
- функции управления компонентами сети.

Большинство служебных протоколов Интернета сочетает в себе функции обеих этих групп.

К задачам протоколов первой группы относятся проверки выбранных параметров сетевых устройств. В зависимости от назначения протокола, эти проверки могут производиться постоянно и автоматически или по требованию администратора.

Управление сетевыми компонентами предполагает изменение их параметров и конфигураций в процессе функционирования. Эти функции в той или иной степени свойственны практически всем служебным протоколам Интернета.

Протоколы мониторинга состояния сети

Функции мониторинга в большей степени свойственны протоколу SNMP, который не только досконально анализирует состояние любого устройства подключенного к сети, но и позволяет самому устройству формировать и направлять сигналы об изменении своего состояния. Функция мониторинга может быть выполнена протоколами RPC (Remote Procedure Call, Вызов удаленной процедуры), TELNET (удаленный терминал) и ICMP (Internet Control

Message Protocol). Протоколы RPC и TELNET относятся к протоколам прикладного уровня и обеспечивают контроль состояния сетевых компонентов в интерактивном режиме с оператором. Протокол RPC, в частности, позволяет администратору сети выполнить процедуру на удаленном узле, а протокол TELNET обеспечивает возможность управления сетевым объектом через эмулируемый виртуальный терминал. Протоколы ICMP и SNMP оперируют на сетевом уровне и встраиваются в приложения (утилиты операционных систем, управляющие платформы). Протокол сетевого обслуживания SNMP обладает практически неограниченными возможностями мониторинга состояний сложных иерархически организованных объектов.

Протоколы управления компонентами сети

Функция управления, исполняемая служебными протоколами Интернета, является более сложной и многогранной, чем описанная выше функция мониторинга. Специфика этой функции заключается в том, что выполнение управляющих действий должно быть проведено в наиболее короткие сроки после установления необходимости этих действий. Вследствие этого выполнение функций управления в большинстве случаев не требует участия оператора и максимально автоматизировано. Это определяется требованием обеспечения постоянного функционирования компонента сети и сокращения времени его простоя из-за неисправности (протокол горячего резервирования маршрутизаторов Cisco Systems HSRP (Hot Standby Router Protocol)) или уменьшения затрат на администрирование этого узла (протокол динамического конфигурирования DHCP (Dynamic Host Configuration Protocol)). При необходимости администратор также может принимать участие в процессе управления сетевыми компонентами в режиме диалога с использованием протоколов RPC и TELNET.

Протокол ICMP

Основной особенностью сети Internet является распределенный характер управления. Узлы этой сети выполняют те или иные действия по передаче дейтаграмм, основываясь на своем понимании о структуре сети. Единый элемент, который координировал бы эти действия, в данном случае отсутствует. Поэтому для обеспечения исправного функционирования компоненты сети Интернет должны обмениваться служебной или диагностической информацией. Необходимость в таком информационном обмене может возникнуть в том случае, если узел не смог определить направление дальнейшей передачи дейтаграммы или должен сообщить об истечении времени ее жизни.

Для обеспечения этого информационного обмена в сети IP предназначен протокол управляющих сообщений ICMP (Internet Control Message Protocol).

Протокол ICMP предназначен для формирования управляющих и диагностических сообщений, обеспечивающих реализацию дополнительных функций управления сетевыми компонентами Интернета или локализацию неисправных компонентов этой сети. На протокол ICMP, однако, не возлагаются функции устранения причин возникновения этих ошибок, его задачей является только передача сообщения о возникновении ошибки при передаче дейтаграммы в адрес ее источника. Обеспечение адекватной реакции на возникновение неисправности возлагается на источник дейтаграммы.

Для передачи сообщений протокола ICMP по сети IP используются дейтаграммы обычного формата. Собственно сообщение ICMP помещается в поле **DATA** (см. рис. 1.5). Заголовок дейтаграммы, предназначенной для переноса сообщений ICMP, имеет следующие значения полей:

- **SERVICE TYPE** = 0;
- **PROTOCOL** = 1 (Протокол ICMP);
- **TIME TO LIVE** — устанавливается в соответствии с типом сообщения в секундах;
- **SOURCE IP ADDRESS** — адрес источника сообщения ICMP;
- **DESTINATION IP ADDRESS** — адрес станции назначения для данного сообщения ICMP.

Структура сообщения ICMP

Сообщение ICMP состоит из заголовка сообщения и собственно сообщения. Заголовок сообщения ICMP может занимать до 8 байтов — два 32-разрядных слова, его размер определяется типом сообщения. В заголовке сообщения размещается идентификатор типа сообщения ICMP. В табл. 3.1 приведены значения поля **TYPE** и типы сообщений протокола ICMP, которые соответствуют этим значениям.

Сообщения ICMP можно условно разделить на парные и непарные. Парные сообщения состоят из двух компонентов — запрос (Request) и ответ (Reply). Сообщение типа ответ высылается станцией назначения в ответ на полученное от источника сообщение типа запрос. К таким сообщениям относятся, например, сообщения Echo Request/Reply. Непарные сообщения формируются асинхронно при возникновении какой либо проблемы при передаче дейтаграммы и передаются в адрес источника этой дейтаграммы. К сообщениям подобного типа относятся, например, сообщения Destination Unreachable и Source Quench. Полный перечень сообщений протокола ICMP приведен в табл. 3.1 (значение поля **TYPE** представлено в десятичной системе исчисления).

Таблица 3.1. *Сообщения протокола ICMP*

TYPE	Сообщение	Назначение
0	Echo Reply	Ответ ICMP — эхо
3	Destination Unreachable	Цель недостижима
4	Source Quench	Сдерживание источника
5	Redirect	Перенаправление маршрута
8	Echo Request	Запрос ICMP — эхо
11	Time Exceeded	Время жизни истекло
12	Parameter Problem	Неверный параметр
13	Timestamp Request	Запрос метки времени
14	Timestamp Reply	Ответ метки времени
15	Information Request	Запрос информации
16	Information Reply	Ответ информации
17	Address Mask Request	Запрос маски адреса
18	Address Mask Reply	Ответ маски адреса

Структура заголовка сообщений ICMP

Заголовки всех сообщений ICMP имеют примерно одинаковый формат. В четырех первых байтах заголовка сообщений ICMP размещаются поля **TYPE**, **CODE** и **CHECKSUM**. Структура заголовка сообщения ICMP приведена на рис. 3.1.

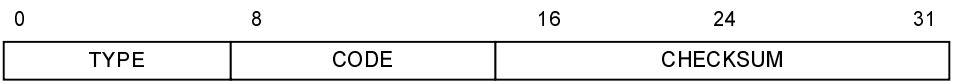


Рис. 3.1. Структура заголовка сообщения протокола ICMP

В поле **TYPE** заголовка сообщений ICMP размещается код, определяющий тип сообщения. В поле **CODE** некоторых сообщений ICMP может быть размещен код дополнительной диагностической информации. В поле **CHECKSUM** заголовка сообщений ICMP размещается контрольная сумма данного сообщения. Контрольная сумма вычисляется суммированием всех полей, начиная с поля **TYPE**. В начале вычисления контрольной суммы значение поля **CHECKSUM** полагается равным 0.

Сообщение Destination Unreachable

Сообщение Destination Unreachable — (цель недостижима) принадлежит к непарным сообщениям ICMP. Это сообщение формируется в том случае, если запрошенный сетевой ресурс является недоступным для запрашивающей его станции. Структура сообщения ICMP Destination Unreachable приведена на рис. 3.2.

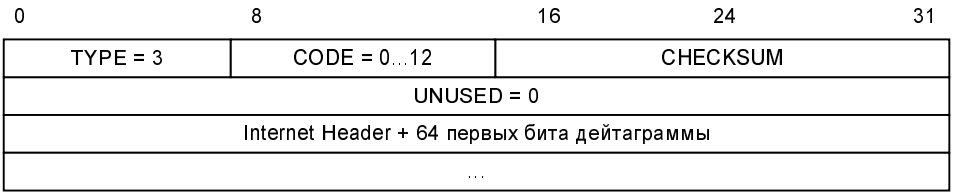


Рис. 3.2. Структура сообщения протокола ICMP Destination Unreachable

В поле **CODE** сообщения Destination Unreachable размещается код, который соответствует типу запрошенного недоступного сетевого ресурса или конкретизирует причину, вследствие которой запрошенный сетевой ресурс оказался недоступен. Диагностические коды сообщений Destination Unreachable представлены в десятичной системе исчисления и их значения приведены в табл. 3.2.

Таблица 3.2. *Диагностические коды сообщений Destination Unreachable*

CODE	Тип кода	Значение
0	Network Unreachable	Сеть назначения недостижима
1	Host Unreachable	Узел назначения недостижим
2	Protocol Unreachable	Протокол недостижим
3	Port Unreachable	Порт недостижим
4	Fragmentation Need & DF set	Необходима фрагментация, однако она запрещена
5	Source Route Failed	Маршрут в сеть назначения неверен
6	Destination Network Unknown	Сеть назначения неизвестна
7	Destination Host Unknown	Узел назначения неизвестен
8	Source Host Isolated	Источник изолирован
9	Communication with destination Network Administratively Prohibited	Взаимодействие с сетью назначения запрещено

Таблица 3.2 *(окончание)*

CODE	Тип кода	Значение
10	Communication with destination Host Administratively Prohibited	Взаимодействие с узлом назначения запрещено
11	Network Unreachable for type of service	Сеть назначения недоступна для запрошенного типа обслуживания
12	Host Unreachable for type of service	Узел назначения недоступен для запрошенного типа обслуживания

Сообщения Destination Unreachable могут быть сформированы как станцией назначения (**CODE** = 2 и 3), так и одним из промежуточных маршрутизаторов — шлюзов (**CODE** = 0, 1, 6 и т. д.). При этом в качестве адреса источника должен быть указан IP-адрес узла, обнаружившего проблему. Например, сообщение со значением в поле **CODE** = 1 — Host Unreachable может быть сформировано последним маршрутизатором, пытавшимся доставить сообщение до узла назначения по непосредственно подключенной сети. Если узел по каким-либо причинам не смог ответить на запрос ARP, непосредственная доставка сообщения не может быть выполнена. Именно в этом случае маршрутизатор и должен сформировать сообщение Host Unreachable (Узел назначения недостижим). При передаче этого сообщения в качестве адреса назначения должен быть использован адрес IP источника первичного сообщения. Для того чтобы станция — источник забракованной дейтаграммы смогла правильно интерпретировать диагностическое сообщение, в тело сообщения Destination Unreachable помещается заголовок и первые 8 байт исходной дейтаграммы.

Сообщение Time Exceeded

Сообщение Time Exceeded — (Время жизни истекло) относится к непарным сообщениям ICMP. Это сообщение должно быть сформировано в том случае, если в процессе передачи дейтаграммы истек установленный интервал времени ее существования в сети. Величина этого интервала определяется значением поля **TTL** предаваемой дейтаграммы. Структура сообщения ICMP Time Exceeded приведена на рис. 3.3.

Принудительное ограничение времени жизни дейтаграмм в сети Интернет используется для того, чтобы избежать бесконечной циркуляции в сети дейтаграмм, отправленных по неправильным маршрутам. Циркуляция может, в частности, возникнуть при возникновении в сети циклических маршрутов, что, в свою очередь, может быть вызвано ошибками при функционировании протоколов маршрутизации. В циклические маршруты могут быть включены два и более маршрутизаторов.

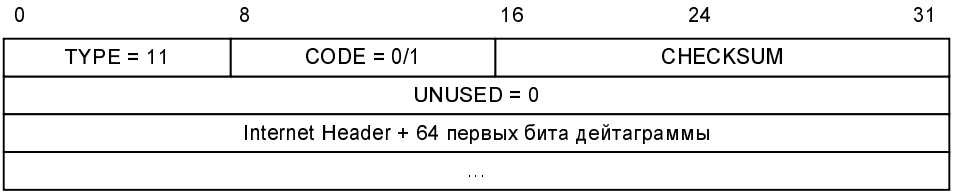


Рис. 3.3. Структура сообщения протокола ICMP Time Exceeded

Значение поля **CODE** в сообщении Time Exceeded используется для уточнения причины, по которой дейтаграмма прекратила существование:

- **CODE** = 0 — в процессе передачи дейтаграммы поле **TTL (Time To Live)** (см. рис. 1.5) приняло значение 0;
- **CODE** = 1 — таймер дефрагментации установился в 0 до полной сборки принятого сообщения.

Сообщения Time Exceeded со значением **CODE** = 0 могут быть сформированы маршрутизатором. Сообщения Time Exceeded со значением **CODE** = 1 должны быть сформированы узлом назначения. При передаче этого сообщения в качестве адреса назначения используется адрес IP источника первичного сообщения. Для того чтобы станция-источник смогла правильно интерпретировать диагностическое сообщение, в тело сообщения Destination Unreachable помещается заголовок и первые 8 байтов исходной дейтаграммы.

Сообщение Parameter Problem

Сообщение Parameter Problem — (Неверный параметр) принадлежит к непарным сообщениям ICMP. Сообщение данного типа должно быть сформировано в том случае, если в процессе обработки заголовка дейтаграммы на узле или маршрутизаторе были обнаружены некорректные аргументы, которые делают невозможным дальнейшее перемещения дейтаграммы. В этом случае дейтаграмма должна быть уничтожена, а в адрес ее источника передается сообщение Parameter Problem. Структура сообщения ICMP Parameter Problem приведена на рис. 3.4.

Рис. 3.4. Структура сообщения протокола ICMP Parameter Problem

В поле **CODE** сообщения Parameter Problem размещается признак типа диагностической информации. В том случае, если в этом поле находится код "0", значение поля **Pointer** сообщения Parameter Problem соответствует номеру байта в заголовке исходного сообщения, который не может быть адекватно интерпретирован. Например, значение поля **Pointer** = 1 в данном случае указывает на возникновение проблемы с интерпретацией поля **Service Type (TOS)** передаваемой дейтаграммы (см. рис. 1.5).

Значение поля **CODE** = 1 должно быть сформировано в ситуации, когда причина остановки дейтаграммы заключается в несоответствии запрашиваемых параметров установленным требованиям. Такими требованиями могут быть, в частности, требования по обеспечению безопасности. При передаче этого сообщения в качестве адреса назначения должен быть использован адрес IP источника первичного сообщения. Так же, как и в предыдущих случаях, для того, чтобы станция-источник смогла правильно интерпретировать диагностическое сообщение, в тело сообщения Parameter Problem помещается заголовок и первые 8 байтов исходной дейтаграммы.

Сообщение Source Quench

Сообщение Source Quench — (Сдерживание источника) относится к непарным сообщениям ICMP. Это сообщение должно быть сформировано в том случае, если в процессе передачи дейтаграммы на одном из промежуточных узлов возникла перегрузка.

Причинами возникновения перегрузки в сети IP могут быть недостаточная пропускная способность канала передачи данных или недостаточная скорость обработки пакетов на промежуточном узле сети. В первом случае на узле начинают накапливаться пакеты, которые невозможно отправить потому, что скорость их поступления на данный узел превышает возможную скорость их передачи. Во втором случае принятые пакеты будут накапливаться потому, что промежуточный узел по какой-либо причине не успевает обрабатывать дейтаграммы в темпе их поступления.

В обоих этих случаях на узле начинает увеличиваться очередь, состоящая из необработанных пакетов. Для временного хранения блоков данных ожидающих обслуживания обычно используются специальные области оперативного запоминающего устройства узла или маршрутизатора называемые буферными ЗУ. Поскольку эти области имеют ограниченный объем, длительное превышение пропускной способности узла передачи данных могут привести к тому, что этот буфер будет переполнен. В этом случае блоки данных, поступившие на данный узел после возникновения перегрузки, потеряются.

Для того чтобы указать на возникновение подобной ситуации или предотвратить ее появление, используется сообщение Source Quench. Это сообщение формируется на при возникновении перегрузки или появлении первых

ее признаков. Сообщение Source Quench передается в адрес источников, формирующих трафик, который не может быть обработан на данном узле вследствие перегрузки. Обычно это сообщение формируется всякий раз, когда уничтожается пакет, который не может быть помещен в буфер из-за его переполнения.

При получении от станции назначения или одной из промежуточных станций сообщения Source Quench источник должен, используя специальные процедуры, уменьшить скорость информационного обмена в указанном направлении.

Структура сообщения ICMP Source Quench приведена на рис. 3.5.

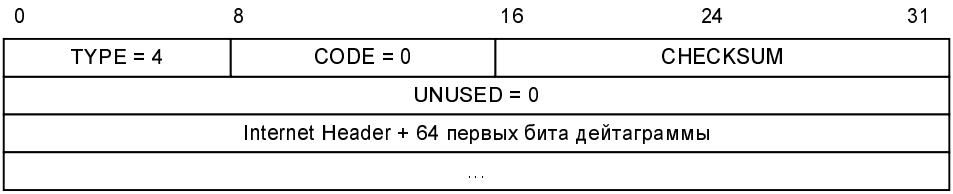


Рис. 3.5. Структура сообщения протокола ICMP Source Quench

При передаче сообщения Source Quench в качестве адреса назначения должен быть использован IP-адрес источника первичного сообщения. Для того, чтобы станция-источник смогла правильно интерпретировать диагностическое сообщение, в тело сообщения Source Quench помещается заголовок и первые 8 байтов исходной дейтаграммы.

В сообщении Source Quench не содержится дополнительной информации, которая могла бы обеспечить локализацию причины возникновения перегрузки. Кроме того, вполне возможно, что сообщения данного типа могут быть переданы в направлении не того абонента, который формирует избыточный для данного узла трафик.

Сообщение Redirect

Сообщение Redirect — (Перенаправление маршрута) принадлежит к непарным сообщениям ICMP. Это сообщение должно быть сформировано в том случае, если при получении дейтаграммы маршрутизатор обнаруживает, что для ее передачи был выбран неоптимальный маршрут. На рис. 3.6 приведен пример использования сообщения Redirect для изменения неверного маршрута.

В данном случае станция A (10.40.0.2) отправляет дейтаграмму в направлении станции B (10.4.5.1), используя для этого в качестве шлюза маршрутизатор M4. После того, как маршрутизатор M4 получает дейтаграмму, он определяет, что данная дейтаграмма адресована в сеть 10.4.5.0. Маршрутизатор M4 определяет также, что кратчайший маршрут для достижения этой

сети проходит через маршрутизатор M2, подключенный к тому же сегменту сети, из которого была получена принятая дейтаграмма.

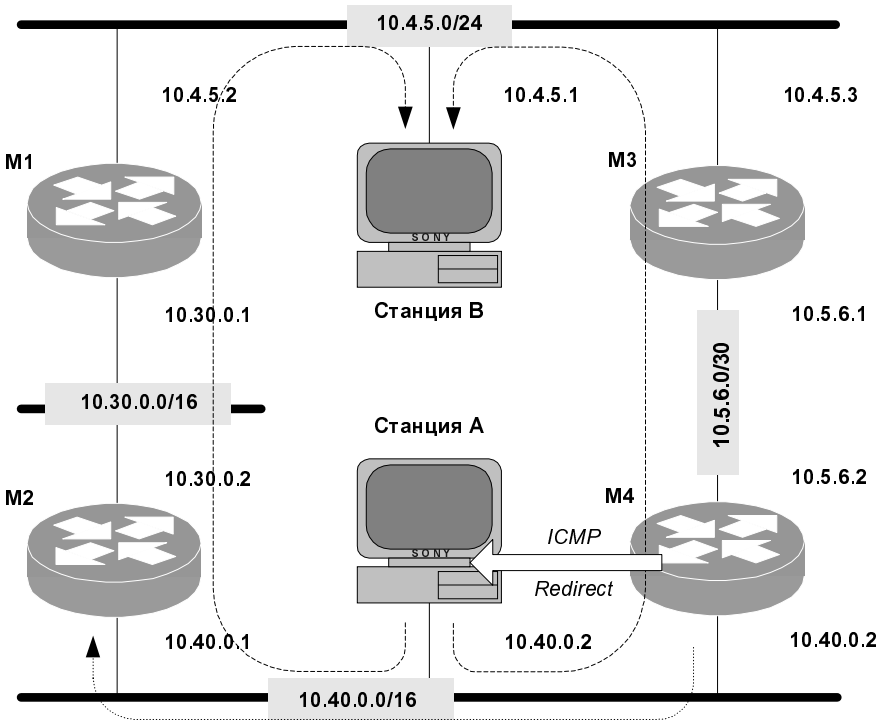


Рис. 3.6. Пример использования сообщения Redirect протокола ICMP

Поэтому маршрутизатор M4 на канальном уровне направляет полученную дейтаграмму маршрутизатору M2 и одновременно формирует сообщение ICMP Redirect, в котором он рекомендует станции A впредь для передачи дейтаграмм в сеть 10.4.5.0 использовать в качестве шлюза маршрутизатор M2.

Сообщения Redirect могут быть сформированы только узлом, выполняющим маршрутизацию в данном сегменте сети. Структура сообщения ICMP Redirect приведена на рис. 3.7.

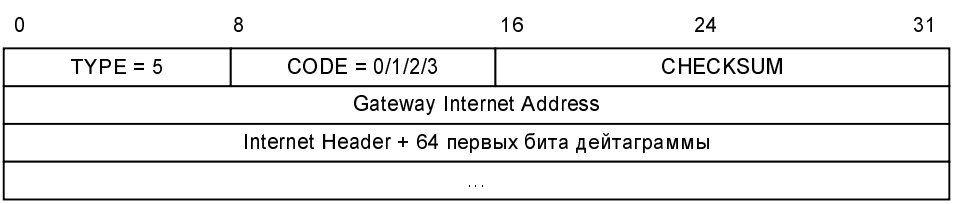


Рис. 3.7. Структура сообщения протокола ICMP Redirect

В заголовке сообщения ICMP Redirect размещается IP-адрес шлюза, который рекомендуется использовать для достижения сетевого ресурса, указанного в исходной дейтаграмме, и код типа маршрута, который должен быть изменен по предложению источника сообщения ICMP. В табл. 3.3 приведены используемые значения поля **CODE** сообщения ICMP Redirect и соответствующие им рекомендации по изменению маршрута.

Таблица 3.3. *Коды сообщений ICMP Redirect*

CODE	Тип кода	Значение
0	Redirect Datagram for networks	Изменение маршрута для сети
1	Redirect Datagram for host	Изменение маршрута для узла
2	Redirect Datagram for the Type of service and networks	Изменение маршрута для типа услуги и сети
3	Redirect Datagram for the Type of service and host	Изменение маршрута для типа услуги и узла

Сообщения Echo Request/Reply

Сообщения типа Echo (эхо) принадлежат к парным сообщениям ICMP. Сообщения этого типа используются для того, чтобы определить возможность достижения и состояние компонентов сети Интернет. В соответствии с установленными правилами формирования сообщений протокола ICMP Echo Request/Reply любой узел сети, получивший адресованное ему сообщение Echo Request, должен сформировать ответное сообщение Echo Reply в адрес источника полученного сообщения.

Таким образом, сообщения Echo Request/Reply позволяют установить наличие правильного маршрута до узла назначения и частично проверить его функционирование.

Поскольку функцию Echo Reply — сервер должен поддерживать любой компонент сети Интернет, с помощью приложений, формирующих сообщения Echo Request, можно достаточно просто установить возможность информационного обмена с тем или иным компонентом сети. Формирование сообщений Echo Request обычно осуществляется утилитами Ping (Packet inter network groper). Наиболее простые утилиты Ping посылают одно сообщение Echo Request в адрес назначения и указывают на получение ответного сообщения Echo Reply. Более сложные утилиты формируют несколько последовательных сообщений Echo Request и измеряют значение интервала

времени, отделяющего момент передачи этих сообщений от момента приема ответных сообщений Echo Reply.

Сообщения Echo Request и Echo Reply имеют похожий формат и отличаются только содержимым поля **TYPE**:

- значение поля **TYPE** = 0 соответствует сообщению Echo Reply;
- значение поля **TYPE** = 8 соответствует сообщению Echo Request.

Структура сообщения ICMP Echo приведена на рис. 3.8.

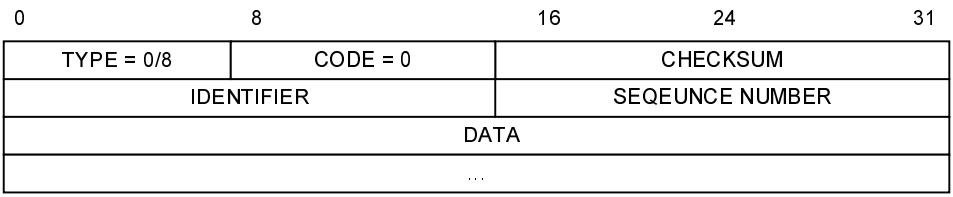


Рис. 3.8. Структура сообщения протокола ICMP Echo

В поле **CODE** сообщений Echo Request/Reply размещается значение 0. В поля **IDENTIFIER** и **SEQEUNCE NUMBER** помещаются последовательные номера сообщений для установления соответствия между переданными и полученными сообщениями.

Примечание

Правила формирования сообщений Echo протокола ICMP не рекомендуют формировать ответ на запрос, который был передан по адресу Broadcast сетевого уровня.

Атаки с использованием ICMP Echo Request/Reply

Как было выше отмечено, сообщения Echo Request/Reply используются для того, чтобы проверить правильность функционирования сети, однако, эти же сообщения могут быть использованы злоумышленником для того, чтобы нарушить ее функционирование. Для этого нарушитель использует специальный тип спланированных действий, который называется атакой.

Примечание

Атакой называют совокупность преднамеренных и координированных действий, которые предпринимаются группой злоумышленников и направлены на то, чтобы нарушить нормальный режим доступа к информационным ресурсам некоторой сети.

Существуют несколько классов атак, предназначенных для того, чтобы блокировать различные сервисы информационного взаимодействия в сети, выбранной в качестве жертвы. Наиболее часто применяются атаки, направленные на инициирование перегрузки каналов доступа к данной сети и ее

активных компонентов потоками служебных сообщений. Поскольку в этом случае остальные клиенты не могут получить доступ к ресурсам сети, такой класс атак называется DoS — Denial of Service (подавление услуги).

При проведении наиболее распространенной атаки класса DOS так называемой Smurf-атаки используются сообщения Echo Request/Reply. На рис. 3.9 представлена схема организация атаки Smurf. В организации этой атаки используются три компонента:

- Нападающий (Attacker);
- Посредник (Mediator);
- Жертва (Victim) — узел, выбранный объектом для нападения.

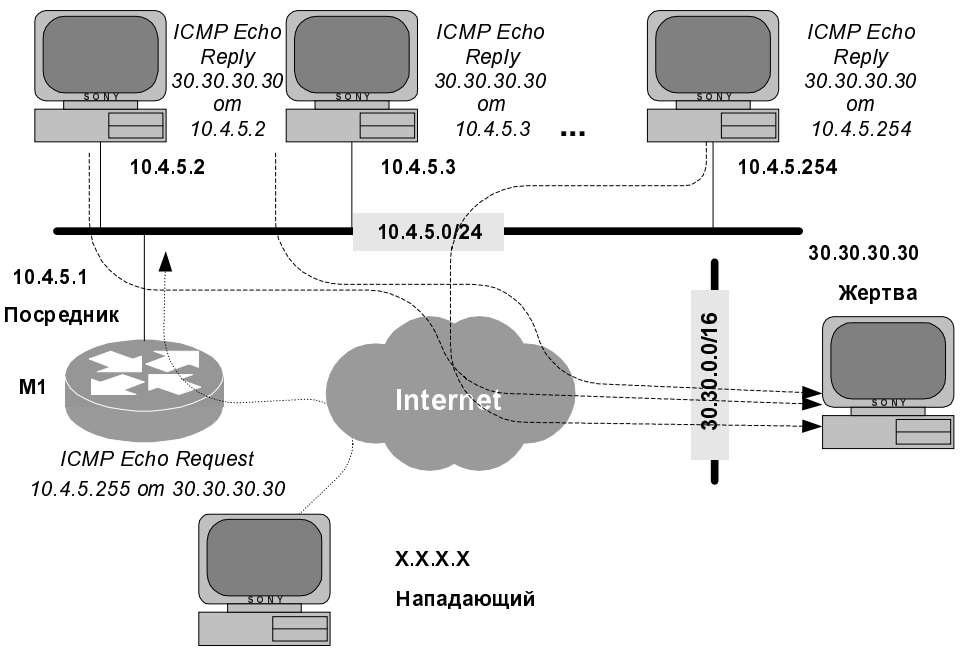


Рис. 3.9. Схема организации атаки Smurf с помощью сообщений ICMP Echo

Для успешной организации атаки данного типа необходимо, чтобы полоса пропускания канала, связывающего посредника с жертвой, была достаточно широкой. Ширина полосы пропускания канала, который связывает нападающего и посредника в данном случае не имеет большого значения.

В момент начала атаки атакующий узел формирует поток фальсифицированных сообщений типа Echo Request. Плотность этого потока обычно

определяется шириной полосы пропускания канала до посредника. Сообщения Echo Request, должны быть сформированы следующим образом:

- адрес назначения — сетевой адрес типа Directed Broadcast для сети посредника (в данном случае — 10.4.5.255);
- адрес источника — сетевой адрес жертвы (в данном случае — 30.30.30.30).

После того как маршрутизатор посредника получает сообщение, адресованное всем узлам подключенной к нему сети, он осуществляет непосредственную доставку данного сообщения, используя при этом broadcast-адрес канального уровня. Таким образом, сообщение Echo Request получают все узлы сети 10.4.5.0/24. В соответствии с правилами формирования сообщений протокола ICMP, каждый из этих узлов сформирует ответное сообщение Echo Reply в адрес источника сообщения Echo Request. Число этих ответных сообщений будет соответствовать числу активных на данный момент узлов в сети посредника. Таким образом, на каждое сообщение, поступившее от нападающего узла, посредник может сформировать десятки и сотни ответных сообщений. Поскольку в качестве адреса источника в дейтаграммах Echo Request указан адрес жертвы, все ответные сообщения будут направлены именно по этому адресу. В данном случае сеть посредника играет роль усилителя, который увеличивает мощность информационного потока, направленного на выбранную жертву.

Атаки такого типа удобны с точки зрения нападающего еще и потому, что способны обеспечить некоторую анонимность нападения, поскольку поток, направленный на жертву, формируется не в нападающем узле. Кроме того, даже если посредник определит, что вовлечен в атаку, определение истинного источника потока сфальсифицированных пакетов может потребовать значительного времени, в течение которого нападающий может либо достичь своей цели, либо сменить направление атаки.

Для осуществления атаки Smurf на один и тот же узел могут быть использованы несколько узлов-посредников. Число атакующих узлов тоже может варьироваться в широких пределах в зависимости от серьезности намерений нападающих.

Способы защиты от атак с использованием ICMP Echo Request/Reply

Для предотвращения атак с использованием ICMP Echo Request/Reply могут быть использованы самые различные способы. В данном случае большую эффективность будут иметь профилактические мероприятия, которые на-

правлены на предотвращение вовлечения сети в атаку типа Smurf в качестве посредника.

В качестве одного из таких эффективных мероприятий можно использовать запрещение приема и распространения сообщений типа Directed Broadcast. Эта мера является достаточно эффективной и простой для реализации. Использование данной меры рекомендовано документом, в соответствии с которым должны строиться правила функционирования маршрутизаторов в сети IP RFC-1122. В частности, в этом документе указано, что сообщения ICMP Echo Request типа directed broadcast, направленные в его адрес, могут быть уничтожены без формирования какого-либо диагностического сообщения.

Примечание

Маршрутизаторы некоторых производителей (например, Cisco Systems) позволяют пользователю управлять возможностью обработки трафика, который направлен в адрес Directed Broadcast с помощью соответствующей команды настройки сетевого интерфейса — "Ip directed broadcast". По умолчанию перенаправление трафика directed broadcast на всех интерфейсах маршрутизаторов Cisco запрещено.

Для противодействия вовлечения компонентов сети в атаку типа Smurf можно использовать также специальные списки доступа (Access Lists), запрещающие прием фальсифицированных пакетов. Для того чтобы отличить нормальный пакет от фальсифицированного, достаточно сопоставить сетевой адрес источника полученного пакета со значениями, указанными в таблице маршрутизации. В том случае, если номер интерфейса, с которого получен пакет, не соответствует представлениям данного маршрутизатора о размещении данной сети, этот пакет также может быть уничтожен.

В качестве меры, которая может снизить эффективность атаки, может быть рекомендовано ограничение пропускной способности канала, выделенной для передачи диагностических сообщений. Использование данной меры предусматривает возможность реализации такого ограничения программным и аппаратным обеспечением маршрутизатора.

Радикальным способом противодействия атакам Smurf является полное запрещение трафика ICMP Echo Request/Reply на входных интерфейсах. Однако данная мера лишает многих пользователей возможности использования эффективных инструментов, которые широко используются для определения статуса данного компонента сети — утилиты PING, TRACEROUTE и т. д.

Утилита PING

Утилита PING входит в состав почти всех пакетов программного обеспечения для работы с семейством протоколов TCP/IP. Параметры утилиты

обычно задаются в виде аргументов командной строки при вызове программы PING. Помимо адреса (или имени) проверяемого узла, который является обязательным параметром, пользователь может установить, например, максимальное время ожидания ответа, интервалы отправки запросов, количество пакетов запросов, размер пакетов и другие параметры программы, перечень которых зависит от ее версии и типа используемой операционной системы.

В табл. 3.4 приведен список параметров утилиты Ping Microsoft Windows.

Таблица 3.4. *Параметры утилиты Ping Microsoft Windows*

Параметр	Назначение
`-t`	Отправка пакетов на указанный узел до прерывания
`-a`	Определение адресов по именам узлов
`-n <число>`	Число отправляемых запросов
`-l <размер>`	Размер буфера отправки
`-f`	Установка флага, запрещающего фрагментацию пакета
`-i <TTL>`	Задание времени жизни пакета (поле "Time To Live")
`-v <TOS>`	Задание типа службы (поле "Type Of Service")
`-r <число>`	Запись маршрута для указанного числа переходов
`-s <число>`	Штамп времени для указанного числа переходов
`-j <Список Узлов>`	Свободный выбор маршрута по списку узлов
`-k <Список Узлов>`	Жесткий выбор маршрута по списку узлов
`-w <Интервал>`	Интервал ожидания каждого ответа в миллисекундах

Утилита TRACEROUTE

Утилита TRACEROUTE (операционная система UNIX) позволяет просматривать маршрут, по которому пакет проходит источника до узла назначения. Эта программа часто используется сетевыми и системными администраторами как средство отладки маршрутизации пакетов из локальной сети организации в Интернете.

Утилита TRACEROUTE отправляет на несуществующий порт удаленного узла последовательность UDP-дейтаграмм. По умолчанию отправляются

три дейтаграммы, для каждой из которых значение TTL устанавливается равным 1. Поскольку параметр TTL уменьшается на 1 в каждом из узлов маршрутизации, первый же маршрутизатор уничтожит данные дейтаграммы и отправит отправителю уведомления. Уведомления представляют собой ICMP-сообщения Time Exceeded (Время жизни истекло) и содержат временную метку поступления UDP-сообщения и адрес маршрутизатора.

После этого TRACEROUTE отправляет UDP-сообщения с параметром TTL = 2, для того чтобы эти сообщения уничтожил второй маршрутизатор на пути к узлу назначения, затем с параметром TTL = 3 и т. д. Этот процесс продолжается до тех пор, пока дейтаграмма не достигнет узла назначения.

Поскольку дейтаграмма адресована в несуществующий порт, узел назначения не сможет ее обработать, и сформирует сообщение ICMP Destination Unreachable (Цель недостижима). Это сообщение является признаком завершения процесса определения маршрута утилитой TRACEROUTE.

В процессе обмена UDP- и ICMP-сообщениями формируется таблица задержек маршрутизации пакетов на каждом из узлов и на пути к удаленному узлу. Эта информация и выводится на экран пользователя.

Утилита TRACEROUTE в том или ином виде входит в состав почти всех пакетов программного обеспечения для работы с семейством протоколов TCP/IP. Параметры утилиты задаются в виде аргументов командной строки вызова программы TRACEROUTE. Кроме адреса (или имени) узла назначения, который является обязательным параметром, пользователь может установить, например, максимальное значение TTL IP-пакетов (по умолчанию этот параметр равен 30), количество отправляемых UDP-дейтаграмм (по умолчанию — 3), время ожидания ответа, номер используемого порта (по умолчанию — 33434).

Утилита PATHPING

Утилита PATHPING (операционная система Microsoft Windows 2000) сочетает в себе возможности популярных утилит PING и TRACEROUTE, поскольку позволяет одновременно определять факт достижимости узла сети и маршрут, который связывает этот узел с проверяющим узлом. Утилита PATHPING обеспечивает ряд дополнительных возможностей:

- задание адреса источника сообщения ICMP;
- проверку поддержки протокола RSVP (ReSerVation Protocol) узлами маршрута;
- возможность использования адресов IPv6 и IPv4;

На рис. 3.10 приведен пример использования утилиты PATHPING.

```
:\ >pathping 172.24.208.3

Трассировка маршрута к non.sense.ural.ru  [172.24.208.3]
с максимальным числом прыжков 30:
 0   fish2k  [172.24.24.64]
 1  172.24.24.36
 2  192.168.66.10
 3   non.sense.ural.ru  [172.24.208.3]

Подсчет статистики за: 75 сек. ...
         Исходный узел     Маршрутный узел
Нор   RTT    Утер./Отпр.   %   Утер./Отпр.  %   Адрес
 0                                    fish2k  [172.24.24.64]
                          0/ 100 =  0%   |
  1    0мс     0/ 100     =  0%     0/ 100 =  0%  172.24.24.36
                          1/ 100 =  1%   |
  2    0мс     2/ 100     =  2%     1/ 100 =  1%  192.168.66.10
                          0/ 100 =  0%   |
  3    0мс       1/ 100 =  1%     0/ 100 =  0%   non.sense.ural.ru [172. 24.208.3]


Трассировка завершена
```

Рис. 3.10. Пример использования утилиты PATHPING (Microsoft Windows 2000)

Сообщения Timestamp Request/Reply

Сообщения Timestamp Request/Reply — (метки времени) принадлежат к парным сообщениям ICMP. Эти сообщения предназначаются для обеспечения взаимной синхронизации счетчиков времени у различных компонентов сети Интернет. Необходимость синхронизации счетчиков времени на узлах этой сети может быть вызвана, в частности, желанием установить точное время возникновения неисправности. Поскольку компоненты сети Интернет не имеют общего управления, значения индивидуальных датчиков времени у каждого из компонентов могут существенно отличаться. Для того чтобы обеспечить взаимную синхронизацию этих датчиков, могут быть использованы сообщения Timestamp Request/Reply.

Сообщения Timestramp Request и Timestramp Reply имеют одинаковый формат и отличаются только содержимым поля **TYPE**:

- значение поля **TYPE** = 14 соответствует сообщению Timestamp Reply;
- значение поля **TYPE** = 13 соответствует сообщению Timestamp Request.

Структура сообщения ICMP Timestamp приведена на рис. 3.11.

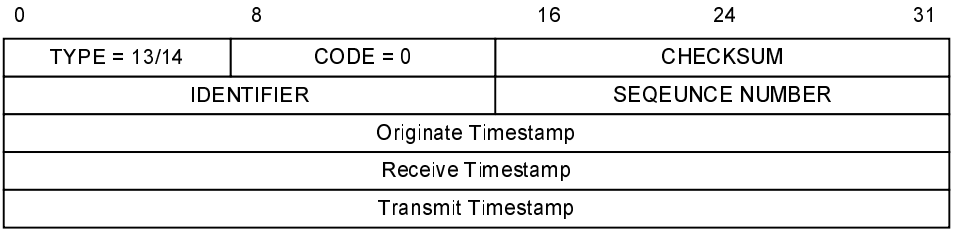


Рис. 3.11. Структура сообщения ICMP Timestamp

В поля **IDENTIFIER** и **SEQEUNCE NUMBER** помещаются последовательные номера сообщений. В поле **Originate Timestamp** сообщения Timestamp Reply отправитель помещает значение своего датчика времени в момент отправки своего сообщения. В момент получения сообщения ICMP Timestamp узел назначения помещает в поле **Receive Timestamp** значение собственного датчика времени. Значения этих полей принимающий узел переписывает в соответствующие поля ответного сообщения Timestamp Reply. Поле **Transmit Timestamp** формируется в момент его передачи в адрес отправителя исходного запроса. Значения этого поля в данном случае будет соответствовать значению счетчика времени принимающего узла в момент отправки сообщения.

Получив сообщение Timestamp Request, отправитель запроса может определить величину задержки распространения между ним и принимающим узлом и таким образом сформировать поправку к своему датчику времени.

Примечание

Описанный метод не может обеспечить достаточной точности хотя бы потому, что задержки распространения сигналов в прямом и обратном направлении могут существенно отличаться и изменяться во времени. Поэтому для обеспечения надежной взаимной синхронизации датчиков времени должны быть использованы несколько сессий синхронизации.

Сообщения Address Mask Request/Reply

Сообщения Address Mask Request/Reply — (Запрос/Ответ маски адреса) принадлежит к парным сообщениям ICMP. Сообщения этого типа предназначены для проверки установленного значения маски для внеклассовой сети.

Использование внеклассовых сетей вызывает необходимость передачи информации о длине используемой маски между компонентами сети. В момент первоначального включения узел должен передать несколько сообщений типа Address Mask Request (Запрос маски адреса). До получения ответа на эти сообщения узел должен считать, что маска сети, к которой он подключен, соответствует классу этой сети. При получении первого ответа Address Mask Reply (Ответ маски адреса) узел должен установить принятое значение в качестве маски для данной сети.

Структура сообщения ICMP Address Mask Request/Reply приведена на рис. 3.12.

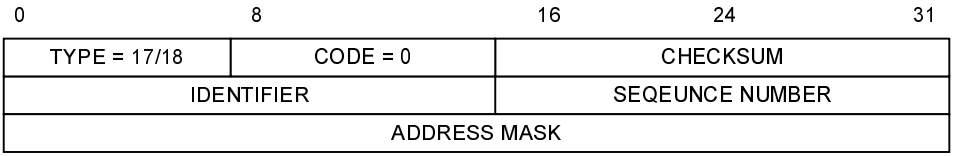


Рис. 3.12. Структура сообщения ICMP Address Mask Request/Reply

Сообщения Address Mask Request и Address Mask Reply имеют одинаковый формат и отличаются только содержимым поля **TYPE**:

- значение поля **TYPE** = 18 соответствует сообщению Address Mask Reply;
- значение поля **TYPE** = 17 соответствует сообщению Address Mask Request.

В поле ADDRESS MASK сообщения Address Mask Reply должна быть размещена маска сети.

Протоколы RARP и DHCP

Наряду с множеством достоинств семейства протоколов TCP/IP его основным недостатком является сложность управления адресами: обычно требуется приложить немало усилий, чтобы поддерживать сеть TCP/IP в рабочем состоянии.

Корректное назначение узлу IP-адреса может оказаться весьма непростой задачей. В частности, во избежание дублирования администратор должен иметь список значений всех ранее присвоенных адресов. Кроме того, если узел был перемещен в другую подсеть, то часть сетевого адреса, содержащая номер подсети, должна быть изменена в соответствии с номером новой подсети, в то время как номер узла не должен совпадать с используемыми номерами в новой сети. Для упрощения процедуры администрирования сетевых адресов на разных этапах развития сети Интернет применялись протоколы RARP (Reverse Address Resolution Protocol) и DHCP (Dynamic Host Configuration Protocol).

Протокол RARP

Протокол RARP (Reverse ARP) предназначен для выполнения функции, диаметрально противоположной функции, выполняемой протоколом ARP. Протокол RARP предназначен для определения логического сетевого адреса для узла сети с определенным физическим MAC-адресом (Media Access Control). Необходимость в использовании такого протокола возникала в тех случаях, когда в локальной сети использовались бездисковые рабочие станции. Для обеспечения функционирования устройств данного типа необхо-

димо после включения питания произвести загрузку в их оперативное запоминающее устройство операционной системы. Проблема заключается в том, что для выполнения этой операции необходимо присвоить каждой станции уникальный сетевой адрес. Для присвоения логического адреса не может быть использован такой же способ, какой используется для присвоения адресов физического уровня — программирование при изготовлении. Поскольку специальных запоминающих устройств, обеспечивающих хранение сетевого адреса, на бездисковой рабочей станции нет и быть не может, следовательно, этот адрес должен быть присвоен ей динамически. Для динамического присвоения сетевого адреса бездисковым рабочим станциям используется протокол RARP. Для обеспечения информационного обмена по данному протоколу используются блоки данных канального уровня — кадры, имеющие такую же структуру, как и кадры протокола ARP. Отличие заключается только в значении поля **Type/Code** (см. рис. 2.5):

- для RARP — 8035;
- для ARP — 0806.

Несмотря на то, что оба этих протокола в той или иной степени используют сетевые адреса, они не могут быть отнесены к протоколам сетевого уровня, поскольку для адресации абонентов применяют их физические адреса. Другая особенность этих протоколов заключается в том, что эти физические адреса дублируются в теле кадра. В частности, запрос ARP содержит два одинаковых значения MAC-адреса станции отправителя. Одно находится в заголовке внешнего кадра, второе значение размещается в кадре ARP. Несмотря на то, что эти значения должны совпадать, для ответа на данный запрос используется значение, которое расположено в кадре ARP.

Организация RARP-взаимодействия компонентов сети

Для определения сетевого адреса $A_л$ компонента сети по известному физическому адресу $A_ф$ станция B, желающая организовать информационное взаимодействие со станцией A, формирует специальный блок данных канального уровня — кадр RARP. В этот кадр, наряду со служебной информацией, помещается физический адрес самой станции. Для того чтобы этот кадр мог достичь всех абонентов данной сети, в качестве MAC-адреса назначения этого пакета используется адрес Broadcast канального уровня. Сформированный таким образом кадр называется RARP-запрос (RARP request). На кадр такого типа может ответить только устройство, которое выполняет функцию RARP-сервера.

RARP-сервер

Функцию RARP-сервера выполняет специальная станция, способная установить соответствие между физическим и математическим адресами других

станций данной сети. Обычно это соответствие устанавливается при помощи специальных динамических таблиц, в которых каждой станции по ее физическому адресу ставится в соответствие логический, т. е. сетевой адрес. Таким образом, информационное взаимодействие при выполнении протокола RARP состоит из следующих этапов:

- RARP-сервер получает запрос от рабочей станции;
- RARP-сервер определяет значение MAC-адреса;
- RARP-сервер определяет по таблице значение сетевого адреса;
- RARP-сервер формирует кадр RARP-ответ.

Очевидно, что использование одной рабочей станции в качестве RARP-сервера не может обеспечить достаточной надежности. Единственный RARP-сервер может выйти из строя или быть слишком перегружен для того, чтобы вовремя ответить на RARP-запрос. Не получив ответ, станция, которая сформировала запрос, будет вынуждена повторять его снова и снова. Поскольку каждый из этих запросов передается по адресу типа broadcast, в сети может возникнуть перегрузка. Единственным выходом из данной ситуации является резервирование RARP-сервера.

Следует учесть, что простое резервирование (например — дублирование) RARP-серверов может привести к возникновению дополнительных проблем. К таким проблемам, в частности, относится возникновение коллизий при одновременном ответе на RARP-запрос двумя RARP-серверами. Для разрешения этой проблемы должно быть проведено ранжирование серверов, то есть разделение их на первичные и вторичные. Для предотвращения коллизий между первичным и вторичными серверами могут быть использованы две схемы задержки ответа вторичного RARP-сервера:

- схема задержки на такт;
- схема задержки на случайный отрезок времени.

Использование первой схемы целесообразно в тех сетях, где используется только один вторичный RARP-сервер. В данном случае вторичный сервер отвечает на RARP-запрос только в том случае, если он был послан повторно. Очевидно, что использование данной схемы не позволяет избежать возникновения коллизии в том случае, когда первый запрос был потерян из-за временной перегрузки первичного RARP-сервера или вследствие возникновения проблем в канале передачи на физическом уровне.

При использовании второй схемы в сети могут находиться несколько вторичных RARP-серверов. Каждый из этих вторичных серверов может ответить на RARP-запрос только по истечении интервала времени, величину которого он определяет случайным образом. Очевидно, что в данном случае вероятность возникновения коллизий при ответе вторичных серверов существенно уменьшается.

Протокол DHCP

Протокол DHCP (Dynamic Host Configuration Protocol) применяется для динамической настройки основных параметров процедур информационного взаимодействия узла на сетевом уровне Интернета. DHCP является расширением и дополнением протокола BOOTP (Bootstrap Protocol), который был спроектирован для автоматического назначения IP-адресов бездисковым рабочим станциям.

DHCP состоит из двух компонент: протокола, обеспечивающего передачу конфигурационных параметров сетевому узлу (например, маски сетевого адреса), и механизма назначения узлам сетевых адресов.

Протокол DHCP использует клиент-серверную модель взаимодействия. Это означает, что один или несколько узлов сети являются DHCP-серверами и управляют передачей параметров на узлы DHCP-клиентов. Сервер DHCP, как правило, назначается и инсталлируется сетевым администратором.

Примечание

Если сеть состоит из нескольких физических подсетей, объединенных маршрутизаторами, эти маршрутизаторы должны выполнять функцию BOOTP Relay Agent. В противном случае необходимо будет установить отдельный DHCP-сервер в каждой подсети.

Список параметров узла, которыми управляет протокол DHCP, определен в RFC-1541. Эти параметры протокол DHCP размещает в базе данных клиентов. Записи этой базы данных состоят из уникального идентификатора клиента (который может, например, включать в себя номер сети и идентификатор узла в этой сети) и значений конфигурационных параметров клиентов. Любой DHCP-клиент из этой базы данных имеет возможность запросить свои конфигурационные параметры, используя описанный выше механизм сообщений.

База данных DHCP содержит список доступных IP-адресов и может содержать дополнительную адресную информацию о различных разделяемых сетевых ресурсах или сервисах. Как только компьютер, который является клиентом DHCP, обращается с запросом к DHCP-серверу, ему назначается уникальный IP-адрес, который затем вместе с адресами разделяемых ресурсов передается на этот компьютер.

Протокол DHCP поддерживает три механизма назначения IP-адресов. При помощи механизма "автоматического назначения" DHCP присваивает узлу клиента постоянный IP-адрес. Механизм "динамического назначения" присваивает IP-адрес узлу на некоторый ограниченный период времени или до тех пор, пока сервер не отзовет этот IP-адрес у клиента. Механизм "ручного назначения" IP-адресов состоит в том, что процедурой присвоения узлу IP-адреса занимается администратор сети, а DHCP-сервер только передает клиенту параметры конфигурации. В сети, в зависимости от политики администрирования, могут одновременно использоваться любые из этих механизмов.

Использование механизмов DHCP назначения IP-адресов имеет явные преимущества.

- Пользователи и администратор сети избавлены от необходимости ручного ввода сетевых параметров.
- Автоматическое конфигурирование сетевых параметров позволяет избегать ошибок при их определении и повышает уровень информационной безопасности в сети.
- В тех случаях, если пользователь перемещает компьютер на новое место или использует ноутбук, содержащий PCMCIA Ethernet-адаптер, правильный IP-адрес будет назначаться автоматически.

Механизм динамического назначения адресов

Механизм динамического назначения адресов обеспечивает выделение клиенту запрашиваемого адреса на некоторый период времени. Механизм назначения адреса DHCP-сервера гарантирует, что этот адрес не будет использоваться в этот период времени каким-либо другим узлом. По окончании работы клиент возвращает (освобождает) этот адрес для дальнейшего использования. Клиент так же может запросить адрес "навсегда". В этом случае сервер разрешает длительное, но все же не "постоянное" использование запрошенного IP-адреса.

Механизм DHCP строится на обмене сообщениями между DHCP-клиентом и сервером.

1. После первоначального включения клиент отправляет запрос на получение IP-адреса в данной сети DHCPDISCOVER (обращение) (здесь и далее приведены наименования сообщений DHCP, указанные в документе RFC-1541). Запрос содержит MAC-адрес компьютера отправителя.
2. Все активные DHCP-серверы посылают ему в ответ широковещательные сообщения (поскольку клиент еще не имеет IP-адреса) DHCPOFFER (предложение). Это сообщение содержит MAC-адрес клиента, предлагаемый ему IP-адрес, маску адреса подсети, продолжительность срока резервирования адреса и IP-адрес сервера, отправившего сообщение.
3. Клиент выбирает для использования один из полученных сетевых адресов и посылает сообщение DHCPREQUEST (запрос) который наряду с согласованными значениями параметров содержит IP-адрес сервера, предложение которого было принято. Остальные серверы освобождают не принятые адреса для дальнейшего использования.
4. Сервер, который представил принятый адрес, отправляет сообщение "признательности" DHCPACK (подтверждение), в котором содержится IP-адрес, маска адреса и согласованные значения дополнительных параметров. После получения этого сообщения клиент обладает всеми необ-

ходимыми данными для информационного взаимодействия на сетевом уровне Интернета. В том случае, если сервер не может принять предложенные клиентом значения дополнительных параметров, он посылает пакет DHCP_NAK (отказ в подтверждении), что вынуждает клиента начать процесс согласования заново.

Механизм продления срока использования IP-адреса также основан на обмене сообщениями между клиентом и сервером, выдавшим этот адрес. Если клиентская машина должна иметь всегда один и тот же IP-адрес, тогда удобно воспользоваться механизмом "ручного назначения" адреса и назначить машине статический адрес, который не будет меняться.

Настройка параметров DHCP-сервера, в частности, базы разрешенных адресов, осуществляется программными средствами той операционной системы, в которой функционирует DHCP-сервер.

Реализация протокола DHCP

Для передачи сообщений протокола DHCP используется транспортный протокол UDP. Сообщения от DHCP-клиента на DHCP-сервер передаются в порт 67, а сообщения от DHCP-сервера DHCP-клиенту передаются в порт 68. Формат DHCP-пакета показан на рис. 3.13 (в скобках указана длина полей в байтах).

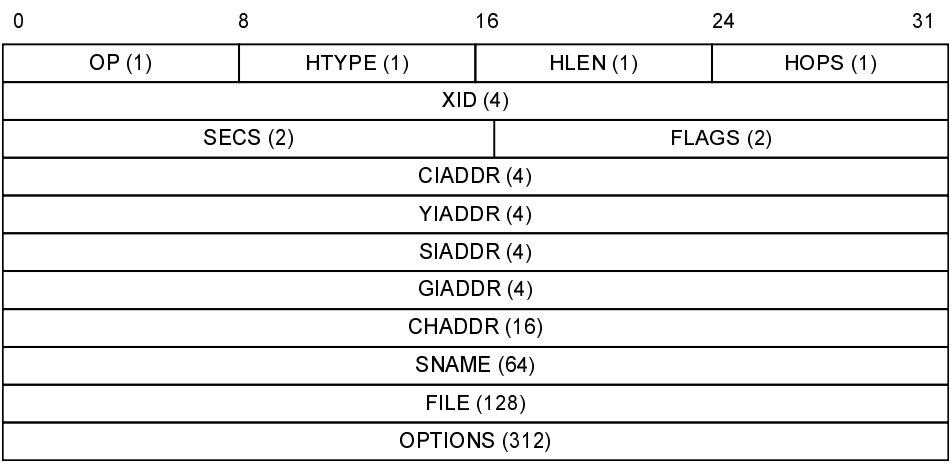


Рис. 3.13. Формат DHCP-пакета

Перечислим назначение полей формата DHCP-пакета.

- **OP** — код операции (1, если запрос; 2, если ответ).
- **HTYPE** — тип адреса канального уровня (1 для 10 Мбайт Ethernet).
- **HLEN** — длина адреса канального уровня (6 для Ethernet).

- **HOPS** — поле используется boot relay-агентом при начальной загрузке через агента. На машине клиента устанавливается в значение 0.
- **XID** — идентификатор транзакции — случайное число, выбираемое клиентом. Используется для идентификации сессии обмена сообщениями между клиентом и сервером.
- **SECS** — количество секунд, прошедших с момента начала загрузки сетевого модуля клиента (устанавливается клиентом и используется для контроля различных временных параметров).
- **FLAGS** — левый бит поля (1000000) поля используется в качестве флага широковещательного сообщения, остальные биты не используются и должны быть равны 0.
- **CIADDR** — клиентский IP-адрес (запрос). Поле заполняется на машине клиента при передаче запроса соответствия параметров узла записям базы данных DHCP.
- **YIADDR** — клиентский IP-адрес (ответ). Поле заполняется на DHCP-сервере по запросу DHCPOFFER на получение IP-адреса.
- **SIADDR** — IP-адрес следующего DHCP-сервера, опрашиваемого при начальной загрузке сетевого модуля клиента.
- **GIADDR** — IP-адрес relay-агента, используемого при загрузке клиента через relay-агента.
- **CHADDR** — адрес канального уровня клиента.
- **SNAME** — строка, содержащая имя узла сервера (необязательный параметр).
- **FILE** — имя файла загрузки, имя группы загрузки или нули. Поле используется для идентификации загрузочной части сетевого обеспечения клиента.
- **OPTIONS** — дополнительные параметры.

Несмотря на то, что по сравнению со всеми предшествующими попытками автоматического назначения адресов протокола DHCP является несомненным достижением, он все же не свободен от недостатков.

Ввиду того, что DHCP использует функциональную модель "клиент-сервер", ни один из клиентов не сможет работать при выходе из строя сервера DHCP. Задание длительного срока аренды для предотвращения слишком частых запросов о продлении, а также установка нескольких DHCP-серверов для обслуживания новых запросов позволяют сделать эту модель более устойчивой. Если один сервер выйдет из строя, то, например, при 24-часовой аренде у администратора будет достаточно времени для восстановления системы или установки нового узла с тем же самым пулом адресов. Недостатком протокола DHCP является также то, что запрос о назначении IP-адреса передается от клиента к серверу с использованием Broadcast-адреса сетевого уровня. В ряде случаев это может привести к разрыванию

зон администрирования и частичной потере управляемости сети. В качестве примера рассмотрим сеть, представленную на рис. 3.14.

В представленной схеме клиент располагается в сети 10.4.5.0/24, в которой нет DHCP-серверов. Поскольку маршрутизаторы не пропускают широковещательный трафик через свои интерфейсы, широковещательный запрос DHCPDISCOVER, сформированный клиентом, не сможет дойти до сервера, расположенного в сети 10.0.1.0/24. Для обеспечения возможности динамического конфигурирования станций, размещенных в подобных сетях, маршрутизаторы Cisco Systems позволяют преобразовывать широковещательные запросы в одноадресные при помощи команды `ip helper-address`. В приведенной схеме эта команда должна быть выполнена на интерфейсе E0 маршрутизатора M1.

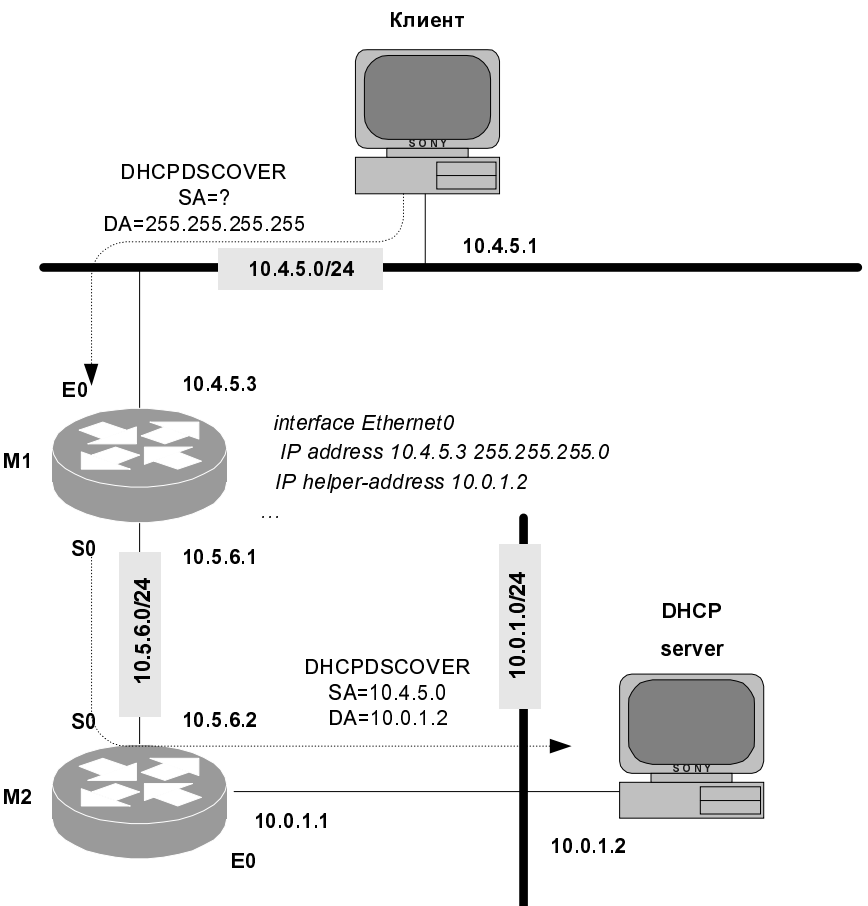


Рис. 3.14. Преобразование широковещательного запроса при передаче сообщения DHCPDISCOVER

Протоколы динамического резервирования

Протоколы Cisco Systems Inc. HSRP (Hot Standby Router Protocol) и VRRP (Virtual Router Redundancy Protocol) предназначены для обеспечения гибкого и прозрачного резервирования маршрутизаторов ответственных узлов сети. Использование этих протоколов позволяет строить надежные резервированные сети, мгновенно и автоматически восстанавливающие свою работоспособность при возникновении неисправностей. Патент на протокол HSRP принадлежит Cisco Systems Inc (US Patent number 5,473,599).

Назначение протокола HSRP

Метод горячего резервирования ответственных узлов сети достаточно часто используется для повышения надежности всей системы в целом. Однако, в том случае, если резервируемым узлом является маршрутизатор, процедура резервирования существенно усложняется. В самом деле, при выходе из строя основного маршрутизатора узлы, использовавшие его IP-адрес в качестве Default Gateway *(см. разд. "Статическое и динамическое определение маршрута" гл. 4)*, для продолжения нормального функционирования должны будут оперативно изменить свою таблицу маршрутизации, вписав в нее адрес резервного маршрутизатора. Этот процесс можно автоматизировать и ускорить в том случае, если таблица маршрутизации узла строится динамически с помощью протоколов маршрутизации. Следует отметить, что даже в этом случае изменение конфигурации произойдет далеко не мгновенно, поскольку для любого протокола маршрутизации определяются ограниченные интервалы времени для определения статуса партнера. Если учесть, что в современных сетях для подключения маршрутизаторов применяются высокоскоростные технологии (100/1000/10000 Мбит/с), то можно легко подсчитать, какими потерями обернется 1–2-минутное выключение пограничного маршрутизатора.

К сожалению, проблема подключения резервного маршрутизатора не ограничивается только сетевым уровнем. Даже если резервный маршрутизатор примет на себя IP-адрес вышедшего из строя основного маршрутизатора, рабочие станции на канальном уровне все равно будут продолжать передавать трафик, используя MAC-адрес неисправного маршрутизатора в течение периода времени жизни записи в ARP-cache. Поэтому протокол, обеспечивающий прозрачное для пользователя резервирование маршрутизаторов, кажется особенно актуальным.

Обязательным условием использования протокола HSRP для резервирования маршрутизаторов является применение технологии множественного доступа при подключении к среде передачи, поскольку в его основе лежит принцип "скольжения" адресов канального и сетевого уровня внутри резервирующей группы маршрутизаторов. Только один член резервирующей

группы обладает адресами виртуального маршрутизатора. Резервирующую группу (Standby Group) образуют основной и резервный (или резервные) маршрутизаторы. Резервирующая группа HSRP представляется остальным компонентам сети в виде одного виртуального маршрутизатора, которому соответствует один адрес сетевого уровня и один адрес канального уровня.

Примечание

Каждый из маршрутизаторов резервирующей группы, естественно, имеет индивидуальный MAC-адрес и может иметь собственный IP-адрес для удобства управления. Таким образом, резервирующая группа из *N* маршрутизаторов имеет *N*+1 MAC-адресов.

После того как один из маршрутизаторов резервирующей группы выбирается в качестве основного (активного), он принимает на себя виртуальные адреса канального и сетевого уровня и собственно функцию маршрутизации. Для того, чтобы подтвердить свой статус, активный маршрутизатор через установленные интервалы времени посылает маршрутизаторам резервирующей группы специальное сообщение HELLO (приветствие). Если маршрутизаторы резервирующей группы не получают HELLO от активного маршрутизатора в течение установленного интервала времени, они констатируют, что активный маршрутизатор вышел из строя или выключен, и выбирают новый активный маршрутизатор. На рис. 3.15 представлен вариант построения резервирующей группы из двух маршрутизаторов. Виртуальному маршрутизатору в этом случае присвоен адрес 10.4.5.1.

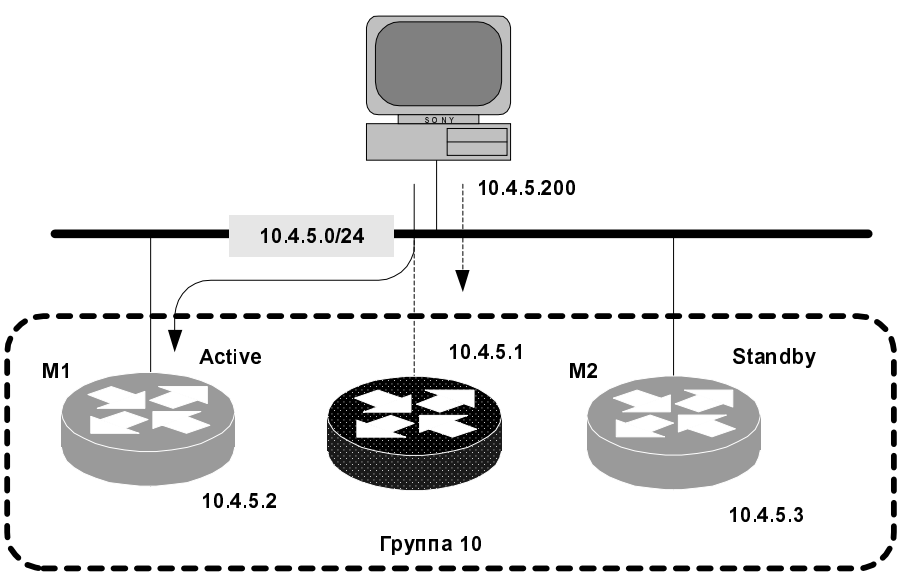


Рис. 3.15. Вариант построения резервирующей группы из двух маршрутизаторов

Состояния события и таймеры маршрутизаторов HSRP

Для обеспечения согласованного взаимодействия внутри резервирующей группы у всех маршрутизаторов, входящих в ее состав, должны быть соответствующим образом определены параметры протокола HSRP, представленные в табл. 3.5.

Таблица 3.5. *Параметры маршрутизаторов HSRP*

Параметр	Назначение
Standby group number	Порядковый номер резервирующей группы
Virtual MAC address	MAC-адрес виртуального маршрутизатора
Priority	Приоритет маршрутизатора
Authentication Data	Данные, используемые для аутентификации
Hello time	Период посылки сообщений Hello
Hold time	Интервал ожидания получения сообщения Hello

Порядковый номер присваивается резервирующей группе для того, чтобы обеспечить возможность одновременного определения нескольких Standby Group (резервирующих групп) на одном сегменте сети. В этом случае один маршрутизатор может являться членом нескольких резервирующих групп одновременно. Значение приоритета определяет относительную степень притязаний данного маршрутизатора на должность активного маршрутизатора резервирующей группы — чем выше приоритет, тем выше вероятность того, что маршрутизатор станет активным.

Таймер Hello timer определяет период формирования сообщений HELLO активным маршрутизатором резервирующей группы. Для маршрутизаторов Cisco Systems это значение по умолчанию составляет 3 секунды.

Значение таймера Hold timer определяет максимальный интервал времени, в течение которого пассивные маршрутизаторы резервирующей группы HSRP ожидают поступление сообщения HELLO от активного маршрутизатора их группы.

Маршрутизатор резервирующей группы может находиться в одном из шести возможных состояний, перечень которых представлен в табл. 3.6.

Таблица 3.6. *Состояния маршрутизаторов HSRP*

Обозначение	Состояние
Initial	Начальное состояние
Learn	Состояние обучения
Listen	Прослушивание сети
Speak	Трансляция сообщений
Standby	Резервный маршрутизатор
Active	Активный маршрутизатор

После включения интерфейса или протокола HSRP маршрутизатор находится в начальном состоянии Initial. В состоянии Learn маршрутизатор HSRP находится до получения первого сообщения HELLO от активного маршрутизатора. Когда маршрутизатор HSRP находится в состоянии Listen, он не является ни активным, ни запасным. В этом состоянии маршрутизатор прослушивает сообщения активных и запасных маршрутизаторов. Маршрутизатор HSRP, находящийся в состоянии Speak, посылает периодические сообщения HELLO для участия в выборах, в итоге которых он становится активным или запасным маршрутизатором.

Каждый маршрутизатор HSRP в процессе работы использует значения трех таймеров:

- Active timer — для мониторинга состояния активного маршрутизатора;
- Standby timer — для мониторинга состояния резервного маршрутизатора;
- Hello timer — для формирования сообщений HELLO.

Формат и типы пакетов HSRP

Для передачи сообщений протокола HSRP используются дейтаграммы протокола UDP (порт 1985). Для адресации сообщений применяется групповой адрес 224.0.0.2. Структура информационной части пакета HSRP представлена на рис. 3.16.

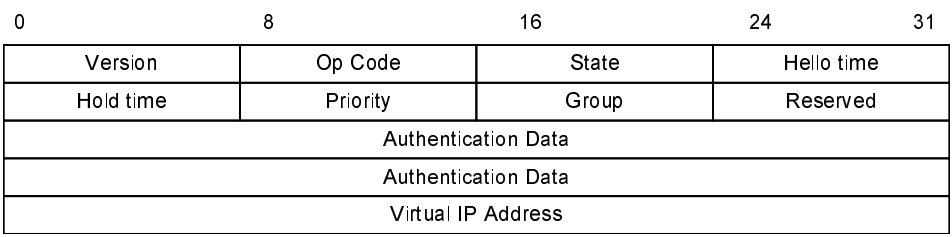


Рис. 3.16. Структура информационной части пакета HSRP

В поле **Version** размещается номер версии протокола HSRP. Содержимое поля **Op Code** определяет тип сообщения HSRP:

- **Op Code 0** — соответствует сообщению Hello;
- **Op Code 1** — соответствует сообщению Coup;
- **Op Code 2** — соответствует сообщению Resign.

Назначение сообщений Hello было достаточно подробно описано выше. Сообщение Coup (заявка) маршрутизатор передает в том случае, если собирается занять место активного маршрутизатора. Если наоборот, маршрутизатор не желает больше быть активным, он должен передать сообщение Resign (отставка). В поле **State** размещается кодировка, определяющая текущее состояние маршрутизатора HSRP, сформировавшего это сообщение. Содержимое полей **Hello time** и **Hold time** имеет значение только в сообщениях HELLO. Если в конфигурации маршрутизатора HSRP значение интервалов **Hello time** и **Hold time** не указано явно, он должен использовать значения, полученные в сообщении HELLO от активного маршрутизатора. В противном случае маршрутизатор должен использовать значения, установленные по умолчанию для **Hello time** и **Hold time** — 3 и 10 секунд, соответственно. В поле **Authentication Data** помещается открытый пароль, используемый для аутентификации участников резервирующей группы. В качестве значения, используемого по умолчанию, RFC 2281 рекомендует использование 0x63 0x69 0x73 0x63 0x6F 0x00 0x00 0x00 ("cisco___").

Взаимодействие HSRP с протоколами ARP и ICMP

Для обеспечения нормального функционирования протокола HSRP каждая группа резервирующих маршрутизаторов должна использовать уникальный виртуальный MAC-адрес. Структура виртуального MAC-адреса определяется типом протокола канального уровня. Для сетей IEEE 802.3 рекомендуется использование виртуального адреса 0x00 0x00 0x0C 0x07 0xAC XX, где XX — соответствует номеру резервирующей группы.

Активный маршрутизатор должен выполнять функции proxy — ARP-сервера в отношении виртуального маршрутизатора. Если активный маршрутизатор принимает ARP-запрос на виртуальный IP-адрес своей резервирующей группы, он должен сформировать ответ, где он указывает соответствующий виртуальный MAC-адрес своей группы.

Особенностью протокола HSRP является требование блокирования функции ICMP Redirect на маршрутизаторах резервирующей группы. Очевидно, что несогласованное использование IP-адресов различными маршрутизаторами из этой группы может нарушить ее правильное функционирование.

Протокол VRRP

Протокол VRRP (Virtual Router Redundancy Protocol) предназначен для выполнения тех же функций, что и "фирменные" протоколы HSRP Cisco Systems Inc. и IP Standby Protocol (IPSTB) Digital Equipment Corporation, Inc. и, является, по сути, их открытым аналогом. Протокол VRRP, в отличие от протокола HSRP Cisco Systems, имеет более развитую систему аутентификации участников резервирующей группы и обеспечивает возможность использования протокола IP Authentication Header (AUTH) для установления подлинности абонентов. Протокол VRRP предусматривает также использование дополнительных процедур, исключающих возможность проведения перевыборов основного (MASTER) маршрутизатора при его нормальном функционировании.

Протокол VRRP так же, как и HSRP Cisco Systems, обеспечивает возможность построения гибких и эффективных схем резервирования с множественным участием маршрутизаторов в резервирующих группах. Рассмотрим схему, представленную на рис. 3.17.

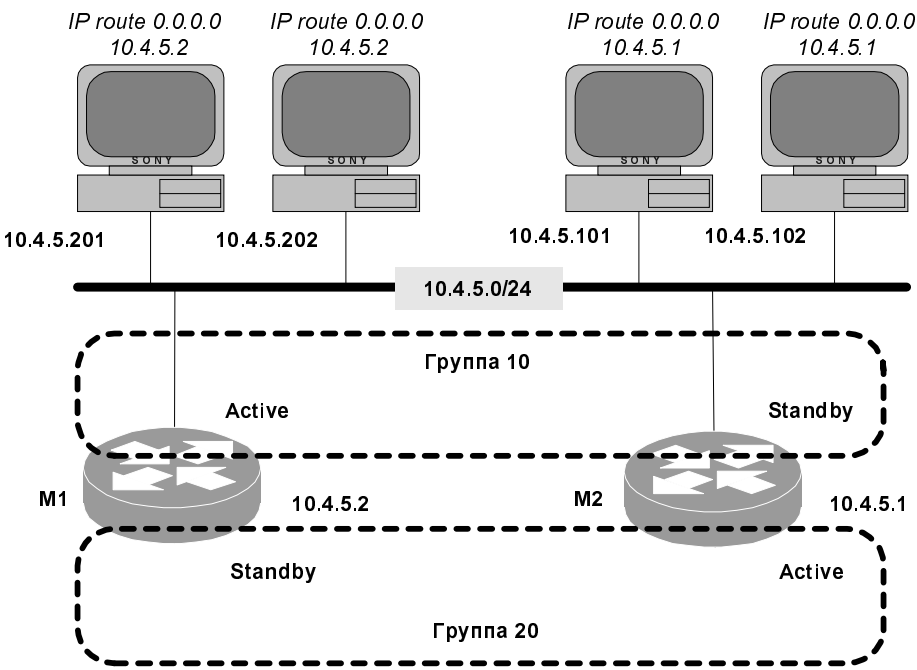


Рис 3.17. Множественное вхождение маршрутизаторов в резервирующие группы

В данном случае маршрутизатор M1 одновременно является основным для узлов 10.4.5.201 и 10.4.5.202 и резервным для 10.4.5.101 и 10.4.5.102.

Маршрутизатор М2 выполняет зеркально противоположные функции. Очевидно, что такая схема включения, помимо обеспечения динамического резервирования, позволяет значительно более эффективно использовать ресурсы сетевых устройств, поскольку резервные маршрутизаторы не простаивают, а обрабатывают трафик наравне с основными маршрутизаторами.

Протокол SNMP

Протокол SNMP (Simple Network Management Protocol, Простой протокол управления сетью) предназначен для обеспечения управления и контроля сетевых узлов. Он позволяет управляющим станциям собирать информацию о "положении дел" в подконтрольной сети.

Сегодня SNMP является самым популярным протоколом управления различными коммерческими, университетскими и исследовательскими объединенными сетями. SNMP представляет собой протокол прикладного уровня, предназначенный для обеспечения обмена управляющей информацией между сетевыми устройствами. Пользуясь информацией, предоставляемой SNMP-протоколом, сетевые администраторы могут более оперативно и просто управлять производительностью сети, обнаруживать и решать многие проблемы, возникающие в процессе эксплуатации сети.

В создание протокола SNMP внесли свой вклад разработки по трем направлениям.

1. *HEMS (High-level Entity Management System).* HEMS — система управления объектами высшего уровня. Она предназначалась для определения принципов и механизмов управления сетями с нестандартными техническими характеристиками, управляющими информационными базами данных объектов, механизмами доступа и системой построения отчетной информации, в том числе и языками построения запросов с использованием механизмов RPC. К сожалению, система HEMS использовалась только в местах ее разработки, что и привело к прекращению ее развития.
2. *SGMP (Simple Gateway Monitoring Protocol)* — Простой протокол управления шлюзом. Первоначально этот протокол предназначался только для управления компонентами сети. Разработка была выполнена инициативной группой сетевых инженеров для обеспечения решения проблем управления быстро растущей сети Интернет. Результатом их усилий стал протокол, предназначенный для управления маршрутизаторами в сети Интернет. В настоящее время протокол SGMP реализован во многих региональных ветвях Интернета.
3. *CMOT (CMIP over TCP).* Данный протокол обеспечивает управление компонентами сети с использованием стандартных протоколов OSI, в частности, протокола CMIP (Common Management Information

Protocol — протокол общего управления). Протокол CMIP предназначен для облегчения управления объединенных сетей TCP.

В 1987 году при решении проблемы стандартизации механизмов управления все эти три направления активно обсуждались. Для формирования единой спецификации протокола сетевого управления в начале 1988 года под руководством IAB (Internet Activities Board — группа, ответственная за техническую разработку протоколов Интернета) были образованы две рабочие группы.

Одна группа стала заниматься спецификацией и разработкой элементов информационной базы управления — MIB (Management Information Base). В дальнейшем работа по этому направлению вылилась в создание структуры управления объектами — SMI (Structure for Management Information).

Другая группа, занимавшаяся развитием протокола управления, пришла к выводу, что временным решением проблемы управления может стать улучшенная версия протокола SGMP (Simple Network Management Protocol), которая позже была названа SNMP (Simple Network Management Protocol). Для долгосрочного применения, после глубоких и обстоятельных доработок, должна была использоваться одна из технологий, базирующихся на OSI (либо CMOT, либо сам CMIP).

Кроме того, было решено во всех системах управления сетями использовать спецификацию ASN.1 (Abstract Syntax Notation One, Спецификация синтаксиса номер один), которая представляет собой универсальный язык описания форматов структур управления. Эта спецификация в настоящее время используется для описания структур и элементов MIB.

Доработка спецификации протокола SNMP продолжается по мере того, как разрабатываются современные прикладные программы управления, основанные на использовании этого протокола.

Несмотря на то, что SNMP является относительно простым протоколом, набор его характеристик является достаточно мощным для решения большинства проблем, возникающих при управлении гетерогенных сетей.

Принципы построения системы управления сетью

Система управления сетью на основе протокола SNMP представляет собой комплекс управляющих станций сети и управляемых объектов. На управляющих станциях выполняются приложения, которые контролируют и управляют действиями "агентов" SNMP, расположенных на сетевых элементах управляемой сети. Сетевыми элементами являются такие объекты, как оконечные узлы, шлюзы, терминальные серверы и другие объекты, на которых могут выполняться модули сетевых агентов.

Агентами в SNMP называются специальные программные модули, размещенные на управляемых объектах — элементах сети. Агенты собирают информацию об управляемых объектах и размещают ее в своих локальных

переменных, которые являются составляющими элементами единой информационной базы данных управления сетью — MIB (Management Information Base).

Протокол SNMP используется для обмена управляющей и контрольной информацией между управляющими станциями и элементами сети. Таким образом, с помощью протокола SNMP локальная информация становится доступной для систем управления сетями NMS (Network Management Systems) и системы управления базой данных MIB.

Управляемый объект может быть узлом любого типа, находящимся в какой-либо сети: это может быть узел, служебное устройство связи, принтер, роутер, мост или концентратор. Некоторые из этих систем могут иметь ограниченные способности управления программным обеспечением, например, они могут иметь процессоры с относительно малым быстродействием или ограниченный объем памяти. Поэтому программы управления должны быть построены таким образом, чтобы минимизировать влияние на управляемый объект, с тем чтобы основная нагрузка при выполнении процедур управления ложилась на управляющие станции.

В качестве управляющих станций NMS обычно используются достаточно мощные компьютеры, которые имеют быстродействующие центральные процессоры, значительный объем накопителей и высокопроизводительные периферийные устройства. В управляемой сети может иметься одна или более NMS-станций. На NMS-станциях выполняются прикладные программы сетевого управления, предназначенные для сбора и обработки информации об объектах управления.

На рис. 3.18 показана общая схема работы системы управления сетью NMS.

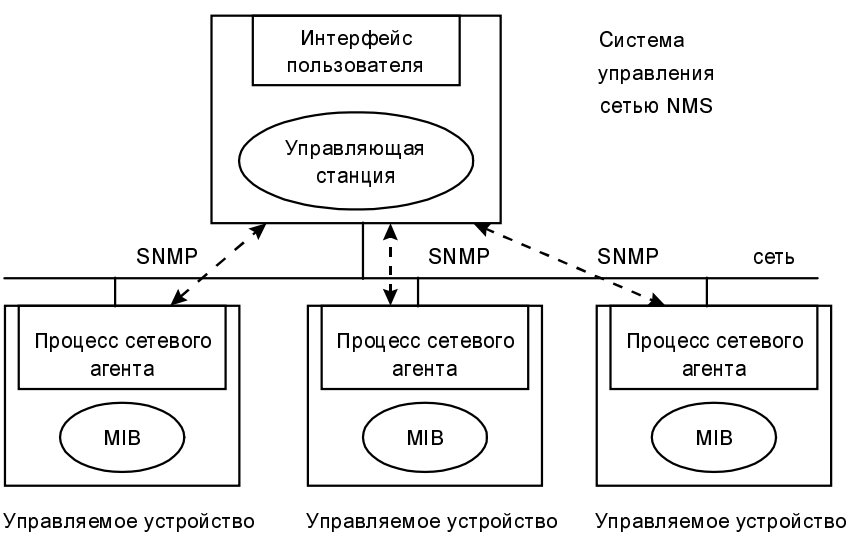


Рис. 3.18. Управление сетью NMS

Глобальное дерево имен

Все управляемые объекты глобальной сети расположены в едином виртуальном "дереве" объектов — Информационной базе управления (MIB — Management Information Base), "листьями" которого являются отдельные информационные элементы. Синтаксис описания "дерева" определяется спецификацией ASN.1.

Каждый объект "дерева" имеет имя, которое отражает функции объекта и идентификатор. Идентификатор представляет собой последовательность целых чисел, отмечающих путь к объекту от "корня дерева" (рис. 3.19).

Корень "дерева" не имеет идентификатора. Первый уровень от корня составляют объекты: itu — он администрирует поддерево 0; iso — администрирует "дерево", спускающееся из узла 1; joint-iso-itu — узел 2. Во втором уровне основного "дерева" расположен организационный узел — org (3), который относится к поддереву iso. Этот узел был определен ISO для делегирования ответственности за продолжение "дерева" на другие организации. Одной из таких организаций третьего уровня является dod (6) (Министерство обороны США), которому, согласно иерархии управления, и принадлежит пакетная сеть Интернет (1), расположенная на четвертом уровне глобального "дерева" объектов управления.

Таким образом, согласно этой иерархии, идентификаторы объектов Интернета начинаются с кода 1.3.6.1. Непосредственно под узлом Интернета расположены 6 узлов: directory (справочник), mgmt (управление), experimental (экспериментальный), private (частный), security (безопасность) и snmp-2.

Например, узел private имеет в настоящее время только одно поддерево — enterprises (предприятия). "Ветка" enterprises состоит из более 1000 узлов, которые занимают различные поставщики оборудования, разработчики программного обеспечения, госучреждения, университеты и др. Каждый из этих узлов позволяет организации определить идентификаторы продуктов и структуры, необходимые для управления этими продуктами. Например, узел 348 (1.3.6.1.4.1.348) занимает Procter&Gamble, а узел 743 (1.3.6.1.4.1.743) — ЦРУ.

"Дерево" mgmt содержит переменные сетевого управления. Например, системная группа ip (4), "ветви" mib-2 (1) "дерева" mgmt (2), идентификатор которой 1.3.6.1.2.1.4, дает информацию, необходимую для управления компьютерами и маршрутизаторами.

"Дерево" MIB постоянно расширяется, благодаря экспериментальным и частным "ветвям". Например, поставщики программного обеспечения могут определять свои собственные "ветви" для своих изделий. В настоящее время вся работа по стандартизации ведется на экспериментальной "ветви".

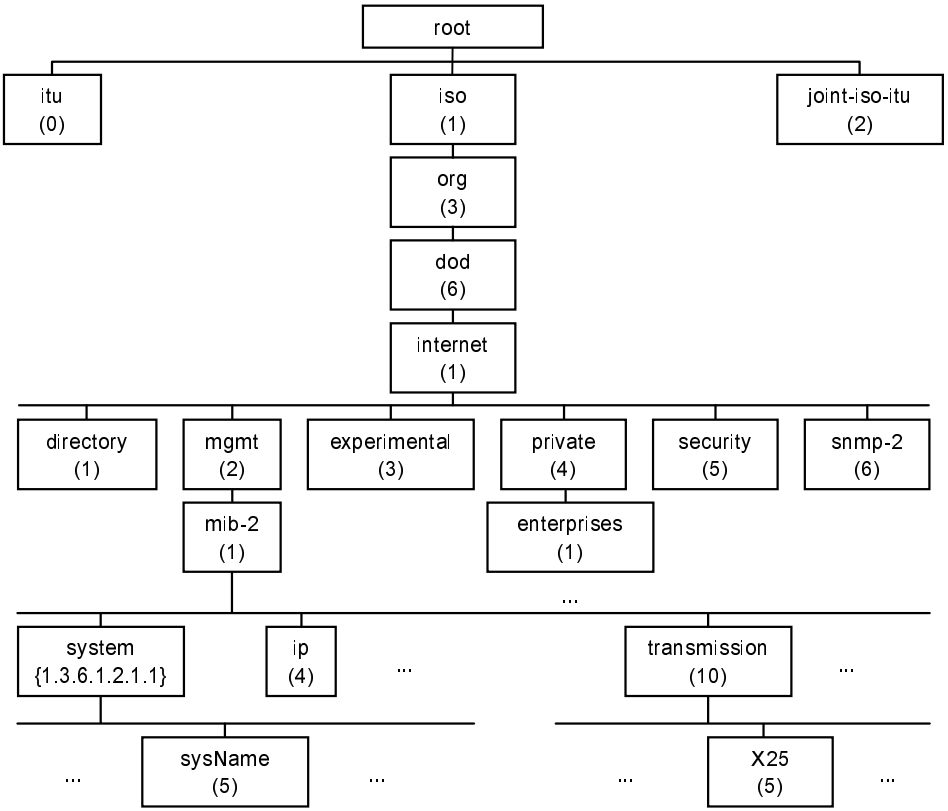


Рис. 3.19. Часть глобального "дерева" имен

С точки зрения управления сетью, задачей глобального "дерева" имен является обеспечение стандартного метода определения (чтения) и формирования (записи) значений параметров управляемых объектов. При этом некоторые объекты могут предоставлять информацию по нескольким пунктам, например, мост или маршрутизатор с несколькими интерфейсами должен предоставить информацию по каждому из своих интерфейсов.

Зная IP-адрес управляемого объекта и положение в глобальном "дереве" имен идентификаторов этого объекта, вы можете сформировать новое значение параметра (или параметров) объекта или определить текущее значение этих параметров, если, конечно, этот объект присутствует в "дереве" имен. Любая из операций с объектом "дерева" выполняется только в том случае, если этот тип операции (чтение/запись) поддерживается определенным для данного объекта методом доступа.

Расположение переменных объекта на глобальном "дереве" определяется его функциональными характеристиками, в то время, как его IP-адрес иденти-

фицирует положение этого объекта в сети. Если какой-либо объект многофункционален — его состояние определяется переменными, расположенными в разных "ветвях" "дерева".

В соответствии со спецификацией SMI, для полного описания объекта, необходимо определить следующее:

1. Текстовое имя "ветви" (или "листа") глобального "дерева", которому соответствует объект.
2. Тип объекта. Объект может быть одного из следующих типов или их комбинацией.
   - *Network addresses (Сетевые адреса)* представляют какой-нибудь адрес из конкретного семейства протоколов. В настоящее время единственным примером сетевых адресов являются 32-битные адреса IP. Хотя этот тип данных может использоваться и для других типов адресов.
   - *IpAddress (IP-адреса)* представляют собой 32-битный Интернет-адрес. Адрес представляется в виде последовательности четырех байт.
   - *Counters (Счетчики)* — неотрицательные целые числа, которые монотонно увеличиваются до тех пор, пока не достигнут максимального значения, после чего они сбрасываются до нуля. Примером счетчика является общее число байтов, принятых интерфейсом.
   - *Gauges (Измерители, мера, размер)* — неотрицательные целые числа, которые могут увеличиваться или уменьшаться и фиксируются при достижении максимальных значений. Примером измерителя является длина очереди, состоящей из выходных пакетов.
   - *Ticks (Тики)* — сотые доли секунды, прошедшие после какого-нибудь события. Примером реализации данного типа является время, прошедшее после перехода интерфейса в некоторое иное состояние.
   - *Opaque (Произвольный)* — произвольное кодирование. Используется для передачи произвольных информационных последовательностей, находящихся вне пределов точного определения данных, которое использует SMI.

Определение объекта. Описание должно содержать специфические характеристики объекта, особенности его применения.

Доступ. Этот параметр характеризует тип доступа к объекту: read-only, read-write, write-only или not-accessible.

Статус объекта: обязательный, необязательный или устаревший.

Например, описание объекта глобального "дерева" управления может выглядеть следующим образом:

1. Имя: sysUpTime {system 3}, идентификатор — 1.3.6.1.2.1.1.3.

2. Тип: Ticks.
3. Определение: Обозначает время (в сотых долях секунды), прошедшее с последней инициализации подсистемы управления сетью.
4. Доступ: read-only (только на чтение).
5. Статус: обязательный (т. е. объект должен обязательно присутствовать в системе).

или

1. Имя: sysDescr {system 1}, идентификатор — 1.3.6.1.2.1.1.1.
2. Тип: Opaque (строка).
3. Определение: Содержит описание объекта. Эта переменная должна содержать полное имя, версию объекта, тип аппаратуры, программного и сетевого обеспечения. Может содержать только печатные ASCII-символы.
4. Доступ: read-only (только на чтение).
5. Статус: обязательный.

Доступ к переменным MIB-базы осуществляется по их идентификаторам и переменным, определяющим конкретное "воплощение" переменной в объекте или объектах. Параметры функции задаются в формате "variable.parametrs". Каждый объект имеет свои параметры, но способ их передачи — единый для всех объектов.

В некотором смысле, объекты MIB-базы можно назвать функциями своих переменных, поскольку они осуществляют "отображение" множества объектов реальной сети во множество параметров этих объектов.

Например, мы хотим определить сетевую маску записей таблицы интерфейса с IP-адресом 89.1.1.42. Для этого мы должны обратиться к MIB-объекту ipAdEntNetMask и задать ему в качестве параметра IP-адрес интерфейса, например: *ipAdEntNetMask.89.1.1.42* — данная строка определяет искомый параметр.

Другой пример. Нам нужно определить состояние TCP-соединения между локальным адресом 89.1.1.42 TCP-порт 21 и удаленным адресом 10.0.0.51 TCP-порт 2059. Запрос о состоянии объекта будет выглядеть так: *tcpConnState.89.1.1.42.21.10.0.0.51.2059.*

Взаимодействие компонентов протокола SNMP

Как уже отмечалось, SNMP представляет собой протокол прикладного уровня, с помощью которого можно контролировать и изменять состояние узлов сети. Для передачи сообщений протокола SNMP используется протокол транспортного уровня UDP.

Каждое сообщение протокола SNMP является самостоятельным и передается независимо от других сообщений. Таким образом, взаимодействие SNMP-агентов осуществляется посредством обмена сообщений, каждое из которых представляет собой самостоятельную UDP-дейтаграмму.

Большинство операций протокола SNMP выполняется для чтения или записи управляющей информации. Обращение за информацией осуществляется напрямую к агенту ресурса, являющемуся его хозяином. При этом допускается, что часть запросов не будет обработана, например, по причине отсутствия запрашиваемых параметров. В таких случаях агент должен возвратить запрашивающей системе уведомление о невозможности получения запрошенной информации.

Примечание

Дейтаграммы UDP, содержащие обычные сообщения SNMP, адресуются в порт 161. Служебные сообщения SNMP (уведомления — TrapPDU) принимаются на порт 162. Термин PDU (Protocol Data Units) в данном случае используется для обозначения сообщений протокола SNMP.

Сообщения протокола SNMP включают в себя следующие части: идентификатор версии протокола (version), имя сообщества (community name), передаваемые данные (data) и параметров аутентификации.

Имя сообщества определяет границы сетевой области, в пределах которой SNMP-агенты находятся под единым административным управлением. *Сообщество* — это, по сути, набор правил, которым подчиняется выделенная группа SNMP-агентов. Информационная часть сообщения содержит указание о типе выполняемой протоколом SNMP операции (get, set и т. д.) и связанные с ней операнды, определяющие в совокупности данную транзакцию SNMP.

В SNMP-протоколе определены четыре операции.

- *Get (извлечь)* — извлекает какой-нибудь параметр объекта из агента.
- *Get-next (достань следующий)* — извлекает следующий параметр объекта из таблицы или перечня, находящихся в агенте.
- *Set (установка)* — устанавливает значение параметра объекта в пределах данного агента.
- *Trap (фиксатор)* — используется агентом для асинхронного информирования системы NMS о каком-нибудь событии.

Для передачи сообщения SNMP-агент должен выполнить следующие действия:

1. Сформировать PDU для запрашиваемого объекта.
2. Скомпоновать получившийся объект с именем сообщества, транспортным адресом отправителя и получателя и передать его сервису аутентификации.
3. Сервис аутентификации возвратит объект в виде ASN.1-сообщения.

4. Полученный объект разбить на пакеты и переправить их по адресу назначения.

Процедура приема SNMP-сообщения включает практически все описанные выше действия, выполняемые в обратном порядке.

Примечание

Особенность обработки SNMP-дейтаграмм заключается в том, что при отсутствии согласования параметров принятой дейтаграммы (например, номера версии) она немедленно уничтожается.

Блок данных протокола SNMP, обеспечивающий выполнение стандартных операций (запрос/ответ) включает в себя следующие объекты.

- *Request-ID* — идентификатор запроса. Он используется для ссылок ответных сообщений на данный запрос.
- *Error-Status* — содержит код ошибки обработки запроса: *NoError* — нет ошибки, *tooBig* — слишком длинное значение переменной, *noSuchName* — неправильно задано имя, *badValue* — неправильный идентификатор, *readOnle* — нарушен метод доступа, *genErr* — другие.
- *Error-index* — содержит индекс объекта, вызвавшего ошибку.
- *Variable bindings* — переменные привязки, которые состоят из данных SNMP PDU. Переменные привязки устанавливают связь между конкретными переменными и их текущими значениями. Список состоит из пар: имя — параметр.

Структура данных протокола SNMP, обеспечивающего передачу сообщений уведомления (PDU-Trap), отличается тем, что содержит дополнительные поля для более детальной идентификации объекта, изменившего свое состояние.

При обмене сообщениями SNMP-агенты используют механизмы аутентификации отправителя и получателя сообщений. Эти механизмы представляют собой наборы правил, по которым модули отдельных сообществ идентифицируют друг друга. Некоторые реализации SNMP используют простейшие схемы, некоторые — очень сложные с высокой степенью защиты.

В качестве примера рассмотрим последовательность SNMP-сообщений обеспечивающих чтение таблицы маршрутизации некоторого узла сети. Содержание передаваемой маршрутной информации представлено в табл. 3.7.

Таблица 3.7. *Маршрутная информация, передаваемая протоколом SMNP*

Destination	NextHop	Metric
9.1.2.3	99.0.0.3	3
10.0.0.51	89.1.1.42	5

В данном случае последовательность запросов (→) и ответов (←) будет выглядеть следующим образом:

```
→ GetRequest (ipRouteDest, ipRouteNextHop, ipRouteMetric1)

← GetResponse ((ipRouteDest.9.1.2.3 = "9.1.2.3"),
               (ipRouteNextHop.9.1.2.3 = "99.0.0.3"),
               (ipRouteMetric1.9.1.2.3 = 3))

→ GetNextRequest (ipRouteDest.9.1.2.3,
                  ipRouteNextHop.9.1.2.3,
                  ipRouteMetric1.9.1.2.3)

← GetResponse ((ipRouteDest.10.0.0.51 = "10.0.0.51"),
               (ipRouteNextHop.10.0.0.51 = "89.1.1.42"),
               (ipRouteMetric1.10.0.0.51 = 5))
```

Протокол RPC

RPC (Remote Procedure Call, Сервис вызова удаленных процедур) представляет интерфейс между удаленными пользователями и определенными программами узла, которые выполняются по запросам этих пользователей. Каждая из таких программ состоит, в свою очередь, из одной или нескольких удаленных процедур. Например, сервис удаленной файловой системы NFS (Network File System), который построен на вызовах RPC, может состоять только из двух программ: одна программа взаимодействует с высокоуровневыми пользовательскими интерфейсами, а другая — с низкоуровневыми функциями ввода-вывода.

В каждом вызове удаленной процедуры участвуют две стороны: активный клиент, который отправляет запрос вызова процедуры на сервер, и сервер, который отправляет клиенту ответ.

Примечание

Следует иметь в виду, что термины "клиент" и "сервер" в данном случае употребляются применительно к конкретной транзакции. Сетевой узел или программное обеспечение (процесс или программа) при этом могут выполнять функцию клиента или сервера. Например, программа, обеспечивающая сервис удаленных процедур, в то же время может быть клиентом при взаимодействии с сетевой файловой системой.

Протокол RPC построен на модели вызовов удаленных процедур, который аналогичен механизму вызовов локальных процедур. При вызове локальной процедуры вы помещаете аргументы в определенное место памяти, в стек

или переменные окружения и передаете управление процессом по определенному адресу. После завершения работы вы читаете результаты по конкретному адресу и продолжаете свой процесс.

В случае работы с удаленной процедурой основное отличие состоит в том, что вызов удаленной функции обслуживают два процесса: клиентский процесс и серверный процесс.

Процесс клиента отправляет серверу сообщение, в которое включены параметры вызываемой процедуры, и ожидает ответного сообщения с результатами ее выполнения. Со стороны сервера процесс-обработчик вызовов находится в состоянии ожидания и при поступлении сообщения считывает параметры процедуры, выполняет ее, отправляет ответ и переходит в состояние ожидания следующего вызова.

RPC-протокол не накладывает каких-либо ограничений на дополнительные связи между процессами и не требует синхронности выполняемых функций, т. е. вызовы могут быть асинхронными и взаимно независимыми, так что клиент во время ожидания ответа может выполнять другие процедуры. Сервер RPC может выделять для каждой функции отдельный процесс или виртуальную машину, поэтому, не дожидаясь окончания работы предыдущих запросов, сразу же может принимать следующие.

Однако между вызовами локальных и удаленных процедур есть несколько важных отличий:

1. *Обработка ошибок.* Клиент в любом случае должен получать уведомление об ошибках, возникающих при вызовах удаленных процедур на сервере или в сети.
2. *Глобальные переменные.* Поскольку сервер не имеет доступа к адресному пространству клиента, при вызовах удаленных процедур нельзя использовать скрытые параметры в виде глобальных переменных.
3. *Производительность.* Скорость выполнения удаленных процедур, как правило, на один или два порядка ниже скорости выполнения аналогичных локальных процедур.
4. *Аутентификация.* Поскольку вызовы удаленных процедур происходят по сети, необходимо использовать механизмы аутентификации клиента.

Принципы построения протокола

Протокол RPC является протоколом сеансового уровня и может использовать несколько различных транспортных протоколов. В обязанности RPC-протокола входит только обеспечение стандартов и интерпретация передачи сообщений. Достоверность и надежность передачи сообщений целиком обеспечивается транспортным уровнем. Однако RPC может контролировать выбор и некоторые функции транспортного протокола. В качестве примера

взаимодействия между RPC и транспортным протоколом рассмотрим процедуру назначения RPC-порта работы прикладного процесса через RPC — *Portmapper*.

Эта функция динамически (по запросу) назначает соединению RPC определенный порт. Функция *Portmapper* используется довольно часто, поскольку набор зарезервированных для RPC транспортных портов ограничен, а количество обслуживаемых процессов обычно достаточно велико. *Portmapper*, например, вызывается при выборе портов взаимодействия клиента и сервера системы NFS. Сервис *Portmapper* использует механизм широковещательных сообщений RPC на определенный порт — 111. На этот порт клиент отправляет широковещательное сообщение запроса порта определенного сервиса RPC. Сервис *Portmapper* обрабатывает такое сообщение, определяет адрес локального сервиса RPC и отправляет клиенту ответ. Сервис RPC *Portmapper* может работать как с TCP, так и с UDP-протоколами.

RPC может работать с различными транспортными протоколами, но никогда не дублирует их функции, т. е. если RPC работает поверх TCP, все заботы о надежности и достоверности соединения RPC возлагает на TCP. Однако если протокол RPC установлен поверх UDP, он может обеспечивать дополнительные собственные функции обеспечения гарантированной доставки сообщений.

Примечание

Прикладные задачи могут рассматривать протокол RPC как определенную процедуру вызова функции по сети JSR (Jump Subroutine Instruction).

Для корректного функционирования протокола RPC необходимо выполнение следующих условий:

1. *Уникальная идентификация всех удаленно вызываемых процедур на данном узле.* RPC-запросы содержат три поля идентификаторов — номер удаленной программы (сервиса), номер версии удаленной программы и номер удаленной процедуры указанной программы. Номер программы назначается производителем сервиса, а номер процедуры указывает на конкретную функцию данного сервиса.
2. *Идентификация версии протокола RPC.* RPC-сообщения содержат поле версии протокола RPC. Идентификация версии протокола используется для согласования форматов передаваемых параметров при работе клиента с различными версиями протокола RPC.
3. *Предоставление механизмов аутентификации клиента на сервере.* Протокол RPC обеспечивает процедуру аутентификации клиента в сервисе и, в случае необходимости, при каждом запросе или отправке ответа клиенту. Кроме того, RPC позволяет использовать различные дополнительные механизмы безопасности.

RPC может использовать четыре типа механизмов аутентификации:

- AUTH_NULL — без использования аутентификации;
- AUTH_UNIX — аутентификация по стандарту UNIX;
- AUTH_SHORT — аутентификация по стандарту UNIX с собственной структурой кодирования;
- AUTH_DES — аутентификация по стандарту DES (Data Encryption Standard — Стандарт шифрования данных).

4. *Идентификация сообщений ответа на соответствующие запросы.* Ответные сообщения протокола RPC содержат идентификатор запроса, на основании которого они были построены. Этот идентификатор можно назвать идентификатором транзакции RPC-вызова. Данный механизм особенно необходим при работе в асинхронном режиме и при выполнении последовательности из нескольких RPC-вызовов.
5. *Идентификация ошибок протокола.* Все сетевые или серверные ошибки имеют уникальные идентификаторы, по которым каждый из участников соединения может определить причину сбоя в работе.

Структуры сообщений протокола

При передаче RPC-сообщений поверх транспортного протокола несколько RPC-сообщений могут располагаться внутри одного транспортного пакета. Для того чтобы отделять одно сообщение от другого, используется маркер записи (RM — Record Marker). Каждое RPC-сообщение "маркируется" ровно одним RM.

RPC-сообщение может состоять из нескольких фрагментов. Каждый фрагмент состоит из четырех байт заголовка и (от 0 до 2^{31}–1) данных. Первый бит заголовка указывает, является ли данный фрагмент последним, а остальные 31 бит указывают длину пакета данных.

Структура сообщения протокола RPC формально описана на языке описания и представления форматов данных — XDR (eXternal Data Representation, Представление внешних данных) с дополнениями, касающимися описания процедур. Можно даже сказать, что язык описания RPC является расширением XDR, дополненным работой с процедурами.

Структура сообщения протокола RPC выглядит следующим образом:

```
struct rpc_msg {
    unsigned int xid;
    union switch (msg_type mtype) {
    case CALL:
        call_body cbody;
    case REPLY:
```

```
        reply_body rbody;
    } body;
};
```

где `xid` — идентификатор текущей транзакции, `call_body` — пакет запроса, `reply_body` — пакет ответа. Структура запроса выглядит примерно так:

```
struct call_body {
    unsigned int rpcvers;
    unsigned int prog;
    unsigned int vers;
    unsigned int proc;
    opaque_auth cred;
    opaque_auth verf;
    /* procedure parameters */
};
```

Ответ (`reply_body`) может содержать либо структуру, передаваемую при возникновении ошибки (тогда она содержит код ошибки), либо структуру успешной обработки запроса (тогда она содержит возвращаемые данные).

Протокол TELNET

Пользователям и администраторам сетей протокол TELNET известен как сервис, позволяющий войти в удаленный компьютер (т. е. другой компьютер сети Интернет) и выполнять на нем программы.

Примечание

Сервис TELNET на заре развития сети Интернет составлял основу ее работы. Большинство пользователей работали с сетью только через сервис TELNET.

Сервис TELNET сегодня в основном используется для удаленного администрирования компьютерных и коммуникационных систем, а также в качестве средства доступа к интернет-узлам для работы с их приложениями.

TELNET — это протокол, который обеспечивает дуплексный, 8-битный канал для передачи данных между участниками соединения. Протокол TELNET позволяет рассматривать все удаленные терминалы как стандартные "сетевые виртуальные терминалы" строчного типа, а также обеспечивает возможность согласования более сложных функций (например, локальный или удаленный эхо-контроль, страничный режим, высота и ширина экрана и т. д.).

Основным предназначением протокола является обеспечение взаимодействия терминальных устройств и терминальных процессов. Помимо обычного взаимодействия терминал — удаленный процесс, протокол может использо-

ваться для обеспечения взаимодействия терминал-терминал и процесс-процесс.

Протокол TELNET хорошо опробован и широко распространен. Создано множество реализаций этого протокола для разнообразных операционных систем.

Принципы построения

TELNET-соединение реализуется с использованием протокола TCP. На прикладном уровне (на стороне пользователя) над TELNET находится либо программа поддержки реального терминала, либо прикладной процесс, который осуществляет терминальный доступ.

Протокол TELNET использует дуплексное, многопользовательское (т. е. одновременно с сервером могут работать несколько пользователей) соединение между портом пользователя и портом сервера (как правило, порт сервера — 23). При образовании соединения, порт сервера TELNET находится в пассивном, "слушающем" режиме.

Примечание

"Пользователем" в системе TELNET называется узел, к которому физически подсоединен пользовательский терминал. "Сервером" называется узел, который предоставляет сервис TELNET. В системах "терминал-терминал" и "процесс-процесс" под "пользователем" подразумевается узел — инициатор соединения.

Протокол TELNET использует три основные принципа. К ним относятся:

1. Принцип виртуальных терминалов — NVT (Network Virtual Terminal).
2. Принцип настраиваемых параметров.
3. Принцип симметрии терминалов и процессов

После того, как установлено TELNET-соединение, предполагается, что каждый участник работает как "Виртуальный сетевой терминал" — NVT. В данном случае NVT представляет собой мнимое устройство, обеспечивающее выполнение набора стандартных сетевых промежуточных функций обычного терминала. Использование виртуальных терминалов исключает необходимость для "серверов" и "пользователей" соединения хранить информацию об особенностях конфигураций, сетевых характеристиках и форматах представления данных, которая необходима для поддержания соединения.

Принцип настраиваемых параметров основан на том, что некоторые узлы способны предоставить дополнительный сервис, в дополнение к предусмотренному для NVT. Основой процедуры настройки параметров TELNET является использование механизма запросов для обеспечения возможности

использования тех или иных расширений стандартного сервиса. Если терминальное устройство желает использовать дополнительные возможности соединения, оно отправляет на противоположную сторону соединения запрос. Если данный запрос будет отвергнут, для пользователя это означает отсутствие поддержки запрашиваемого сервиса. Механизм согласования дополнительных параметров построен таким образом, что сначала противоположные стороны согласуют основные параметры и уже после этого обмениваются информацией о согласовании дополнительных параметров. Для согласования используются квитанции — флаги принятия (или непринятия) тех или иных опций. Например, квитанция `WILL XXX` может быть отправлена любой из сторон, которая предлагает использовать опцию `XXX`. Квитанции `DO XXX` и `DON'T XXX` обозначают положительный и отрицательный ответы соответственно. Подобным же образом квитанция `DO XXX` отправляется, чтобы запросить другую сторону о возможности использования опции `XXX`. Положительным и отрицательным ответами на запрос будут, соответственно, квитанции `WILL XXX` и `WON'T XXX`.

Принцип симметрии терминалов и процессов подразумевает отсутствие какой-либо иерархии и привилегий между участниками соединения.

Сетевые виртуальные терминалы — NVT

NVT (Network Virtual Terminal, Сетевой виртуальный терминал) представляет собой объект сети для ввода и вывода 7-битных US ASCII-символов. Все преобразования кодировки и согласования считаются локальными проблемами узла и не влияют на работу NVT. Каждый NVT имеет устройство ввода — "виртуальную клавиатуру" и устройство вывода — "виртуальный принтер" (как правило, "виртуальный принтер" обозначает дисплей).

Примечание

Например, если у терминала установлена опция "echo", это значит, что при вводе символа с NVT-клавиатуры, символ будет попадать как на другой конец соединения, так и на устройство отображения — NVT-принтер.

Поступившие данные выводятся на NVT-принтер в соответствии с очередностью их поступления и управляющими символами.

Устройство NVT-вывода — принтер, не имеет ограничений по ширине и длине страницы и может выводить все 95 US ASCII-символов (с 32 по 126).

Управляющие коды (с 0 до 31 и 127) и 128 дополнительных (128–255) имеют специальное назначение для NVT-принтера. В табл. 3.8 представлены общепринятые значения управляющих кодов протокола TELNET.

Таблица 3.8. *Общепринятые значения управляющих кодов протокола TELNET*

Общепринятое обозначение	Код	Значение
NULL (NUL)	0	Нет операции
Line Feed (LF)	10	Перенос курсора на следующую строку с сохранением позиции
Carriage Return (CR)	13	Перевод курсора в начало текущей строки

Комбинация символов CR + LF во всех системах NVT обозначает начало новой строки. Часть кодов может не поддерживаться NVT, но NVT должен, в зависимости от параметров настройки, предпринять по их получении определенные действия. В табл. 3.9 представлены коды управления положением курсора протокола TELNET.

Таблица 3.9. *Коды управления положением курсора протокола TELNET*

Общепринятое обозначение	Код	Значение
BELL (BEL)	7	Звуковой или видео сигнал
Back Space (BS)	8	Перенос курсора на предыдущую позицию (по направлению к началу строки)
Horizontal Tab (HT)	9	Перевод курсора на следующую позицию горизонтальной табуляции
Vertical Tab (VT)	11	Перевод курсора на следующую позицию вертикальной табуляции
Form Feed (FF)	12	Перевод курсора на начало следующей страницы с сохранением горизонтальной позиции

Остальные управляющие коды (0—31) не влияют на работу NVT-принтера.

Данные для передачи, поступающие от NVT-клавиатуры, накапливаются в буфере NVT локального узла до тех пор, пока не будет завершена строка, или не будет подан специальный сигнал форсирования передачи введенных в буфер данных.

Использование такого механизма является оправданным, поскольку, во-первых, во многих системах частые вызовы сетевых функций обходятся очень дорого; во-вторых, многие системы по завершении ввода строки производят

дополнительные действия, которые можно было бы согласовать с передачей данных по сети; в-третьих, этот способ передачи позволяет избежать передачи пользовательских ошибок при вводе отдельных команд. Если система передает данные без символов конца строки — для нее существуют методы форсированной передачи буфера ввода по специальному сигналу (например, определенной последовательности символов) или команде.

NVT-клавиатура (устройство ввода) должна иметь клавиши, позволяющие генерировать коды или последовательности всех 128 символов US ASCII без исключения (безусловно, часть из них может не восприниматься NVT-принтером, но NVT-клавиатура должна иметь такую возможность). Кроме того, NVT-клавиатура должна уметь генерировать управляющие коды, которые представляют собой команды TELNET. Ниже представлены основные команды управления потоками ввода-вывода протокола TELNET.

- `IP` (Interrupt Process, Прервать процесс), код 244. Эта команда останавливает операции или процесс пользователя. Она используется при "зависании" или ошибочных действиях процесса.
- `AO` (Abort Output, Прервать вывод), код 245. Команда используется для принудительного завершения или остановки вывода. Она также очищает буфер вывода, который еще не выведен на принтер (или дисплей). Например, если во время вывода длинной строки была получена эта команда, строка прерывается символами CR + LF, что служит приглашением для ввода следующей команды пользователя (сессия TELNET при этом не закрывается).
- `AYT` (Are You There, Здесь ли ты?), код 246. Эта команда используется для проверки рабочего состояния системы, например, когда противоположная сторона длительное время "молчит".
- `EC` (Erase Character, Удаление символа), код 247. Команда используется для удаления последнего введенного символа, например, при ошибках ввода пользователя.
- `EL` (Erase Line, Удалить строку), код 248. Команда удаляет все данные из текущей строки ввода, до символов CR + LF.
- `GA` (Go Ahead, Далее), код 249. Эта команда передает контроль соединения противоположной стороне без передачи какой-либо информации.
- `SB` (Subnegotiation, Параметры расширения), код 250. Команда указывает, что следующий байт — параметр дополнительной опции.
- `SE` (End of subnegotiation, Конец параметров расширения), код 240. Команда сообщает о конце строки параметров дополнительных опций.
- `WILL`, `WON'T`, `DO`, `DON'T` — квитанции согласования дополнительных параметров работы, которые имеют, соответственно, коды 251, 252, 253, 254.

Все дополнительные и управляющие ключи в пользовательских системах должны быть соотнесены с определенными локальными кодами систем. Например, команда EC должена быть соотнесена с локальным символом удаления, а команда EL — с кодом удаления строки и т. д.

Описание протокола

Все команды TELNET состоят как минимум из двух байтов: управляющий символ IAC (Interpret as Command, Интерпретировать следующий байт как команду) — код 255, и следующий за ним байт команды. Команды TELNET, которые предназначены для согласования дополнительных параметров работы NVT, состоят из трех байтов (третий байт означает код настраиваемой опции).

Пусть, например, пользователь ошибочно ввел на локальном терминале символ и хочет его удалить.

- Для этого он нажимает клавишу <Backspace>.
- Код этого символа попадает на локальный NVT как код команды EC (Erase Character). Поскольку это управляющий символ — код передается немедленно.
- Далее по протоколу TELNET передается последовательность двух байт: "IAC EC" (255, 247). Теперь этот символ удален на другом конце TELNET-соединения.

Например, для согласования параметров дополнительной опции "ABC" TELNET использует следующую последовательность команд:

- IAC WILL ABC — терминал 1 (например, пользователь) предлагает использовать опцию "ABC".
- IAC DO ABC — терминал 2 (например, сервер) соглашается.
- IAC SB ABC <parameters> IAC SE — терминал 1 отправляет параметры.

Большинство систем в локальном режиме обрабатывают все сигналы ввода пользователя, однако, в случае работы через TELNET, не позволяют контролировать пользовательский процесс полностью (по причине загрузки сети, переполнения буфера команд процесса TELNET и др.). TELNET-протокол предоставляет дополнительный механизм контроля работы "зациклившихся" процессов — "SYNC".

В основе этого механизма лежит тот факт, что TCP-сегменты с флагом **URG** (Urgent — срочный; см. рис. 5.5) не обрабатываются TELNET-соединением, поэтому с их помощью можно обращаться к процессу, обрабатывающему команды TELNET напрямую, не используя буферы и сетевые средства TELNET, которые чаще всего являются причиной блокировки доступа к процессу.

TELNET — универсальный клиент

Сервис TELNET входит в состав практически всех сетевых операционных систем и реализован в виде оболочки с текстовым или графическим интерфейсом. При работе с сервисом TELNET, предоставляемым удаленным пользователям, утилита TELNET эмулирует на удаленном компьютере терминал пользователя узла сервиса TELNET и позволяет работать с ним как с локальным. Правила работы целиком определяются программным обеспечением удаленного узла, реализующим данный сервис.

На рис. 3.20 показана TELNET-сессия клиента с компьютером шлюза внутренней сети.

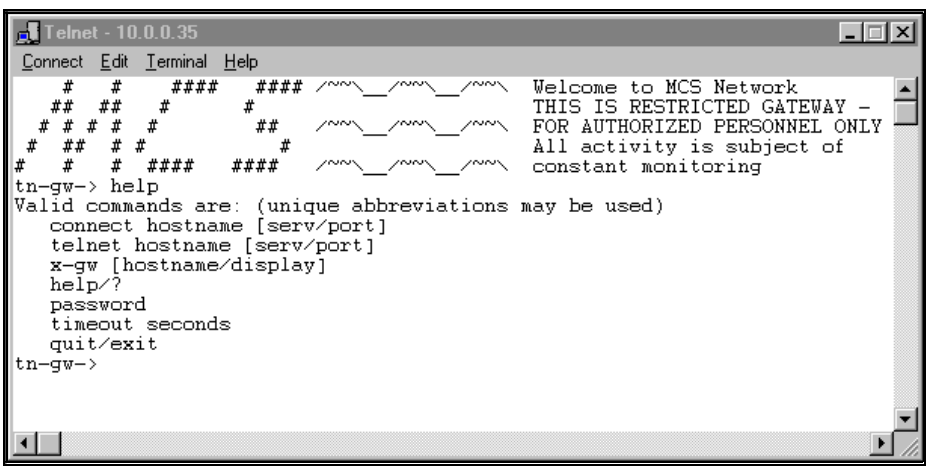


Рис. 3.20. TELNET-сессия клиента с компьютером шлюза внутренней сети

Как уже было отмечено, TELNET работает поверх TCP-протокола и предоставляет пользователю возможность передавать вводимые с клавиатуры данные на любой из портов удаленного узла, поддерживающего TCP-соединение. Например, используя TELNET, пользователь может подсоединиться к порту удаленного узла *daytime* (дата и время — порт 13 протокола TCP) и получить на экран строку текущей даты и времени этого узла (рис. 3.21).

На рис. 3.22 показана сессия TELNET, протокол TCP, порт 13.

На практике эта особенность обеспечивает возможность использования протокола TELNET для организации взаимодействия с любым приложением, управляемым командной строкой, например, приложений, реализующих протоколы: SMTP, POP3, IMAP4, NNTP, FTP, HTTP и др. Для пояснения приведем несколько примеров.

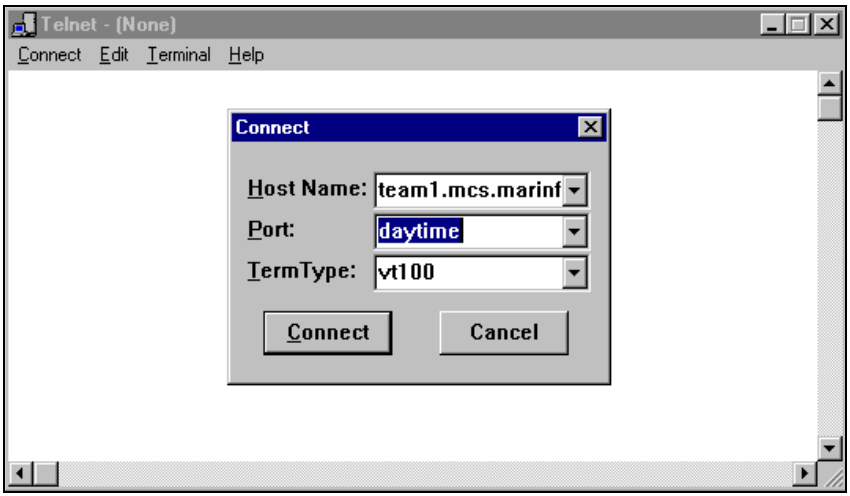


Рис. 3.21. Процедура соединения TELNET с узлом по протоколу TCP с помощью порта daytime

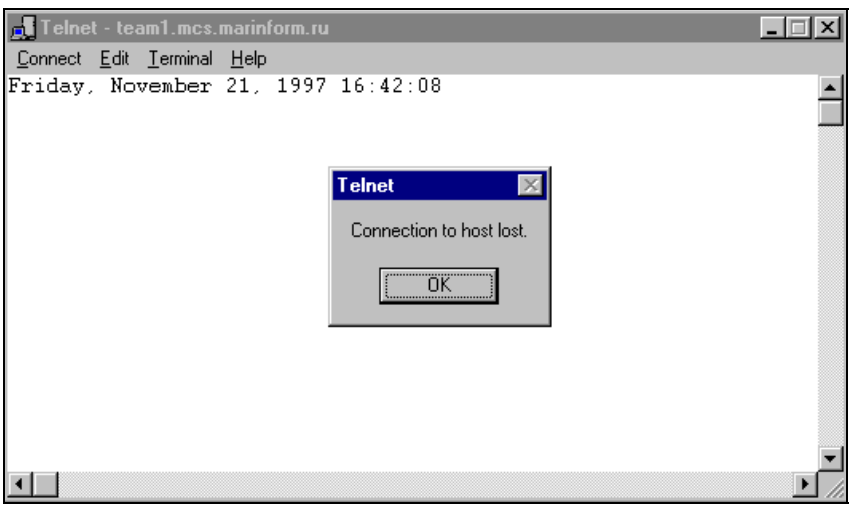


Рис. 3.22. Ответ узла при обращении к TCP-порту daytime

Предположим, что вам нужно срочно отправить небольшое e-mail-сообщение, а находящийся рядом компьютер не имеет программы почтового клиента. Тогда вы можете это сделать через утилиту TELNET. Для этого сначала подсоединитесь к SMTP-серверу, как это показано на рис. 3.23. Затем, воспользовавшись протоколом SMTP, отправьте ваше сообщение (рис. 3.24).

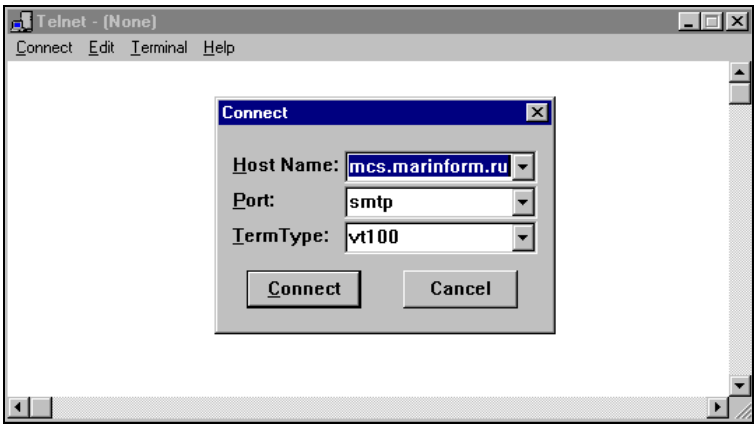


Рис. 3.23. Процедура подсоединения TELNET к серверу SMTP

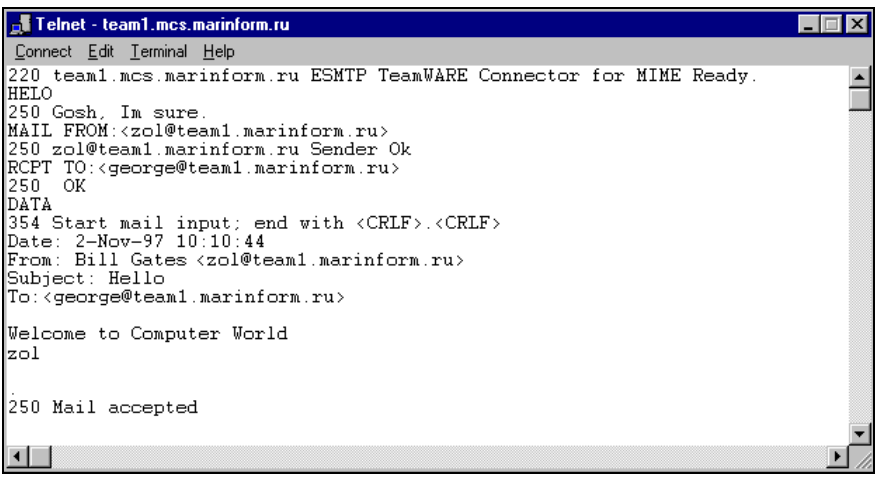


Рис. 3.24. SMTP-сессия отправки сообщения через TELNET

Для того чтобы прочитать с помощью TELNET поступившее сообщение, нужно знать, какой почтовой системой пользуется данная организация. От этого будет зависеть, каким протоколом вам предстоит воспользоваться. Например, если клиенты получают письма по POP3, вам нужно будет подсоединиться к POP3-серверу (порт 110), зарегистрироваться и... прочитать ваше сообщение (рис. 3.25).

Если интересующий вас почтовый сервер работает, например, с IMAP4 (Internet Message Access Protocol ver. 4), разница будет только в том, что вам нужно подсоединиться к IMAP4-порту (143) и воспользоваться правилами работы IMAP4-протокола (рис. 3.26).

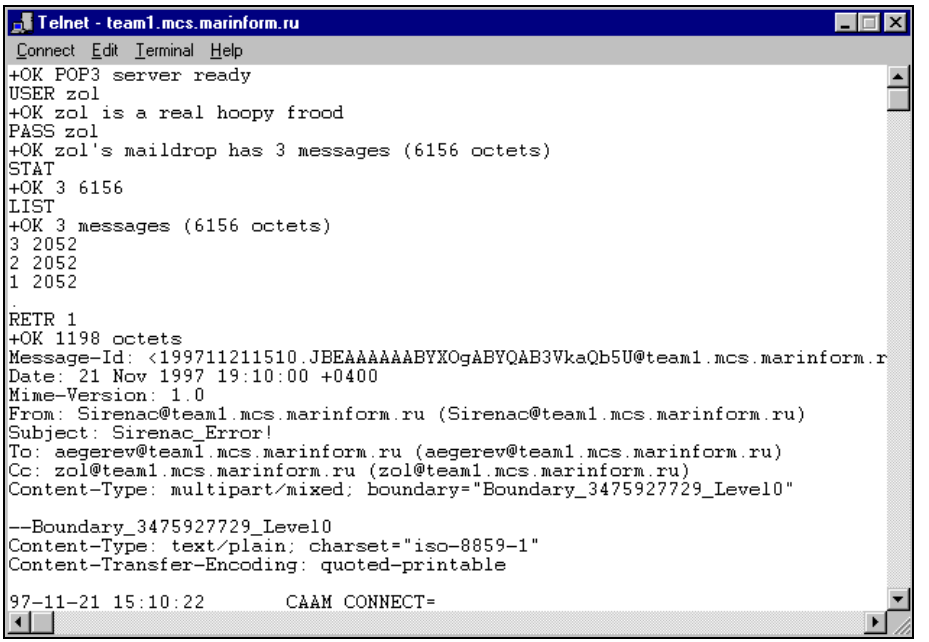


Рис. 3.25. POP3-сессия просмотра поступивших сообщений

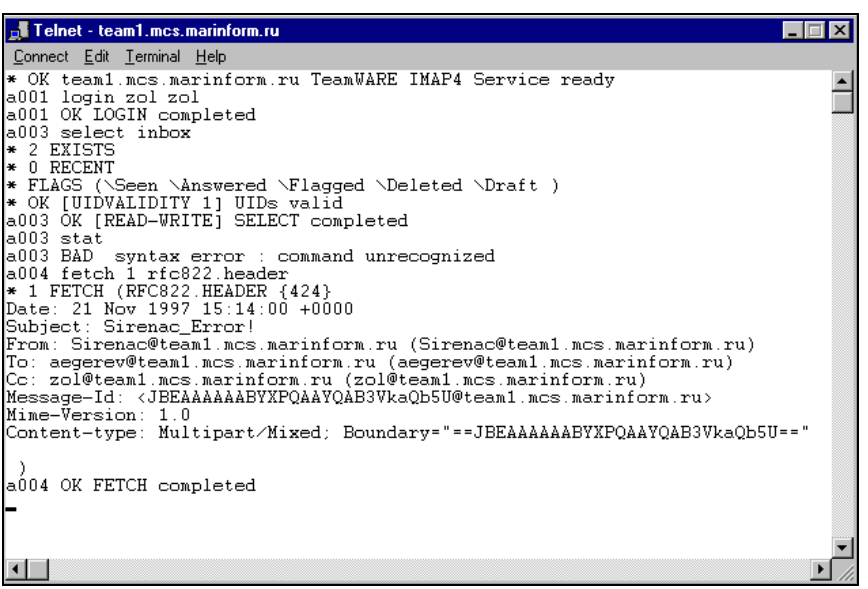


Рис. 3.26. IMAP4-сессия просмотра заголовков поступивших сообщений

Теперь, когда мы так быстро и просто расправились с почтой, можно почитать и статьи новостей USENET — с помощью той же утилиты TELNET и знаний NNTP-протокола (рис. 3.27).

Примечание

Прочитанная статья написана в кодировке KOI 8-R, а фонт утилиты TELNET был установлен для чтения статей в кодировке windows-1251, поэтому поле Subject содержит "нечитаемый текст". Если изменить шрифт экрана TELNET на KOI 8-R, тема статьи, заголовок которой изображен на рис. 3.27, будет следующей: "Re: пикапить? Re: долгий пронзительный взгляд (from FIDO)".

Безусловно, TELNET не может просматривать вставленные в статью рисунки или присоединенные файлы — для этого значительно быстрее и удобнее воспользоваться стандартными инструментами чтения статей новостей. Но с помощью утилиты TELNET пользователь может практически на любом компьютере (с минимальным программным обеспечением) быстро получить требуемую статью, прочитать или отправить простое письмо по электронной почте.

С помощью утилиты TELNET вы также можете строить HTTP-запросы и управлять передачей файлов через FTP или SFTP (поскольку все эти протоколы также работают через TCP).

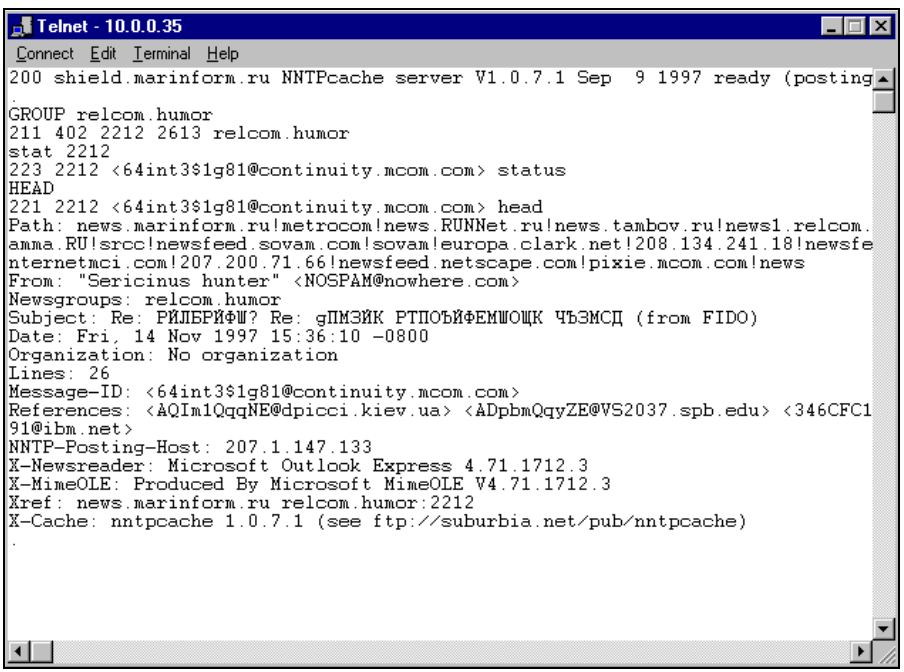


Рис. 3.27. Сессия NNTP и чтение заголовков статей конференции relcom.humor

Безусловно, TELNET не обеспечивает возможности просмотра графических изображений, запуска Java-приложения, интерпретации гипертекстовых файлов или передачи исполняемых файлов в 8-битной кодировке — для этих целей проще воспользоваться Web-браузером или FTP-утилитой. Преимущество утилиты TELNET заключается в предоставляемой возможности напрямую работать с командами соответствующего высокоуровневого протокола и выполнять нестандартные операции.

На рис. 3.28 показана сессия TELNET, работающая с WWW-сервером по протоколу HTTP.

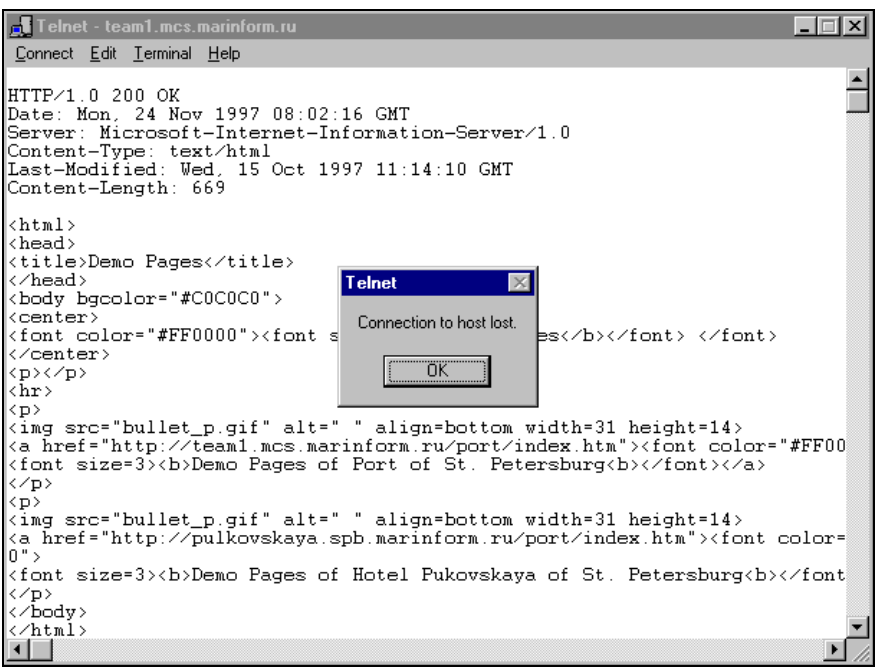
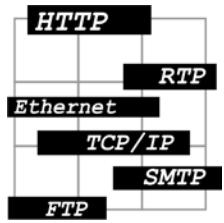


Рис. 3.28. Сессия TELNET, работающая с WWW-сервером по протоколу HTTP

Глава 4



Маршрутизация в сетях TCP/IP

Одной из основных задач протоколов сетевого уровня является определение сетевого маршрута. Под *сетевым маршрутом* (маршрутом) обычно понимается последовательность узлов и сетей, через которые должен пройти блок данных сетевого уровня (дейтаграмма) для того, чтобы достичь сети узла назначения.

Эта задача может выполняться специальным сетевым устройством — маршрутизатором или обычной вычислительной машиной. Обязательным условием в данном случае является наличие у этого устройства одного или нескольких интерфейсов для подключения к сегментам сети (сетевых интерфейсов). Каждый из сетевых интерфейсов маршрутизатора должен иметь как минимум один сетевой адрес. Возможно также использование одного сегмента канального уровня для организации нескольких сетей. Например, один интерфейс 10BaseT маршрутизатора может быть использован для подключения к двум или более сетям. В этом случае интерфейсу должно быть присвоено по одному адресу из каждой подключенной сети. Для того чтобы выполнять свою задачу, маршрутизатор должен иметь таблицу соответствия между адресами сетей и, как минимум, номерами своих интерфейсов. Такая таблица называется таблицей маршрутизации. Реальные таблицы маршрутизации могут иметь более сложную структуру и будут подробно описаны в последующих разделах данной главы.

Принцип "Hop-by-Hop"

Обычно процесс маршрутизации выполняется для каждой последующей дейтаграммы независимо от того, как он был выполнен для предыдущих дейтаграмм, сформированных тем же процессом пользователя. При получении очередной дейтаграммы маршрутизатор анализирует сетевой адрес узла назначения, находящийся в ее заголовке, и определяет, через какой из интерфейсов следует передать данную дейтаграмму для того, чтобы она смогла достичь сети узла назначения. В том случае, если искомая

сеть подключена к одному из интерфейсов маршрутизатора, выполняется непосредственная доставка дейтаграммы узлу назначения с использованием соответствующего протокола канального уровня. Если искомой сети нет в списке непосредственно подключенных сетей, маршрутизатор определяет, какой из соседних узлов-маршрутизаторов находится ближе к сети назначения, и передает дейтаграмму этому маршрутизатору через соответствующий сетевой интерфейс. В обоих случаях доставка дейтаграммы производится на канальном уровне. Отличие заключается в том, что в первом случае блок данных канального уровня, содержащий дейтаграмму, направляется непосредственно узлу назначения, а во втором — следующему маршрутизатору (Next Hop). Первый вариант называется прямой доставкой — (Direct Delivery), второй — косвенной доставкой (Indirect Delivery). Последующий маршрутизатор решает точно такую же задачу и либо доставляет дейтаграмму узлу назначения, либо передает ее одному из своих соседей. Таким образом, если все маршрутизаторы сети функционируют корректно, рано или поздно дейтаграмма, по принципу от "узла к узлу" (Hop-by-Hop), достигнет сети назначения и будет непосредственно доставлена последним маршрутизатором из цепочки. В процессе доставки содержимое самой дейтаграммы обычно не изменяется, за исключением содержимого поля **TTL** (Time To Live, Время жизни), которое обычно уменьшается на 1 при прохождении очередного маршрутизатора — для обеспечения уничтожения "заблудившихся" дейтаграмм. Исключение составляет случай, когда дейтаграмма должна быть фрагментирована из-за ограничения максимального размера поля полезной нагрузки блока данных канального уровня (MTU — Maximum Transfer Unit).

Для успешного выполнения процедуры поиска сетевого маршрута необходимо наличие скоординированных таблиц на всех маршрутизаторах, выполняющих косвенную доставку дейтаграммы. Неправильная запись в таблице маршрутизации любого маршрутизатора уведет дейтаграмму с истинного пути, что в лучшем случае приведет к увеличению времени ее доставки, а в худшем — к потере дейтаграммы. Существует два основных способа определения маршрута и построения таблиц маршрутизации — статический и динамический

Статическое и динамическое определение маршрута

При использовании статического способа таблицы маршрутизации строятся администратором сети вручную. Для их построения используются специ-

альные команды маршрутизатора (обычно это команда `route` , с помощью которой определяется маршрут для указанной сети). Параметрами этих команд служат адрес и маска сети назначения, адрес Next Hop для этой сети и имя или адрес интерфейса, через который должна быть передана дейтаграмма. Для корректной доставки дейтаграммы достаточно первых трех параметров. При использовании внеклассовых сетей в таблице маршрутизации вполне могут появиться конфликтующие маршруты, поэтому указание маски является обязательным. Маршрутизатор в первую очередь должен рассматривать маршруты с наиболее длинными масками. Например, для дальнейшей передачи дейтаграммы, адресованной в 10.9.8.7, такой маршрутизатор отдаст предпочтение сети 10.9.8.0/24, а не сетям 10.9.0.0/16 или 10.0.0.0/8, если таковые встретятся ему в таблице маршрутизации. Следовательно, наибольший приоритет имеют маршруты, с точностью до узла имеющие длину маски 32. В качестве примера рассмотрим сеть, представленную на рис. 4.1.

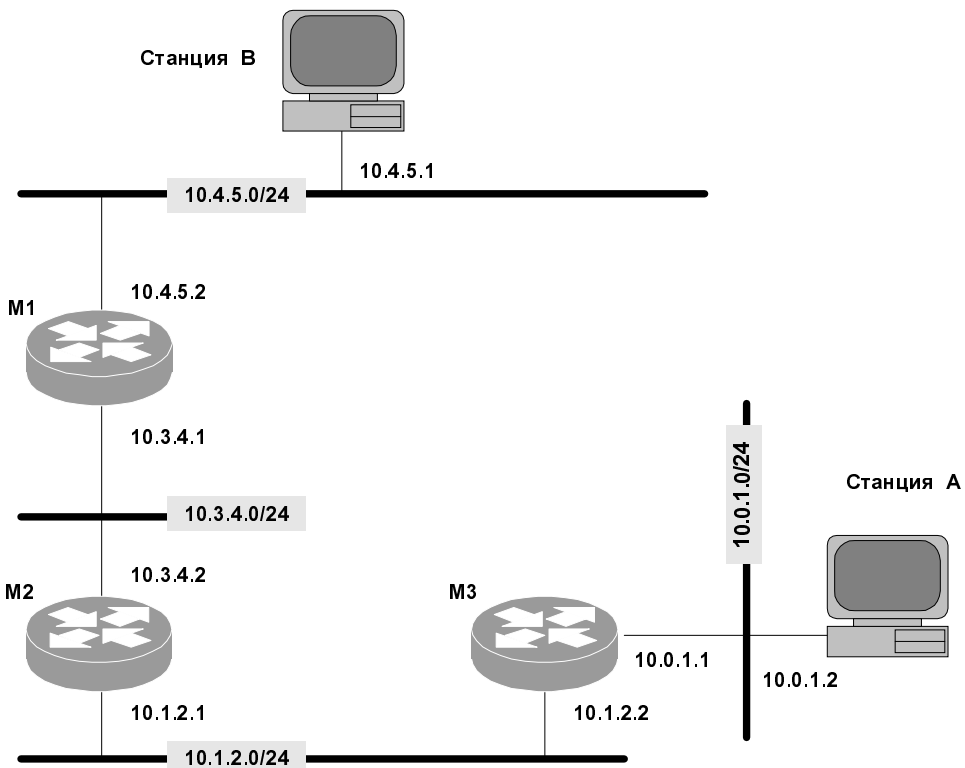


Рис. 4.1. Сеть со статической маршрутизацией

Пример таблицы статической маршрутизации для М3

Сеть	Маска	Next Hop	Интерфейс
10.0.1.0	255.255.255.0	10.0.1.1	10.0.1.1
10.1.2.0	255.255.255.0	10.1.2.2	10.1.2.2
10.3.4.0	255.255.255.0	10.1.2.1	10.1.2.2
10.4.5.0	255.255.255.0	10.1.2.1	10.1.2.2

Сети 10.0.1.0/24 и 10.1.2.0/24 в данном случае являются непосредственно подключенными, поэтому в качестве адреса Next Hop для них указаны адреса соответствующих сетевых интерфейсов маршрутизатора М3: 10.0.1.1 и 10.1.2.2. При передаче на канальном уровне дейтаграмм в сети 10.3.4.0/24 и 10.4.5.0/24 маршрутизатор М3 в качестве посредника будет использовать адрес 10.1.2.1 маршрутизатора М2.

Совершенно очевидно, что построение полной таблицы маршрутизации, в которой были бы указаны все сети, образующие Интернет, невозможно из-за огромного количества этих сетей. Для того чтобы упростить процедуру построения таблицы маршрутизации, в нее может быть включен специальный узел, куда необходимо передать дейтаграммы, адрес сети назначения которых не указан в таблице маршрутизации. Этот специальный узел называется Default Gateway (шлюз по умолчанию) и используется для маршрутизации соответственно в режиме "по умолчанию". Для обозначения маршрута к Default Gateway в качестве адресов сети и маски принято использовать нулевые значения. Таким образом, для изображенной на рис. 4.1 станции А, таблица маршрутизации в общем случае может состоять из одной записи:

```
ip route   0.0.0.0   0.0.0.0   10.0.1.1.
```

У внимательного читателя может возникнуть вопрос: зачем вообще этой станции нужна маршрутизация — ведь интерфейс-то у нее всего один? В данном случае, однако, важно не количество интерфейсов на узле, а количество маршрутизаторов, подключенных к сегменту его сети. Если таких маршрутизаторов больше одного, вполне может возникнуть ситуация, когда некоторые маршруты из данного узла будут построены через разные маршрутизаторы. Следовательно, в зависимости от того, куда адресована дейтаграмма, узел должен будет использовать разные адреса для передачи ее ближайшему маршрутизатору на канальном уровне.

Таким образом, при использовании статической маршрутизации администратор может и должен формировать маршруты на каждом узле (маршрутизаторе) подчиненной ему сети. Для сетей небольшого размера с постоянной структурой такой способ определения маршрута вполне может оказаться оптимальным. Однако по мере роста сети этот способ решения задачи маршрутизации будет становиться все менее целесообразным. Но основным недостатком данного метода является не размер и количество создаваемых

вручную таблиц маршрутизации, а тот факт, что эти таблицы являются фиксированными и, следовательно, не могут реально соответствовать текущей конфигурации сети. В качестве иллюстрации рассмотрим сеть, представленную на рис. 4.2.

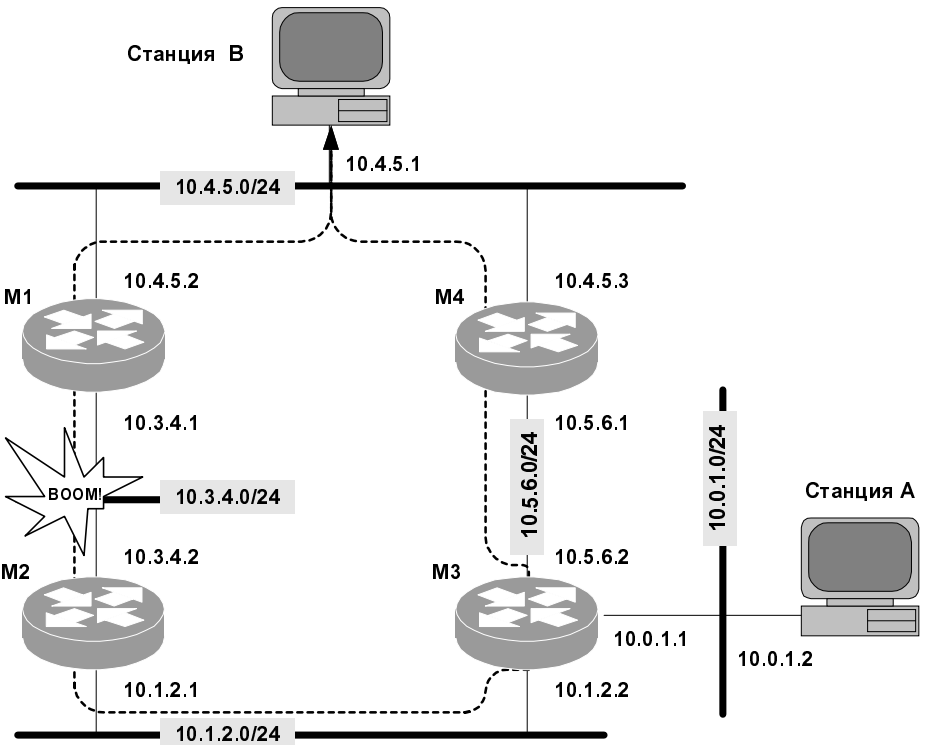


Рис. 4.2. Сеть с динамической маршрутизацией

Предположим, что в этой сети, так же, как в сети, представленной на рис. 4.1, маршрутизатор M3 для доставки дейтаграмм в сеть 10.4.5.0/24 использует маршрут, проходящий через M2. В таком случае, при возникновении неисправности на промежуточной сети 10.3.4.0/24 сеть 10.4.5.0/24 станет недоступной для маршрутизатора M3, поскольку альтернативный маршрут через M4 отсутствует в таблице маршрутизации. Современные маршрутизаторы (Cisco Systems, например) позволяют преодолеть этот недостаток статического определения маршрутов и сконфигурировать несколько различных статических маршрутов для одной сети назначения. При этом выполняется процедура выравнивания загрузки каналов (Load Balancing) — дейтаграммы будут передаваться по очереди в каждый из сконфигурированных маршрутов. Следует отметить, что если канал сети 10.5.6.0/24 имеет меньшую пропускную способность, чем каналы основного

маршрута, использование указанной процедуры может привести к снижению общей скорости передачи данных. Ниже приведен пример конфигурации для маршрутизатора М3.

Фрагмент конфигурационного файла маршрутизатора М3

```
ip route 10.4.5.0    255.255.255.0    10.1.2.1
ip route 10.4.5.0    255.255.255.0    10.5.6.1
...
```

Таким образом, существенным недостатком статического принципа определения маршрутов является отсутствие механизмов, которые могли бы обеспечить:

- получение информации о новых сетях;
- выбор наиболее эффективного маршрута в сети.

Второй основной принцип определения маршрута — динамическая маршрутизация — не имеет ни одного из вышеуказанных недостатков.

При использовании динамической маршрутизации формирование маршрутных таблиц производится маршрутизаторами автоматически в результате постоянного выполнения специального алгоритма маршрутизации. В процессе выполнения этого алгоритма маршрутизатор передает своим соседям информацию об известных ему маршрутах, получая от них взамен аналогичную информацию. После обработки полученной информации маршрутизатор строит заново или корректирует свою существующую таблицу маршрутизации. Поскольку информация о состоянии маршрутов поступает на маршрутизатор постоянно, использование такого алгоритма обеспечивает постоянное соответствие содержимого таблицы маршрутизации реальному состоянию сети. В зависимости от того, каким образом производится обмен маршрутной информацией между соседними маршрутизаторами, различают два типа алгоритмов маршрутизации:

- Distance-Vector (алгоритмы вектора расстояния);
- Link-State (алгоритмы состояния канала).

При использовании алгоритма типа Distance-Vector маршрутизатор через заранее определенные промежутки времени передает соседним маршрутизаторам содержимое своей таблицы маршрутизации. При использовании алгоритма типа Link-State маршрутизатор передает информацию только об изменении состояния системы. Далее будут подробно рассмотрены конкретные примеры этих типов алгоритмов маршрутизации.

Во время построения маршрутной таблицы могут быть сформированы несколько маршрутов, ведущих в одну сеть. Для того чтобы маршрутизатор мог выбрать один из них в качестве предпочтительного, он должен использовать обобщенную характеристику качества маршрута — метрику (Metric).

Каждый алгоритм маршрутизации использует свой алгоритм расчета этой характеристики. В наиболее простом случае в качестве метрики маршрута используется число узлов, отделяющих этот маршрутизатор от сети назначения. Более сложные метрики учитывают характеристики физических каналов, составляющих маршрут. Некоторые алгоритмы маршрутизации для увеличения скорости информационного обмена позволяют одновременно использовать несколько маршрутов, ведущих к одной сети.

Домены маршрутизации и автономные системы

Таким образом, динамическая маршрутизация представляет собой удобный и надежный способ построения маршрутных таблиц. При использовании динамической маршрутизации, в отличие от статической маршрутизации, таблицы могут быть построены автоматически, без какого-либо участия администратора. Полная самостоятельность маршрутизаторов, однако, может быть нецелесообразной и даже помешать реализации политики информационной безопасности. Ведь в данном случае распространение за пределы ответственной сети информации о ее внутренней структуре ничем не ограничивается. С другой стороны, злоумышленник может передавать в ответственную сеть сфальсифицированные маршрутные сообщения для того, чтобы исказить содержимое таблиц маршрутизации и нарушить тем самым правильное функционирование системы. Общеизвестно, что для управления сложными системами одинаково неприемлемым является как принцип централизации, так и принцип полной децентрализации. Истина, как обычно, находится между крайностями. Оптимальным можно будет считать протокол маршрутизации, способный обеспечить административное ограничение распространения маршрутной информации. Ограниченная область распространения маршрутной информации называется доменом маршрутизации (Routing Domain). Для формирования и распространения маршрутной информации маршрутизаторы одного домена используют единый алгоритм, называемый процессом маршрутизации. Для маркировки сообщений, содержащих маршрутную информацию, используется цифровой идентификатор (номер) процесса маршрутизации. Для построения таблицы маршрутизации процесс может использовать маршрутную информацию из сообщений, помеченных соответствующим номером. В качестве примера можно рассмотреть сеть, представленную на рис. 4.3.

Маршрутизаторы M1—M4 этой сети входят в домен 100, в то время как маршрутизаторы M1 и M5—M7 образуют домен 200. Таким образом, маршрутизатор M1 может получать информацию о сетях N2—N4 и N5—N7, в то время как маршрутизатор M4 располагает информацией о сетях только своего домена (N1—N4). Следовательно, пользователи сетей N2—N4 не смогут обращаться к ресурсам сетей N5—N7, и наоборот. Использование

доменов маршрутизации, помимо обеспечения информационной безопасности, позволяет в ряде случаев значительно повысить эффективность использования каналов передачи данных вследствие сокращения объема передаваемой маршрутной информации. Для дополнительного повышения уровня информационной безопасности некоторые протоколы маршрутизации используют также аппарат аутентификации источников маршрутной информации.

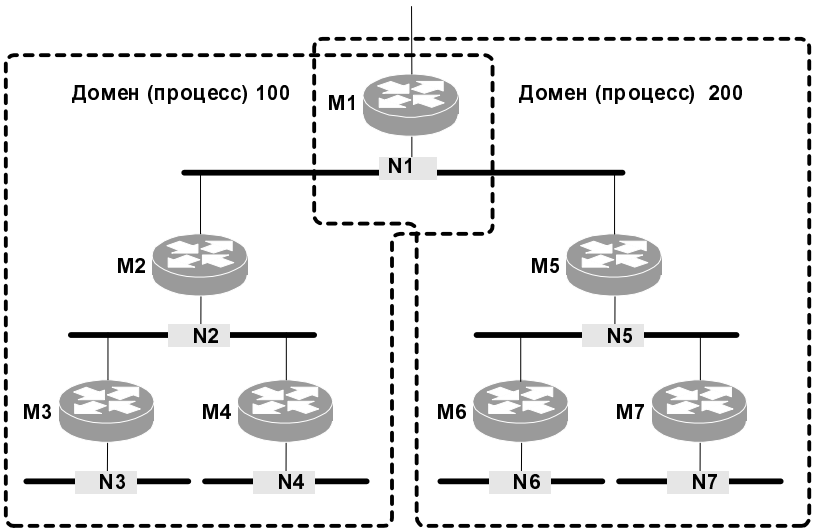


Рис. 4.3. Пример сети с двумя доменами маршрутизации.

Протоколы внешней и внутренней маршрутизации

Современные сети, как правило, имеют сложную и распределенную структуру. Для объединения фрагментов этих сетей могут использоваться различные каналы передачи данных, в том числе и арендуемые. Совокупность сетей, находящихся под единым административным управлением, принято называть *автономной системой*. Для определения внутренних маршрутов в автономных системах, как правило, используется один или несколько протоколов маршрутизации. В автономных системах этот класс протоколов принято называть протоколами внутренней маршрутизации (Interior Gateway Protocol). Специальные протоколы маршрутизации используются для определения маршрутов, выходящих за пределы автономной системы. Такие протоколы мы будем называть протоколами внешней маршрутизации (Exterior Gateway Protocol). Применение протоколов внешней маршрутизации позволяет администратору реализовать совокупность мер повышения

надежности и экономической эффективности информационного взаимодействия с внешними системами. При выборе маршрута, в отличие от протоколов внутренней маршрутизации, протоколы внешней маршрутизации анализируют не только физические характеристики каналов передачи данных маршрута. В число параметров, используемых современным протоколом внешней маршрутизации для определения качества маршрута, входят предпочтительность маршрута, последовательность проходимых автономных систем и другие параметры.

Протоколы RIP и IGRP

Протокол маршрутизации RIP (Routing Information Protocol) является наиболее популярным алгоритмом маршрутизации класса "Distance-Vector". Самое раннее описание этого класса алгоритмов было сделано Фордом (Ford) и Фалкерсоном (Fulkerson). По этой причине данный алгоритм иногда называют алгоритмом Форда-Фалкерсона. Термин "алгоритмы Беллмана-Форда" также может быть использован для обозначения алгоритмов данного типа, поскольку принципы построения алгоритмов данного класса были сформулированы в монографии Форда "Уравнения динамического программирования". Этот алгоритм является одним из первых алгоритмов маршрутизации, которые были использованы в информационно-вычислительных сетях вообще и в сети Интернет — в частности.

Формирование таблицы маршрутизации в алгоритмах "Distance-Vector"

Упрощенно функционирование маршрутизаторов класса "Distance-Vector" можно описать следующим образом.

- Маршрутизатор строит первичную таблицу маршрутизации, в которую помещает номера непосредственно подключенных сетей. Эта таблица содержит перечисленные ниже поля:
  - `Address` (`Адрес`). Сетевой (IP) адрес сети или узла назначения.
  - `Router` (`Маршрутизатор`). Сетевой адрес первого маршрутизатора на маршруте к сети или узлу назначения
  - `Interface` (`Интерфейс`). Сетевой адрес или номер интерфейса связи с первым маршрутизатором.
  - `Metric` (`Метрика`). Числовая характеристика маршрута от 0 до 15. Значение 0 соответствует непосредственно подключенной сети, метрика 15 указывает на недостижимость сети или узла назначения. В остальных случаях соответствует количеству промежуточных маршрутизаторов на маршруте к сети или узлу назначения.

- `Timer` (`Таймер`). Показатель актуальности информации о сети или узле назначения. Если информация не подтверждается источником в течение установленного временного интервала, запись о маршруте удаляется из таблицы.

- Маршрутизатор рассылает оформленную в виде специального сообщения об обновлении (Update) текущую версию таблицы маршрутизации соседним маршрутизаторам.

- При приеме аналогичного сообщения от соседнего маршрутизатора выполняются следующие действия.

  - Если сообщение содержит информацию о сети N, которой нет в таблице маршрутизации, адрес этой сети заносится в таблицу со следующими значениями полей: `Router` — адрес источника сообщения, `Interface` — адрес интерфейса, принявшего сообщение, в поле `Metric` заносится значение соответствующего поля исходного сообщения, увеличенное на весовой коэффициент интерфейса (обычно все весовые коэффициенты интерфейсов принимаются равными 1). Значение поля `Timer` у созданной записи устанавливается равным утроенному значению периода регулярных обновлений (90 с).

  - Если сообщение содержит информацию о сети N, которая есть в таблице маршрутизации, выполняется сравнение содержимого полей `Router` существующей записи и принятого сообщения. Если источник маршрутной информации в обоих случаях был один и тот же, поле `Metric` существующей записи модифицируется по обычному алгоритму значением соответствующего поля принятого сообщения. Поле `Timer` для модифицированной записи формируется так же, как и для вновь созданной.

  - Если информацию об известной сети N содержит сообщение, принятое от нового источника, маршрутизатор сравнивает содержимое полей `Metric` существующей записи и принятого сообщения. Если метрика существующего маршрута больше метрики нового маршрута, прежняя запись в таблице маршрутизации заменяется новой. В противном случае таблица маршрутизации никак не модифицируется.

- В том случае, если значение поля `Timer` у существующей записи стало равным 0, запись удаляется из таблицы маршрутизации.

- Процессы, описанные в двух последних пунктах, периодически повторяются, что позволяет динамически отслеживать изменения конфигурации сети.

Рассмотрим основные принципы формирования таблицы маршрутизации для сети, изображенной на рис. 4.2. Первоначальная таблица маршрутизатора М3 содержит только непосредственно подключенные сети.

Пример первоначальной таблицы маршрутизации M3

Address	Router	Interface	Metric	Timer
10.0.1.0	10.0.1.1	10.0.1.1	0	90
10.1.2.0	10.1.2.2	10.1.2.2	0	90
10.5.6.0	10.5.6.2	10.5.6.2	0	90

Изменение таблицы маршрутизации M3 после получения обновлений от M2 и M4

Address	Router	Interface	Metric	Timer
10.0.1.0	10.0.1.1	10.0.1.1	0	90
10.1.2.0	10.1.2.2	10.1.2.2	0	90
~~10.1.2.0~~	~~10.1.2.1~~	~~10.1.2.2~~	~~1~~	~~90~~
10.3.4.0	10.1.2.1	10.1.2.2	1	90
10.5.6.0	10.5.6.2	10.5.6.2	0	90
~~10.5.6.0~~	~~10.5.6.1~~	~~10.1.2.2~~	~~1~~	~~90~~
10.4.5.0	10.5.6.1	10.5.6.2	1	90

Зачеркнутые строки удаляются из таблицы маршрутизации, поскольку соответствуют более длинным маршрутам (значение поля `Metric` равно 1) в известные сети. После получения второй порции обновлений таблица маршрутизации не претерпевает изменений.

Пример таблицы маршрутизации M3 после получения второй порции обновлений

Address	Router	Interface	Metric	Timer
10.0.1.0	10.0.1.1	10.0.1.1	0	90
10.1.2.0	10.1.2.2	10.1.2.2	0	90
10.3.4.0	10.1.2.1	10.1.2.2	1	90
10.5.6.0	10.5.6.2	10.5.6.2	0	90
10.4.5.0	10.5.6.1	10.5.6.2	1	90
~~10.4.5.0~~	~~10.1.2.1~~	~~10.1.2.2~~	~~2~~	~~90~~

Поскольку сеть 10.4.5.0, представленная маршрутизатором M2 по информации от M1, имеет метрику 2, она также исключается из таблицы маршрутизации. Таким образом, уже после второй порции обновлений таблица маршрутизации принимает окончательный вид, что объясняется относительной простотой структуры рассмотренной сети.

Возникновение циклических маршрутов

Описанный выше алгоритм формирования маршрутной таблицы является предельно упрощенным и по многим причинам не может быть использован

в качестве алгоритма маршрутизации в реальных системах. Одной из таких причин является потенциальная нестабильность формируемых маршрутов, обусловленная использованным в данном случае способом рассылки маршрутной информации. Маршрутизатор М3 может получить от М2 или М4 обратно информацию о сети 10.0.1.0 (с соответствующей метрикой). Рассмотрим этот процесс на примере более простой сети, изображенной на рис. 4.4.

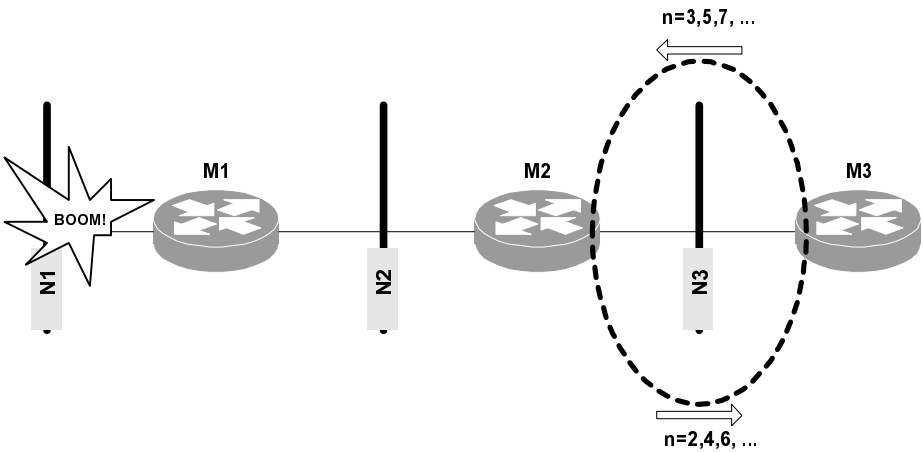


Рис. 4.4. Возникновение циклического маршрута

При нормальном функционировании всех компонентов маршрутизатор М2 имеет маршрут в сеть N1 через маршрутизатор М1 с метрикой 1, а маршрутизатор М3 имеет маршрут в сеть N1 через маршрутизатор М2 с метрикой 2. Мы будем обозначать его следующим образом — [N1, М2, 2]. Предположим теперь, что маршрутизатор М1 потерял связь с сетью N1 и перестал включать эту сеть в свои периодические обновления, рассылаемые соседним маршрутизаторам (в данном случае — М2). Однако только по истечении установленного интервала времени тайм-аута для маршрута маршрутизатор М2 должен будет пометить маршрут [N1, М1, 1] как нерабочий и исключить его из своих обновлений. Маршрутизатор М3, таким образом, узнает о проблеме с N1 со значительным опозданием. Следовательно, маршрутизатор М3 в течение шести периодов будет рассылать обновления несуществующего уже маршрута [N1, М2, 2]. Если маршрутизатор М2 получит это обновление после того момента, когда маршрут [N1, М1, 1] был объявлен нерабочим, он скорректирует свою таблицу маршрутизации, введя в нее неверный маршрут [N1, М3, 3]. Поскольку этот маршрут будет включен и в периодические обновления, рассылаемые соседям маршрутизатора М2, маршрутизатор М3 вскоре также должен будет скорректировать свою таблицу маршрутизации и увеличить на 2 метрику своего несуществующего маршрута в сети N1. Таким образом, возникает циклический маршрут в несуществующую сеть. Помимо того что весь полезный трафик, адресованный

в сеть N1 и проходящий через маршрутизаторы M2 и M3, будет потерян, следствием возникновения циклического маршрута может явиться также перегрузка этим трафиком каналов передачи данных.

Описанный выше процесс может продолжаться бесконечно долго и может быть прерван либо в том случае, когда первоначально потерянная связь с сетью N1 будет восстановлена, либо при достижении предельной длины маршрута. В протоколе маршрутизации RIP в качестве предельного значения метрики маршрута используется значение 15. Сети, удаленные от данного узла на расстояние, которое превышает 15 переходов, считаются недостижимыми — unreachable. Таким образом, циклический маршрут в сети, представленной на рис. 4.4, будет автоматически разорван, когда его длина превысит это значение. Однако, с одной стороны, установление подобного ограничения негативно сказывается на эксплуатационных характеристиках протокола маршрутизации, а с другой стороны, указанное ограничение не позволяет уничтожать циклические маршруты в достаточно короткие сроки. Циклические маршруты в основном появляются вследствие того, что информация о возникшем маршруте передается маршрутизатором в составе очередного периодического обновления, в то время, как исключение пропавшего маршрута может быть сделано не ранее чем через три периода обновлений маршрутов. Очень точно эту особенность алгоритмов Distance-Vector подметил Duglas Comer в своей книге "Internetworking with TCP/IP": "Good news travels quickly; bad news travels slowly" (Добрые новости летят на крыльях; плохие ползут еле-еле). Для противодействия возникновению циклических маршрутов алгоритмы маршрутизации Distance-Vector вообще и алгоритм протокола RIP в частности используют целый комплекс дополнительных мер и процедур для выравнивания скоростей распространения по сети информации о новых и пропавших маршрутах. Для противодействия возникновению циклических маршрутов чаще всего применяются методы, позволяющие увеличить скорость распространения информации о пропавших маршрутах. Такой принцип используют процедуры Split Horizon (расщепленный горизонт), Poison Reverse (отравленный обратный путь) и метод управляемых обновлений, подробное описание которых приведено в последующих разделах настоящей главы.

Процедуры Split Horizon и Poison Reverse

Процедура Split Horizon (расщепленный горизонт) наиболее часто используется в алгоритмах маршрутизации Distance-Vector для противодействия возникновению циклических маршрутов. Основная идея, лежащая в основе данной процедуры, достаточно проста и интуитивно понятна. Действительно, если проблема возникает из-за того, что информация о некотором маршруте передается обратно его источнику, то не проще ли запретить такую передачу, тем более, что пользы от этой информации источнику все равно никакой нет и быть не может. Таким образом, правило Split Horizon гласит:

информация о маршруте в некоторую сеть N, полученная от маршрутизатора M, не может быть включена в регулярные обновления, отправляемые этому маршрутизатору. Использование такой процедуры позволяет гарантированно избежать появления циклических маршрутов между двумя маршрутизаторами. Кроме того, процедура Split Horizon повышает эффективность использования пропускной способности канала передачи данных за счет сокращения неинформативной составляющей сообщения об обновлении маршрутов. Однако, в том случае, если циклический маршрут образован несколькими маршрутизаторами, применение этой процедуры не даст желаемого эффекта. В качестве иллюстрации последнего утверждения рассмотрим пример, приведенный на рис. 4.5.

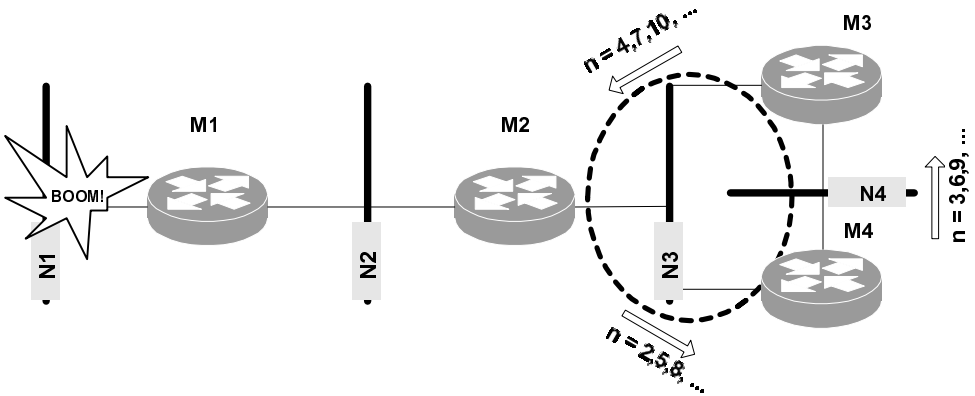


Рис. 4.5. Возникновение циклического маршрута из трех маршрутизаторов

Предположим, что у маршрутизатора M3 в представленной сети маршрут [N1, M2, 2] был удален по тайм-ауту раньше, чем аналогичный маршрут у маршрутизатора M4. Следовательно, получение маршрутизатором M3 очередного обновления от маршрутизатора M4 приведет к появлению несуществующего маршрута [N1, M4, 3]. Отметим, что применение процедуры Split Horizon не помешает маршрутизатору M4 передавать обновления для сети N1 маршрутизатору M3, поскольку первичная информация об этой сети была получена от маршрутизатора M2. И в дальнейшем, при передаче обновления от маршрутизатора M3 к M2, использование этого правила также не поможет, поскольку маршрутизатор M3 о сети N1 узнал от маршрутизатора M4, а не от M2. В результате маршрутизатор M2 тоже получит обновление для несуществующего маршрута и порочный круг опять замкнется.

Аналогичный принцип противодействия возникновению циклических маршрутов используется также и процедурой Poison Reverse (отравленный обратный путь). Отличие от ранее описанной процедуры в данном случае заключается только в том, что информация о маршруте в некоторую сеть N,

полученная от маршрутизатора M, включается в регулярные обновления, отправляемые этому маршрутизатору с метрикой 16. Понятно, что в результате использования этой процедуры потенциально опасные маршруты будут удалены из таблицы маршрутизации. Но если при использовании чистой процедуры Split Horizon эти маршруты будут удалены по истечению таймера, то использование Poison Reverse приведет к их мгновенному уничтожению. Кроме того, в некоторых случаях процедура Poison Reverse может быть использована для противодействия возникновению длинных кольцевых маршрутов. Если метрика какого-либо маршрута начинает увеличиваться, это может быть следствием того, что он является фрагментом кольцевого маршрута. Безусловно, что применение процедуры Poison Reverse в этих условиях может привести к ошибочному уничтожению прямых, но нестабильных маршрутов. Следует, однако, иметь в виду, что, во-первых, длинный циклический маршрут можно обнаружить и обезвредить только описанным выше способом, а во-вторых, ошибочно удаленные маршруты будут обязательно восстановлены при получении очередных периодических обновлений. Основной недостаток процедуры Poison Reverse заключается в том, что при ее использовании значительно увеличивается объем периодических обновлений, отправляемых маршрутизатором. По этой причине руководящий документ IETF (Internet Engineering Task Force) STD 56 рекомендует при реализации алгоритма RIP-маршрутизации обеспечивать возможность выбора пользователем типа процедуры противодействия возникновению циклических маршрутов.

Управляемое обновление маршрутов (Triggered Update)

Наиболее мощным средством борьбы с длинными циклическими маршрутами является использование апериодических управляемых обновлений маршрутов. Важной особенностью этих обновлений является то, что маршрутизатор формирует обновление при каждом изменении своей таблицы маршрутизации, не дожидаясь наступления момента передачи очередного периодического обновления. Если при получении такого управляемого обновления последующий маршрутизатор скорректирует свою таблицу маршрутизации, он, в свою очередь, сформирует управляемое обновление, которое направит своим соседям. Таким образом, информация об изменении конфигурации распространяется по сети действительно мгновенно. Более того, вследствие особого дифференциального принципа формирования таких обновлений, они распространяются по сети от источника только в нужных направлениях, поскольку маршрутизатор, не изменивший свою таблицу маршрутизации при получении управляемого обновления, не сформирует вторичное обновление и блокирует его дальнейшее распространение. Следует отметить, что при формировании управляемых обновлений обязательно

должен быть использован принцип Split Horizon — обновления не должны передаваться обратно вверх по маршруту.

К сожалению, несмотря на вышеописанные достоинства данного метода, применение управляемых обновлений не может обеспечить полного исключения возможности возникновения циклических маршрутов. В достаточно большой сети вполне возможна ситуация, когда маршрутизатор после получения управляемого обновления примет запоздавшее регулярное обновление. Очевидно, что в таком случае применение управляемого обновления не сможет принести желаемого эффекта.

Кроме того, практическая реализация управляемых обновлений требует особой осторожности, поскольку каскад мгновенных обновлений маршрутных таблиц нескольких маршрутизаторов может полностью блокировать функционирование сети на некоторое время. Для предотвращения перегрузки сети управляемые обновления передаются порциями с интервалом, величина которого случайным образом выбирается из диапазона (1...5) секунд. После передачи очередного управляемого обновления маршрутизатор определяет момент времени для формирования последующего обновления и так далее. Очередное управляемое обновление не формируется маршрутизатором при выполнении одного из следующих условий:

Со времени передачи последнего обновления таблица маршрутизации оставалась неизменной (с учетом выполнения процедуры Split Horizon).

Пришло время передавать очередное регулярное обновление таблицы маршрутизации.

Управляемые обновления обычно имеют существенно меньший размер, чем периодические обновления, поскольку содержат только информацию об изменившихся маршрутах. Проект стандарта IETF STD 56 настоятельно не рекомендует использование аппарата управляемых обновлений для передачи полной таблицы маршрутизации.

Особенности алгоритма RIP II

Алгоритм маршрутизации RIP давно и успешно применяется во многих сетях. В настоящее время наиболее активно используется вторая редакция этого алгоритма, в которой разработчики постарались устранить слабости предыдущей редакции и обеспечить выполнение некоторых новых дополнительных функций.

Таймеры

Выше было отмечено, что для динамического формирования таблиц маршрутизации в алгоритме RIP используются периодические обновления — update. В том случае, если какая либо из сетей (например — N) перестала быть доступной через данный маршрутизатор, он должен исключить

указание об этой сети из своих периодических обновлений. Таким образом, остальные маршрутизаторы могут получить информацию об изменении конфигурации сети только после получения от данного маршрутизатора периодического или управляемого обновления. В зависимости от реализации, в этом обновлении сеть N либо не будет представлена совсем, либо будет представлена с "бесконечной" метрикой. Маршрутизаторы, которые примут данное обновление, должны будут соответствующим образом скорректировать свои таблицы маршрутизации. Однако, вполне возможно, что очередное периодическое обновление было просто потеряно в сети из-за возникновения краткосрочной перегрузки или временной неработоспособности канала передачи данных. Для того чтобы в этой ситуации маршруты не были ошибочно удалены из таблицы маршрутизации, каждому маршруту ставится в соответствие специальный счетчик времени, который называется timeout-timer. В тот момент времени, когда данный маршрут включается в таблицу маршрутизации или когда для него приходит очередное обновление, значение счетчика timeout-timer устанавливается равным $T_{\text{timeout max.}} = 90$ секунд и этот счетчик начинает обратный отсчет времени. В том случае, если счетчик timeout-timer какого либо маршрута достигнет значения 0, этот маршрут должен быть исключен из числа активных маршрутов и отправлен на мусорную свалку (garbage-collection).

Окончательное исключение неактивного маршрута из таблицы маршрутизации производится после истечения временного интервала, который определяется счетчиком времени garbage-collection timer. В тот период времени, когда исключенный из активного обращения маршрут находится на свалке (120 секунд), маршрутизатор рассылает своим соседям уведомления об уничтожении данного маршрута: включает его в регулярные обновления с метрикой 16.

Сообщения протокола RIP

Как было отмечено выше, для взаимного обмена маршрутной информацией со своими соседями-коллегами маршрутизаторы протокола RIP используют сообщения специального формата. Для отправки этих сообщений маршрутизаторы первой версии протокола RIP обычно использовали широковещательный адрес (Broadcast) сетевого уровня. Особенно негативно эта особенность протокола проявлялась в сетях множественного доступа (например — Ethernet), где она могла приводить к значительному снижению эффективности использования сетевых ресурсов. Маршрутизаторы, поддерживающие вторую версию протокола RIP, используют специально выделенный групповой (multicast) адрес 224.0.0.9 для формирования периодических обновлений. В некоторых случаях маршрутизатор RIP II может также использовать передачу маршрутной информации конкретному соседу в обычном одноадресном (unicast) режиме.

Для передачи маршрутной информации маршрутизатор RIP использует дейтаграммы, которые адресуются в порт 520 протокола UDP. Сообщение протокола RIP состоит из заголовка сообщения и следующих за ним маршрутных записей Route Entries (RTE). Обычно в сообщении протокола RIP содержится не более 25 маршрутных записей. Следовательно, для передачи большой таблицы маршрутизатор должен использовать несколько последовательных сообщений. Формат сообщения протокола маршрутизации RIP приведен на рис. 4.6.

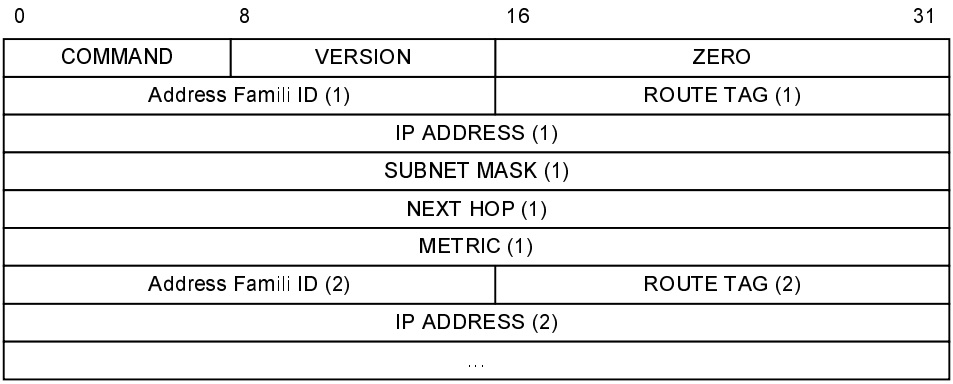


Рис. 4.6. Формат сообщения протокола RIP II

В заголовке сообщения RIP размещается код сообщения: значение 1 соответствует сообщению типа request (запрос), значение 2 — response (ответ).

Сообщения типа request маршрутизатор использует для обращения к соседям при начальном формировании или обновлении таблицы маршрутизации. Необходимость формирования такого сообщения может возникнуть, например, при первоначальном включении маршрутизатора. Обычно сообщения данного типа передаются с использованием адреса сетевого уровня типа Broadcast (при использовании первой версии протокола) или Multicast (для второй версии).

Сообщения типа request могут быть переданы и в том случае, если маршрутизатор хочет получить информацию о маршруте до какой-либо конкретной сети N либо группы сетей. В этом случае маршрутизатор должен указать в тексте запроса адрес этой сети или группы сетей. Если маршрутизатор запрашивает содержимое всей таблицы маршрутизации, сообщение типа request формируется следующим образом — в поле **Address Family ID** помещается значение 0, а в поле **METRIC** помещается значение 16. Наличие такого сочетания полей у запроса типа request является признаком обращения за полной таблицей маршрутизации.

Маршрутизатор RIP формирует сообщения типа response в ответ на запрос, полученный от одного из соседних маршрутизаторов, или самопроизвольно

по истечении установленного интервала времени (периодические обновления). Сообщения response могут быть также сформированы при изменении таблицы маршрутизации (Triggered Update). Если сообщение response формируется в ответ на сообщение типа request, отвечающий маршрутизатор обязан произвести тщательную проверку сетевого адреса источника данного сообщения. Ответ должен быть сформирован только для сообщения, отправленного маршрутизатором, который находится в одной из непосредственно подключенных сетей.

В поле **VERSION** размещается номер версии протокола маршрутизации RIP (1 или 2).

В поле **Address Family ID** сообщения RIP размещается признак типа протокола сетевого уровня, определяющий схему формирования адресов. Следует отметить, что протокол маршрутизации RIP используется для определения маршрутов в сетях, имеющих различные схемы построения сетевого адреса (IP, IPX). Протоколу IP, в частности, соответствует значение 2 поля **Address Family ID**.

Поле **ROUTE TAG** занимает два байта в заголовке сообщения протокола RIP. В том случае, когда протокол маршрутизации RIP используется для передачи информации о маршрутах между автономными системами, то есть выполняет функции протокола маршрутизации BGP (Border Gateway Protocol) или EGP (Exterior Gateway Protocol), в поле **ROUTE TAG** может быть помещен номер автономной системы.

Поле **IP ADDRESS** занимает 4 байта в заголовке сообщения протокола RIP. В сообщении типа RIP request в данном поле содержится адрес сети, для которой необходимо определить маршрут. В сообщениях типа RIP response в данное поле помещается адрес сети, которая является достижимой для источника данного сообщения.

Поле **SUBNET MASK** занимает 4 байта в заголовке сообщения протокола RIP. В этом поле сообщений протокола RIP II размещается маска сети, адрес которой расположен в соответствующем поле **IP ADDRESS**. У сообщений, сформированных протоколом RIP I, содержимое данного поля должно быть равно 0.

В поле **NEXT HOP** сообщений протокола RIP II размещается адрес непосредственно подключенного маршрутизатора для использования в качестве next hop для передачи данных в сеть назначения. Использование содержимого поля **NEXT HOP** имеет практический смысл в том случае, если несколько маршрутизаторов RIP II будут расположены в одной локальной сети.

Метрика маршрута размещается в поле **METRIC,** занимающем 4 байта в заголовке сообщения. В соответствии со спецификацией протокола RIP метрика маршрута определяется целым положительным числом, не превышающим число 16.

Маршрутизация для внеклассовых сетей

Наиболее существенным недостатком протокола RIP являлось отсутствие поддержки популярной спецификации CIDR (Class Less Inter Domain Routing), появившейся существенно позже первой версии этого протокола. Использование протокола RIP I в сетях с удлиненными масками могло привести к существенным ошибкам маршрутизации. В частности маршрутизатор RIP I рассматривает внеклассовые сети типа 10.1.0.0/16, 10.2.0.0/16 и так далее, как одну сеть класса A 10.0.0.0/8, и формирует для нее один маршрут, что, естественно, приводит к потере пакетов, направляемых в подсети. Этот недостаток существенно сдерживал применение RIP и был устранен во второй версии протокола путем введения в маршрутную информацию дополнительной характеристики SUBNET MASK (маска сети назначения).

Установление подлинности источника маршрутной информации

В ряде случаев для обеспечения информационной безопасности целесообразно установление подлинности (аутентификация) источника маршрутной информации. Не секрет, что получение фальсифицированного маршрута может существенно нарушить информационный обмен в системе и даже привести к утечке информации. Во второй версии протокола RIP предусмотрена возможность использования процедуры аутентификации источника маршрутной информации на уровне сообщений. Для передачи параметров процедуры установления подлинности используется первая маршрутная запись. Признаком того, что сообщение содержит пароль аутентификации источника, является наличие кода 0xFFFF в поле **Address Family ID** этой записи. На рис. 4.7 приведена структура первой маршрутной записи.

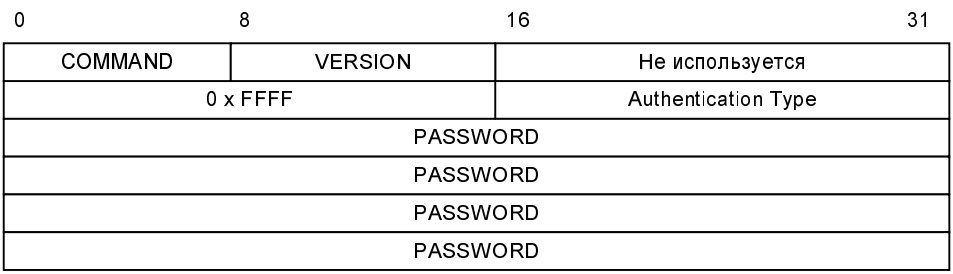


Рис. 4.7. Аутентификация источника маршрутной информации RIP II.

В оставшихся 16 байтах маршрутной записи размещается пароль источника сообщения. В настоящее время к использованию предлагается простая схема аутентификации, когда пароль является статическим, поэтому использование описанной процедуры не может обеспечить гарантированной защиты маршрутизатора от фальсифицированной маршрутной информации.

В заключение следует отметить, что область возможного применения протокола RIP является достаточно ограниченной. Протокол RIP не обеспечивает решение всех возможных проблем, возникающих в процессе определения маршрута в сетях передачи данных. Как уже упоминалось выше, в первую очередь он предназначен для использования в качестве протокола IGP (Interior Gateway Protocol) в гомогенных сетях небольшого размера. Кроме того, использование этого протокола приводит к появлению специфических ограничений на параметры администрируемой сети. Использование протокола RIP целесообразно в сетях, самый длинный путь в которых составляет не более 15 переходов (hops). Как было показано выше, это ограничение определяется принятым в данном алгоритме способом вычисления метрики маршрута. Существенным недостатком данного метода является то, что при его использовании преимущество получают более короткие маршруты. В реальных системах зачастую оправдывается поговорка "Кривой дорогой — ближе". В качестве примера еще раз обратимся к сети, представленной на рис. 4.2. Совершенно очевидно, что для построения маршрута в сеть 10.4.5.0/24 маршрутизатор М3 выберет маршрут, проходящий через маршрутизатор М4. Этот выбор может, однако, оказаться далеко не оптимальным в том случае, если маршрутизаторы М3 и М4 связаны друг с другом низкоскоростным каналом. Теоретически администратор может помочь маршрутизатору сделать правильный выбор, если установит большие (в данном случае 3) значения метрик для "тормозных" каналов. Следует, однако, иметь в виду, что при использовании метрик больше 1 фундаментальное ограничение протокола RIP (15 переходов) может вызвать проблемы даже для не очень большой сети. Кроме того, в итоговой редакции второй версии протокола RIP не нашлось места для доменов маршрутизации, что также существенно ограничивает возможности использования этого протокола.

Протокол IGRP

От многих врожденных недостатков, присущих протоколу RIP, свободен протокол маршрутизации IGRP (Interior Gateway Routing Protocol), реализованный в маршрутизаторах Cisco Systems. Ниже будет приведено описание основополагающих принципов построения этого протокола.

Протокол IGRP, так же как и RIP, является протоколом маршрутизации типа Distance-Vector со всеми вытекающими последствиями. Основное отличие от RIP заключается в том, что протокол IGRP для сравнения маршрутов использует так называемую композитную метрику (composite metric), при расчете которой используются характеристики, определяющие реальную полезность маршрута:

- **B** — Bandwidth — минимальное значение пропускной способности среди каналов маршрута;

- **D** — Delay — суммарная задержка распространения сигнала от источника до пункта назначения маршрута;
- **R** — Reliability — показатель надежности маршрута. При вычислении этого значения используется соотношение числа принятых и переданных технологических сообщений;
- **L** — Loading — соотношение загрузка/пропускная способность на наиболее загруженном участке маршрута;
- **MTU** — минимальное значение maximum transfer unit у каналов, образующих маршрут.

Композитная метрика M маршрута может быть рассчитана по следующей формуле:

$$M = \{[(K_1/B\times(1-L)]+(K_2\times D)\}\times R.$$

Значения пары весовых коэффициентов K_1 и K_2 устанавливаются в зависимости от типа трафика (содержимое поля TOS (Type of Service) дейтаграммы) и выбираются таким образом, чтобы наименьшие значения метрики получали маршруты с наиболее благоприятными характеристиками именно для выбранного типа трафика. Например, для передачи оцифрованного голосового сигнала должен быть выбран маршрут с минимальной задержкой, в то время как для передачи трафика ЛВС предпочтение должно быть отдано маршруту с максимальной пропускной способностью.

Еще одно важное отличие протокола маршрутизации IGRP заключено в возможности определения и использования нескольких параллельных путей для передачи трафика по принципу выравнивания загрузки каналов (Load Balancing). Это свойство протокола маршрутизации позволяет существенно увеличить скорость информационного обмена при объединении нескольких параллельных каналов и достаточно широко используется в современных сетях передачи данных. Этот же механизм может быть использован для обеспечения динамического резервирования каналов, когда резервные каналы используются для передачи данных наравне с основными каналами, а не только при аварийном отключении последних. Следует отметить, что задача одновременного включения нескольких каналов, имеющих различную пропускную способность, не является тривиальной. Для эффективного использования этих каналов необходимо обеспечивать загрузку полезным трафиком пропорционально их пропускной способности. Коэффициент пропорциональности метрик маршрутов называют допустимым расхождением (variance). В том случае, если допустимое расхождение метрик для протокола установлено равным 1, в качестве альтернативных маршрутов будут использоваться только те, метрики которых полностью совпадают с минимальной. Если коэффициент допустимого расхождения будет равен двум, маршруты, метрика которых не более чем в два раза превышает метрику минимального маршрута, также могут использоваться для передачи данных в этом направлении.

Существенным преимуществом протокола IGRP по отношению к RIP является возможность определения доменов маршрутизации. Как было отмечено выше, использование нескольких процессов маршрутизации в пределах одной сети позволяет обеспечить большую эффективность управления информационными потоками внутри нее. Достоинством протокола IGRP можно считать также существенное (90 секунд) увеличение периода циклических обновлений маршрутов.

Как и всякий протокол маршрутизации Distance-Vector, протокол IGRP использует множество процедур для предотвращения возникновения циклических маршрутов. К ним относятся описанные ранее процедуры Triggered Updates, Split Horizon и Poison Reverse. В качестве дополнительного средства борьбы с циклическими маршрутами в протоколе маршрутизации IGRP могут быть использованы таймеры удержания. Этот таймер включается при переводе маршрута в пассивное состояние и определяет минимальную величину временного интервала, по истечении которого удаленный маршрут может быть восстановлен при получении соответствующих обновлений. В силу того, что использование удерживающих таймеров однозначно приводит к увеличению времени установления маршрута, данный механизм должен использоваться с особой осторожностью.

Недостатками протокола IGRP можно считать отсутствие поддержки внеклассовых сетей и использование широковещательного адреса для передачи периодических обновлений. Эти недостатки, однако, устранены в последующей версии протокола, о которой будет подробно рассказано в разделе Enhanced IGRP (EIGRP).

Примеры практического применения протоколов RIP II и IGRP

"Тихий" RIP

В ряде случаев бывает полезно, чтобы маршрутизатор получал информацию о маршрутах, но сам не принимал участие в их формировании путем передачи регулярных массивов update. Разновидность такого алгоритма маршрутизации называют silent RIP — "Тихий RIP". При использовании этой разновидности алгоритма маршрутизатор может или совсем не формировать массивы update, или не направлять эти массивы на какой-либо из своих интерфейсов. Такое поведение маршрутизатора может оказаться целесообразным, в частности, для предотвращения появления циклических маршрутов, а также по соображениям обеспечения безопасности. Рассмотрим пример, изображенный на рис 4.8. Поскольку соединение с сетью 10.1.0.0/16 обеспечивается только через маршрутизатор M1, использование регулярных периодических обновлений для этого маршрута может в данном случае оказаться нецелесообразным.

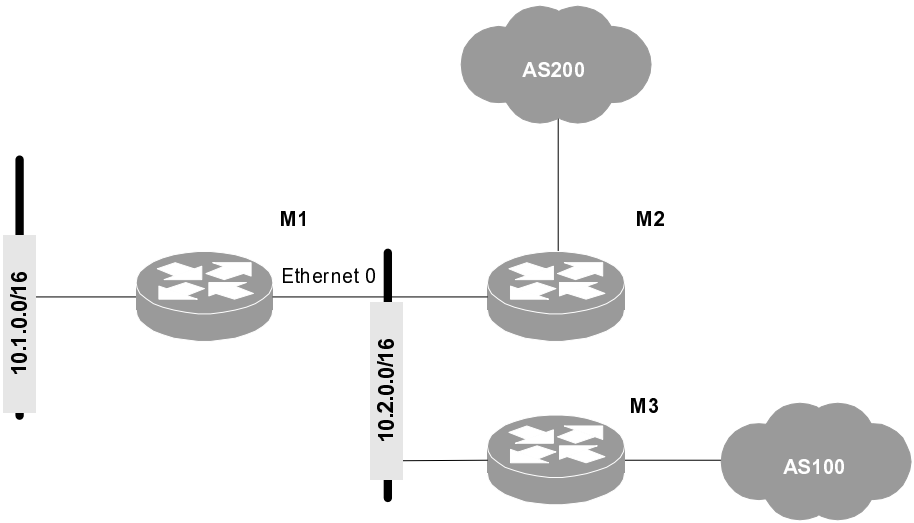


Рис. 4.8. Пример использования режима маршрутизации "Тихий RIP"

Ниже приведен фрагмент конфигурационного файла для маршрутизаторов M1 и M2. Включение пассивного режима маршрутизатора RIP на M1 обеспечивает получение необходимой маршрутной информации от M2 и M3 и в то же время позволяет более экономно использовать пропускную способность каналов, связывающих M2 и M3 с автономными системами (Autonomous System) AS100 и AS200.

Фрагмент конфигурационного файла для маршрутизатора M1

```
...
router rip
network 10.0.0.0
passive-interface Ethernet 0
```

Фрагмент конфигурационного файла для маршрутизатора M2

```
...
router rip
redistribute static subnets
ip route 10.1.0.0 255.255.0.0 10.2.0.2
```

Протокол маршрутизации OSPF

Характерной чертой протоколов маршрутизации класса "Distance-Vector" является полное равноправие маршрутизаторов, приводящее к неоптимальному

использованию пропускной способности каналов передачи данных при обмене маршрутной информацией. Действительно, если каждый маршрутизатор этого класса периодически передает свою таблицу маршрутизации соседям, содержащаяся в ней информация неминуемо будет многократно тиражироваться. Кроме того, далеко не каждый маршрутизатор, в силу особенности своего расположения в сети, способен быть источником такой информации. Большинство маршрутизаторов в данном случае просто является посредником при передаче маршрутной информации, а поскольку сам процесс передачи никак не скоординирован, информация опять-таки тиражируется.

По принципиально иной схеме построены алгоритмы маршрутизации класса "Link-State" (Состояние Канала). Протоколы класса "Link-State" свободны от большинства недостатков, присущих протоколам "Distance-Vector" и обладают рядом существенных дополнительных достоинств. К протоколам этого класса относится протокол OSPF (Open Shortest Path First).

В отличие от универсальных протоколов (например, RIP или EIGRP) протокол OSPF предназначен для построения маршрутов только в сетях TCP/IP.

Основными достоинствами протокола OSPF являются:

- отсутствие ограничений на размер сети;
- поддержка внеклассовых сетей;
- передача сообщений протокола с использованием multicast-адресов;
- высокая скорость установления маршрутов при изменении состояния системы;
- встроенная процедура установления подлинности источника маршрутной информации.

При использовании протокола маршрутизации OSPF допускается существование нескольких маршрутов в направлении некоторого узла сети. В том случае, если эти маршруты обеспечивают одинаковое качество передачи данных, информационный поток в адрес данного узла может быть направлен по всем этим каналам одновременно, что обеспечит существенное увеличение скорости передачи данных.

Использование Load balancing позволяет реализовать гибкое и эффективное динамическое резервирование каналов передачи данных.

Применение процедуры установления подлинности целесообразно в тех информационных системах, где большое внимание уделяется информационной безопасности. В таких системах маршрутизаторы, участвующие в процессе определения маршрута, должны выполнить совокупность действий для установления приемником подлинности источника передаваемых данных (authentication procedure). Только в том случае, если источник передаваемых

данных успешно выполнил процедуру аутентификации, данные о полученных от него маршрутах, принимаются для обработки. Целесообразность использования процедуры аутентификации может быть проиллюстрирована на следующем примере. Предположим, что злоумышленник, который расположен в локальной сети, передает фальсифицированные пакеты протокола маршрутизации, в которых определяет свой компьютер в качестве next hop для какой-либо сети N. Очевидно, что в данном случае весь трафик, направленный в данную сеть будет проходить через этот компьютер, который может либо просто сканировать этот трафик, либо фальсифицировать его. Использование процедуры установления подлинности при приеме обновлений маршрутов позволит если и не избежать появлений подобных нарушений информационного обмена, то сделать их реализацию значительно более трудоемкой для злоумышленника.

Примечательной особенностью протокола маршрутизации OSPF является использование адресов типа multicast для информационного обмена между маршрутизаторами в процессе определения маршрута. Использование таких адресов позволяет отказаться от использования адресов типа Broadcast, что, в свою очередь, приводит к повышению эффективности использования вычислительных ресурсов сети. Для обеспечения информационного обмена в протоколе маршрутизации OSPF предусмотрено использование нескольких Multicast-адресов. Отдельные адреса применяются для передачи и приема информации о маршрутах в информационной системе, что обеспечивает дополнительное повышение эффективности использования информационных ресурсов системы.

В отличие от протокола маршрутизации RIP, который для сравнения маршрутов может использовать только их длину, выраженную в числе переходов, протокол маршрутизации OSPF использует для этой же цели специальный критерий, который называется метрика. Метрика маршрута в протоколе OSPF формируется по специальному алгоритму и учитывает следующие параметры:

- пропускную способность канала;
- величину задержки распространения сигнала в канале;
- надежность канала;
- загруженность канала;
- размер максимального блока данных, который может быть передан через данный канал.

Применение такой метрики позволяет более объективно оценивать маршруты и, при наличии выбора, принимать эффективное и целесообразное решение. В качестве примера эффективного использования комплексной метрики можно рассмотреть сеть, в которой два узла связаны одним каналом ТЧ (тональной частоты, т. е. частоты в диапазоне от 300 Гц до 3400 Гц)

или двумя переходами по LAN. При использовании протокола маршрутизации RIP преимущество отдается маршруту, проходящему через один канал ТЧ. Очевидно, что большую пропускную способность в данном случае будет иметь другой маршрут, который и будет выбран при использовании комплексной метрики протоколом OSPF.

Кроме перечисленных выше преимуществ, использование протокола OSPF обеспечивает также ряд дополнительных возможностей, к которым относятся:

- иерархическая маршрутизация;
- возможность построения и обслуживания параллельных маршрутов (для обеспечения использования процедуры Load balancing).

Формирование маршрутных таблиц алгоритмами "Link-State"

Иерархическая маршрутизация предполагает отказ от принципа равноправия маршрутизаторов, практикуемого некоторыми алгоритмами Distance-Vector. Выделение областей, ограничивающих распространение маршрутной информации, а также жесткое распределение функций по формированию такой информации между маршрутизаторами каждой такой области позволяют существенно повысить эффективность использования каналов передачи данных за счет сокращения доли передаваемого по ним необязательного служебного трафика.

Разбиение сети на логические области производится по территориальному или функциональному принципу таким образом, чтобы основная доля информационного обмена была сосредоточена внутри области. Использование такой схемы логического построения сети позволяет существенно уменьшить долю неинформативного трафика, передаваемого по каналам передачи данных между областями.

Важной особенностью процесса организации информационного обмена у протокола маршрутизации OSPF является также использование аппарата "назначенных" (designated) маршрутизаторов. Использование этой возможности позволяет существенно сократить объем служебного трафика в том случае, когда несколько маршрутизаторов подключены к одной сети. В этом случае каждый из маршрутизаторов вправе претендовать на то, чтобы предоставлять информацию о внешних маршрутах узлам, которые подключены к данной сети. Для того чтобы избежать дублирования этой информации, только один из маршрутизаторов выбирается назначенным для данной сети и используется для передачи данных.

На рис. 4.9 приведен пример иерархической системы, состоящей из двух функционально независимых областей — 1 и 2. Определение взаимных маршрутов и информационный обмен между областями в данном случае производится через центральную область — Область 0. Построение маршрутов

в пределах каждой из этих областей может быть произведено независимо. Более того, абонентам области 1, например, не нужна информация о внутренних маршрутах области 2 и наоборот. Так, абоненту сети 10.1.3.0/24 совершенно не нужно знать, через какой маршрутизатор — M6 или M7 — можно достичь сети 10.2.2.0/24, поскольку маршрут в любом случае будет пролегать через M2 и M5. Поэтому маршрутная информация, направляемая в соседнюю область, может быть просуммирована с использованием принципа CIDR (Classless Inter-Domain Routing). При использовании этого принципа несколько маршрутов, ведущие в подсети некоторой сети, могут быть объединены в маршрут, ведущий в объединяющую их сеть. Применительно к данному варианту построения сети маршрутизатор M2 может сформировать и обслуживать единый суммарный маршрут для всей области 2: 10.2.0.0/24 через M5, что, естественно, приведет к сокращению объемов передаваемой маршрутной информации.

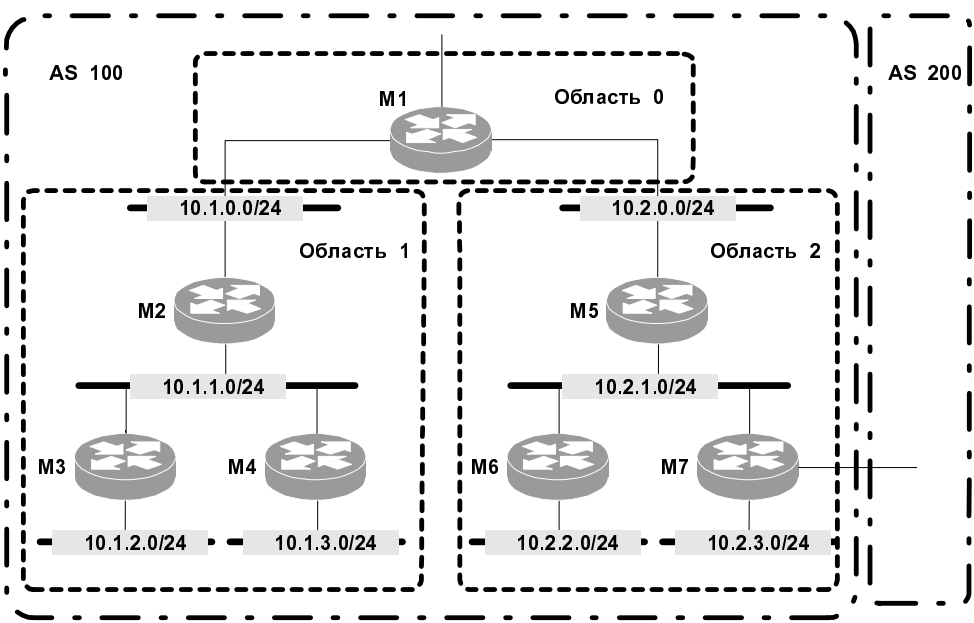


Рис. 4.9. Иерархия маршрутной информации

Иерархия областей и роли маршрутизаторов OSPF

Как и все протоколы маршрутизации класса "Link-State", протокол OSPF предназначен для построения внутренних маршрутов автономной системы (Autonomous System). *Автономной системой* (AS) мы будем называть группу маршрутизаторов, использующих единый процесс маршрутизации для обеспечения взаимного обмена маршрутной информацией.

Поскольку протокол OSPF обеспечивает иерархическую маршрутизацию, автономная система разбивается на независимые области по функциональному принципу. Как уже отмечалось, центральная (магистральная) область играет роль магистрали (backbone) и используется для обеспечения информационного взаимодействия между остальными (периферийными) областями. В зависимости от того, к какой области принадлежит маршрутизатор и какие информационные потоки через него проходят, различают четыре типа маршрутизаторов протокола OSPF:

- внутренний маршрутизатор (IR — Internal Router);
- пограничный маршрутизатор области (ABR — Area Border Router);
- пограничный маршрутизатор автономной системы (ASBR — AS Boundary Router);
- магистральный маршрутизатор (BR — Backbone Router).

Внутренний маршрутизатор (IR) не имеет непосредственных связей с маршрутизаторами других областей. Внутренними являются маршрутизаторы M3, M4 и M6, изображенные на рис. 4.9. Внутренние маршрутизаторы магистральной области называются магистральными маршрутизаторами (BR). Пограничные маршрутизаторы периферийных областей (M2 и M5) связаны маршрутизаторами магистральной области, в данном случае — M1.

Представление информации о состоянии компонентов сети (LSA)

Все маршрутизаторы OSPF принимают участие в формировании маршрутной информации автономной системы путем передачи специальных сообщений, содержащих информацию о текущем состоянии фрагмента сети. Эти сообщения называются LSA (Link State Advertisement) — объявление состояния канала. В том случае, если объявление представляет состояние маршрутизатора, оно должно содержать информацию о статусе его интерфейсов и близких ему маршрутизаторов. Каждое такое объявление распространяется по всей автономной системе. Совокупность таких LSA формирует базу данных маршрутизации в каждом из маршрутизаторов.

Сообщения LSA обязательно формируются при любом изменении состояния контролируемого компонента сети. Для обеспечения большей надежности сообщения LSA могут быть сформированы и при отсутствии каких-либо изменений в сети через достаточно большие интервалы времени, например, один раз за полчаса.

В частности, такие сообщения формируются в том случае, если в сети появился новый маршрутизатор или изменилось состояние канала передачи данных. Принятые сообщения образуют в каждом маршрутизаторе базу данных состояния сети (Link State Data Base). При получении сообщения об изменениях в структуре сети маршрутизатор вносит соответствующие

изменения в свою копию базы данных. Таким образом, в каждый момент времени все базы данных маршрутизаторов, находящихся внутри одной автономной системы, являются идентичными и адекватно отображают структуру этой системы. Для того чтобы определить маршрут, по которому должна быть передана дейтаграмма, маршрутизатор на основании своей копии базы данных строит дерево кратчайших путей, в вершине которого размещает себя самого. Построение кратчайших путей маршрутизатор выполняет всякий раз, когда происходит изменение состояние сети.

В зависимости от характера изменений состояния сети, маршрутизаторы протокола OSPF могут формировать несколько различных типов LSA. Сообщения каждого типа формируются маршрутизаторами одного или нескольких типов (IR, ABR и так далее). Различают пять типов сообщений LSA протокола маршрутизации OSPF:

- LSA type 1 — (тип 1), объявление каналов маршрутизатора;
- LSA type 2 — (тип 2), объявление сети Broadcast;
- LSA types 3, 4 — (тип 3, 4), объявление суммированных каналов;
- LSA type 5 — (тип 5), объявление внешних каналов.

Сообщения LSA типа 1

Сообщения первого типа (router link advertisement) формируются маршрутизатором для каждой области, в которой он имеет активные интерфейсы. Сообщения LSA этого типа содержат объединенную информацию о состоянии каналов маршрутизатора в данной области и идентификатор этого маршрутизатора. При формировании сообщений LSA для идентификации маршрутизаторов используются IP-адреса их интерфейсов. По умолчанию идентификатором маршрутизатора становится наименьший адрес его активного интерфейса. В некоторых случаях может оказаться целесообразным использование фиктивного адреса, который присвоен его виртуальному интерфейсу маршрутизатора. Сообщения первого типа распространяются только в пределах одной области.

Сообщения LSA типа 2

Сообщения LSA второго типа (network link advertisement) формируются только в сетях Broadcast (Ethernet) или NBMA (Non Broadcast Multi Access). Для того чтобы исключить возможность многократного представления информации об одной сети, для подключенных к ней маршрутизаторов осуществляется специальная процедура выбора. По итогам выполнения этой процедуры один из маршрутизаторов (DR — Designated Router) выбирается в качестве назначенного для данной сети и формирует для нее сообщения второго типа. Сообщение LSA типа 2 содержит идентификаторы всех маршрутизаторов, подключенных к указанной сети.

Сообщения LSA типа 3, 4

Сообщения LSA типа 3, 4 — (summary link advertisement) формируются пограничными маршрутизаторами области и направляются в соседние области. Каждое сообщение данного типа представляет собой суммарный маршрут, который может быть использован для информационного обмена с указанной областью в пределах одной автономной системы. В частности, LSA типа 3 описывают маршруты к сетям, LSA типа 4 — к пограничным маршрутизаторам автономной системы, находящимся в данной области.

Сообщения LSA типа 5

Сообщения LSA типа 5 (external link advertisement) формируются пограничными маршрутизаторами автономной системы и содержат информацию о маршрутах, внешних по отношению к данной автономной системе. Информация сообщений данного типа распространяется по всем областям автономной системы за исключением отдельных специально выделенных областей. Такие области обычно не имеют никаких внешних связей, кроме связи с магистральной областью, и потому называются тупиковыми (Stub-Area). Очевидно, что нецелесообразно направлять конкретную информацию о внешних сетях в такую область, поскольку для нее "все дороги ведут в Рим", в данном случае — в магистральную область.

Построение и обслуживание топологической базы маршрутов протокола OSPF

Существенной особенностью алгоритма протокола маршрутизации OSPF является специальная процедура информационного обмена между маршрутизаторами в сетях с множественным доступом. Маршрутизаторы, подключенные к одной и той же сети, называются *соседними маршрутизаторами* (Neighboring Routers). Маршрутизаторы протокола OSPF устанавливают и обслуживают соседские отношения, используя специальный дополнительный протокол Hello. С помощью этого протокола определяется состав подключенных к сети маршрутизаторов, динамически определяется их работоспособность и производится выбор одного из них в качестве назначенного маршрутизатора. Назначенный маршрутизатор формирует сообщения, содержащие список подключенных к сети маршрутизаторов, и передает содержимое текущей базы данных по запросу, полученному от одного из них. Для передачи запроса назначенному маршрутизатору используется групповой адрес 224.0.0.6. Если по каким-либо причинам назначенный маршрутизатор перестал функционировать, его функции автоматически переходят к запасному назначенному маршрутизатору (BDR — Backup Designated Router), выбираемому одновременно с основным. Для обмена информацией о маршрутах соседние маршрутизаторы должны установить партнерские отношения (Adjacency). В некоторых случаях для установления партнерских отношений от маршрутизатора может потребоваться выполнение процедуры

аутентификации. При передаче своих сообщений LSA в сеть с множественным доступом обычный (не DR) маршрутизатор в качестве адреса назначения использует групповой адрес 224.0.0.6, таким образом это сообщение адресуется основному и запасному назначенным маршрутизаторам. На первый взгляд эта стадия информационного обмена может показаться излишней. Действительно, при использовании другого адреса (например, Broadcast) маршрутизатор мог непосредственно направить свою информацию всем своим соседям одновременно. Однако использование режима многоадресного обмена в чистом виде не позволяет обеспечить требуемую достоверность передачи данных, поскольку в данном случае невозможно установить факт потери информации одним из получателей. Для ретрансляции полученного сообщения в сеть с множественным доступом основной (или запасной) назначенный маршрутизатор использует в качестве адреса назначения групповой адрес 224.0.0.5, следовательно, это сообщение принимают все маршрутизаторы протокола OSPF данной сети. В тех случаях, когда маршрутизаторы протокола OSPF связаны прямыми каналами "точка-точка" описанная выше процедура не используется.

Для передачи маршрутной информации маршрутизаторы протокола OSPF используют различные типы объявлений о состоянии сетевых компонентов (LSA). После проверки корректности и актуальности информации, содержащейся в очередном полученном сообщении LSA типа 1—4, маршрутизатор при необходимости корректирует свою базу данных и отправляет сообщение своим соседям по автономной области. Процесс распространения LSA в пределах автономной системы называется *затоплением* (Flooding). Сообщения LSA типа 5 распространяются внутри всей автономной системы за исключением ее тупиковых областей.

Для хранения маршрутной информации маршрутизаторы протокола OSPF используют специальные топологические базы данных, которые называются Link-State Database. База данных формируется из принятых маршрутизатором сообщений LSA и отображает текущее состояние и структуру информационных связей в рассматриваемой области маршрутизации. Содержимое баз данных у всех маршрутизаторов, расположенных в пределах одной области, должно быть идентичным.

Топологическая база данных представляется в виде таблицы, столбцы и строки которой образуют идентификаторы маршрутизаторов и сетей данной области. Содержимое ячейки этой таблицы, находящейся на пересечении столбца M*n* и строки N*m* соответствует стоимости непосредственной доставки дейтаграммы из маршрутизатора M*n* в сеть N*m*. Если указание стоимости доставки отсутствует, это означает, что маршрутизатор M*n* и сеть N*m* непосредственно не связаны. На основании данной таблицы каждый маршрутизатор строит дерево кратчайших маршрутов, соединяющих его самого с остальными компонентами области, и собственно таблицу маршрутизации.

Несмотря на то, что описанный способ построения маршрутных таблиц кажется чрезмерно громоздким по сравнению со способом, используемым алгоритмами "Distance-Vector", именно он обеспечивает ряд неоспоримых преимуществ алгоритмов "Link-State". К таким преимуществам относятся ранее отмеченные возможность использования нескольких параллельных маршрутов и практически мгновенное завершение переходных процессов при изменении состояния системы.

Сообщения протокола OSPF

Для передачи технологических сообщений протокола OSPF используются специализированные дейтаграммы с установленным типом протокола №89. Эти дейтаграммы состоят из заголовка, формат которого является одинаковым для всех типов сообщений, и информационной части с переменной структурой. Формат заголовка сообщения протокола OSPF приведен на рис. 4.10.

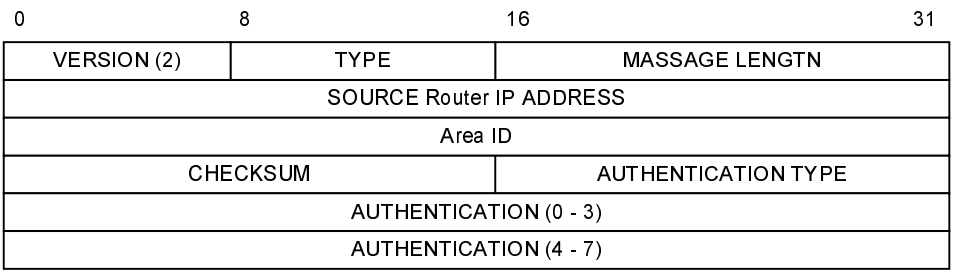


Рис. 4.10. Формат заголовка сообщения протокола OSPF

В поле **VERSION** размещается номер версии протокола OSPF. Значение поля **TYPE** определяет тип сообщения протокола OSPF. Возможные типы сообщений и их назначение приведены в табл. 4.1.

Таблица 4.1. *Типы и назначение сообщений протокола OSPF*

TYPE	Тип сообщения	Назначение сообщения
1	Hello	Протокол Hello
2	Database Description	Описание базы данных
3	Link-State request	Запрос состояния канала
4	Link-State update	Обновление состояния канала
5	Link-State ack	Подтверждение состояния канала

В поле **MESSAGE LENGTH** содержится выраженная в байтах длина сообщения протокола OSPF. В поле **SOURCE Router IP ADDRESS** размещается IP-адрес, выполняющий в данном случае функцию идентификатора источника сообщения протокола OSPF. В поле **Area ID** помещается 32-битный номер той области автономной системы, где было сформировано сообщение. В некоторых реализациях алгоритма маршрутизации OSPF для написания и отображения номера области используется десятичная — точечная форма, принятая для IP-адресов. При использовании этой формы идентификатор для, например, нулевой области будет выглядеть следующим образом: 0.0.0.0. Обычно маршрутизатор помечает сообщение протокола OSPF идентификатором области, где расположен интерфейс, через который оно было передано. Исключение из этого правила могут представлять сообщения, передаваемые через виртуальные каналы, организуемые внутри области. Виртуальные каналы в протоколе OSPF применяются для того, чтобы привести топологическую структуру автономной системы к каноническому виду: центральная магистральная область, связанная радиальными каналами с периферийными областями. При использовании алгоритма маршрутизации протокола OSPF в реальной сети у администратора может появиться потребность в том, чтобы объединить несвязанные фрагменты магистральной области. Виртуальный канал может быть использован и в том случае, если необходимо сконфигурировать некоторый участок сети, непосредственно не связанный с магистральной областью в качестве отдельной OSPF-области.

В поле **CHECKSUM** размещается двухбайтовая контрольная сумма сообщения протокола OSPF. При формировании этой суммы не учитывается содержимое байтов аутентификации. В том случае, если длина сообщения не может быть представлена целым числом 16-разрядных чисел, последнее слово дополняется справа необходимым количеством нулевых байтов.

В поле **AUTHENTICATION TYPE** размещается кодировка, соответствующая типу используемой схемы аутентификации источника маршрутной информации. Назначение кодировок, устанавливающих тип аутентификации протокола OSPF, приведено в табл. 4.2.

Таблица 4.2. *Типы используемых схем аутентификации протокола OSPF*

TYPE	Тип сообщения	Назначение сообщения
0	Null authentication	Аутентификация не используется
1	Simple password	Простой пароль
2	Cryptographic authentication	Криптографическая аутентификация
Более 2	Reserved by IANA	Зарезервировано IANA

Если содержимое поля **TYPE** равно 0, это означает, что для определения подлинности абонента никакая схема не используется. В таких случаях в двух последующих полях может быть размещена любая незначащая информация.

Применение простого пароля позволяет избежать ошибок маршрутизации, которые могут быть вызваны некорректным поведением маршрутизаторов. В данном случае использование простого 32-разрядного пароля позволит отбросить сообщения от маршрутизаторов, которые ошибочно считают себя находящимися в данной области. Использование этой схемы не позволяет, однако, противостоять умышленному искажению информации, поскольку открытый пароль легко можно перехватить и воспроизвести.

Для противодействия атакам, направленным на умышленное искажение маршрутной информации, следует использовать криптографическую схему аутентификации протокола OSPF. В этом случае каждое OSPF-сообщение наделяется уникальным сочетанием ключевого кода и последовательного номера сообщения. Поскольку в этом случае код шифрования не передается в явном виде, содержимое кодовых полей не может быть использовано вторично.

Сообщение HELLO

Для установления взаимных отношений маршрутизаторы протокола OSPF используют протокол Hello. Использование этого протокола обеспечивает выполнение следующих функций:

- поиск соседних маршрутизаторов в сегменте сети;
- выбор маршрутизаторов DR и BDR;
- определение статуса канала;
- установление партнерских отношений между маршрутизаторами.

Для того чтобы маршрутизаторы протокола OSPF стали партнерами, необходимо выполнение следующих условий:

- маршрутизаторы должны быть подключены к одному сегменту сети и принадлежать одной логической области протокола OSPF;
- оба маршрутизатора должны удовлетворительно пройти процесс аутентификации, в том случае, если это необходимо для установления партнерских отношений;
- установленные у маршрутизаторов значения таймеров Hello и Dead Interval должны совпадать (назначение таймеров приведено далее по тексту);
- установленные у маршрутизаторов признаки тупиковой области (stub area flag) также должны совпадать.

Пакеты протокола Hello периодически отправляются через все интерфейсы маршрутизатора. В этих пакетах маршрутизатор размещает IP-адреса соседних маршрутизаторов, от которых он получил сообщения Hello. Когда

маршрутизатор обнаруживает в принятом от партнера пакете Hello свой собственный идентификатор, это является признаком установления партнерских отношений.

В сетях с множественным доступом Broadcast (Ethernet) или NBMA (Non Broadcast Multi Access, — Frame Relay) протокол Hello используется для выбора назначенного маршрутизатора.

В сетях типа Broadcast маршрутизатор протокола OSPF для передачи пакетов Hello использует multicast адреса 224.0.0.5 или 224.0.0.6. Эти пакеты содержат представления данного маршрутизатора по поводу кандидатуры назначенного маршрутизатора, а также список маршрутизаторов, с которыми у отправителя установлены партнерские отношения.

В сетях типа NBMA по понятным причинам такой режим адресации не может быть использован для доставки сообщений Hello. Поэтому в таких сетях маршрутизатор направляет свои пакеты Hello всем своим соседям в режиме Unicast, используя при этом статически определяемый список.

На рис. 4.11 приведена структура сообщения Hello. Первые 24 байта занимает стандартный заголовок сообщения протокола OSPF с установленным признаком сообщения Hello (TYPE=1).

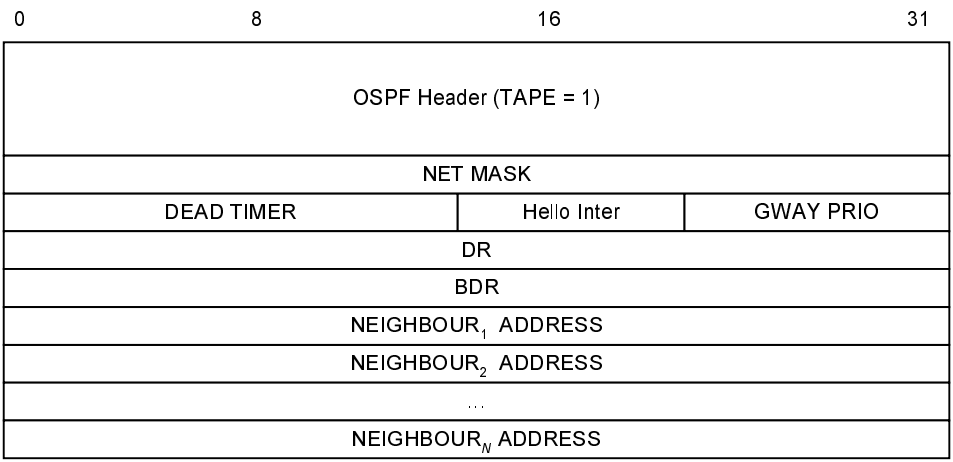


Рис. 4.11. Структура сообщения Hello

В поле **Net MASK** размещается маска Broadcast или NBMA сети. В поле **DEAD TIMER** размещается максимальное значение интервала времени (в секундах), в течение которого маршрутизатор может ожидать получение очередного сообщения Hello от соседнего маршрутизатора. При получении очередного сообщения таймер вновь устанавливается в максимальное значение и перезапускается. По истечении этого интервала соседний маршрутизатор объявляется "мертвым". Значение **DEAD TIMER** должно быть

одинаковым у всех маршрутизаторов, подключенных к одной сети. В поле **Hello Inter** размещается выраженное в секундах значение периода посылки сообщений Hello. В поле **GWAY PRIO** размещается приоритет маршрутизатора, учитываемый при выборе назначенного маршрутизатора, причем чем больше значение — тем выше приоритет. В полях **DR** и **BDR** маршрутизатор размещает свои предложения по кандидатурам на посты назначенного маршрутизатора и его заместителя. В остальной части сообщения Hello располагаются идентификаторы соседних маршрутизаторов (**NEIGHBOUR ADDRESS**).

Сообщения Database Description

Сообщения Database Description передаются в том случае, если возникает необходимость в передаче всей базы данных маршрутизации. На рис. 4.12 представлена структура сообщения Database Description.

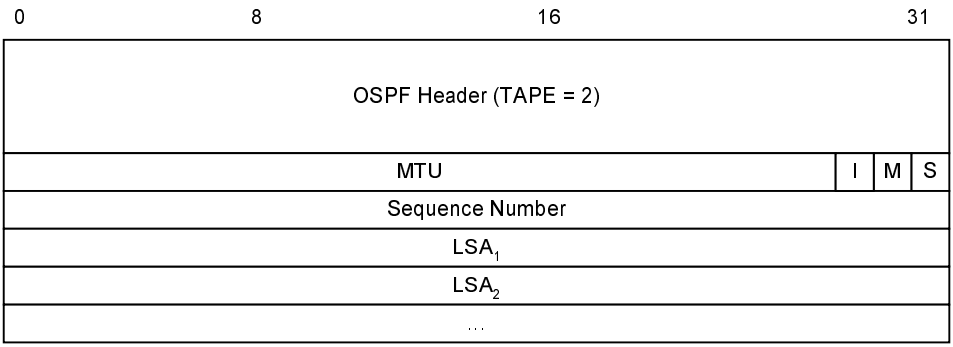


Рис. 4.12. Структура сообщения Database Description

В процессе передачи базы данных участвуют два маршрутизатора протокола OSPF. Один из маршрутизаторов, выполняющий функцию Slave (подчиненный) (**S**=0), передает запрос на получение базы данных. Маршрутизатор типа Master (главный) (**S**=1) формирует ответное сообщение, содержащее полную базу данных области маршрутизации. В том случае, если передаваемая база занимает более одного IP-пакета, в сообщении размещаются специальные признаки начала (**I**=1) и продолжения (**M**=1). Для того чтобы обеспечить возможность восстановления потерянных сообщений, каждое из них снабжается индивидуальным последовательным номером (**Sequence Number**).

Сообщения Link-State Request

Сообщения Link-State Request используются для получения фрагмента базы данных маршрутизации. На рис. 4.13 представлена структура сообщения Link-State Request.

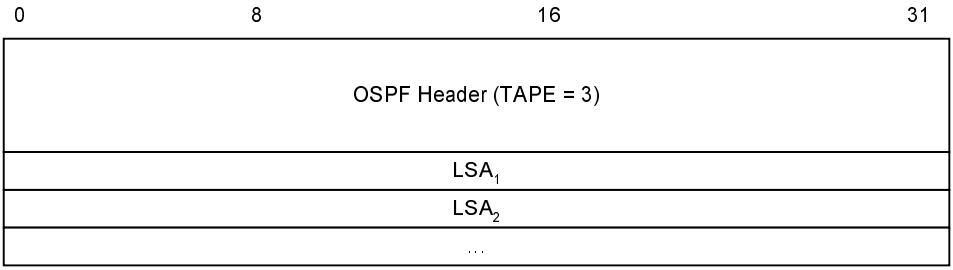


Рис. 4.13. Структура сообщения Link-State Request

Непосредственно после заголовка сообщения Link-State Request инициатор запроса помещает фрагмент своей базы данных, нуждающийся в обновлении.

Сообщения Link-State Update

Сообщения Link-State Update используются для передачи фрагмента базы данных маршрутизации. Структура сообщения Link-State Update представлена на рис. 4.14.

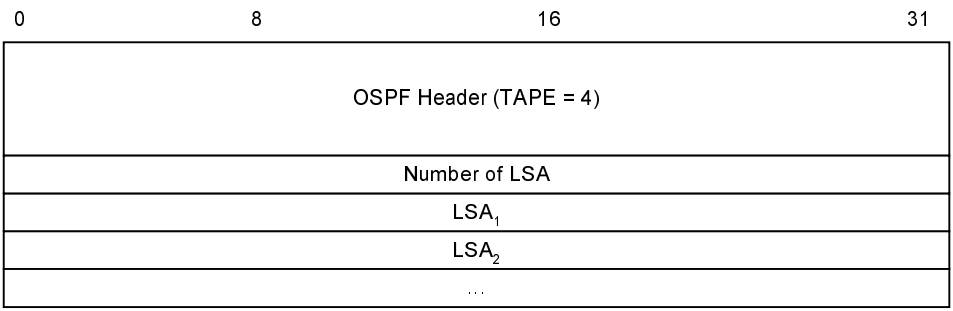


Рис. 4.14. Структура сообщения Link-State Update

Структура сообщения Link-State Update практически совпадает со структурой сообщения Link-State Request. В дополнительном по отношению к сообщению Link-State Request поле **Number of LSA** помещается объем передаваемого фрагмента базы данных, выраженный в количестве LSA.

Сообщения Link State Acknowledgment.

Сообщения Link-State Acknowledgment используются для подтверждения получения фрагмента базы данных маршрутизации. Структура сообщения Link-State Acknowledgment представлена на рис. 4.15.

Для передачи сообщения Link-State Acknowledgment могут быть использованы все режимы адресации, применяемые в алгоритме протокола маршрути-

зации OSPF. Структура сообщения Link-State Acknowledgment совпадает со структурой сообщения Link-State Request. Отличие состоит в том, что вместо собственно сообщений LSA передаются только их заголовки.

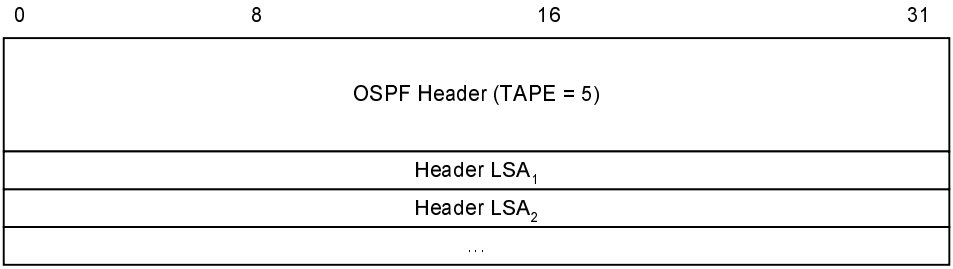


Рис. 4.15. Структура сообщения Link-State Acknowledgment

Пример практического применения алгоритма протокола OSPF

На рис. 4.16 приведена схема участка сети, в которой для построения маршрутов используются алгоритмы протоколов RIP и OSPF (процесс номер 77). В частности протокол маршрутизации OSPF включен на интерфейсах S0 и E0 маршрутизатора M2.

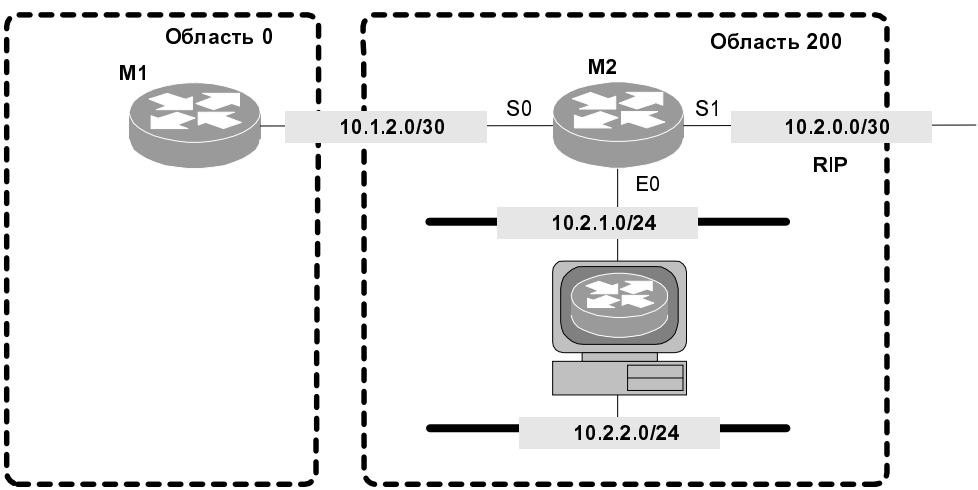


Рис. 4.16. Пример практического применения алгоритма протокола OSPF

В данном случае целесообразно, используя команду `ip ospf priority`, форсировать выбор маршрутизатора M2 в качестве DR для сети 10.2.1.0/24. В противном случае функция DR может перейти к серверу-маршрутизатору,

что приведет к неэффективному использованию сетевых ресурсов. Ниже приведены фрагменты конфигураций маршрутизаторов M1 и M2.

Фрагмент конфигурационного файла маршрутизатора M1

```
...
router ospf 77
network 10.1.2.0 0.0.0.3 area 0
...
```

Фрагмент конфигурационного файла маршрутизатора M2

```
...
interface S0
ip address 10.1.2.2 255.255.255.252
interface S1
ip address 10.2.0.2 255.255.255.252
interface E0
ip address 10.2.1.1 255.255.255.0
ip ospf priority 100
...
router ospf 77
network 10.1.2.0 0.0.0.3 area 0
network 10.2.2.0 0.0.0.255 area 200
redistribute rip  subnets
router rip
version 2
redistribute ospf 77  subnet
passive-interface Ethernet 0
passive-interface Serial 0
...
```

Здесь команды `redistribute` используются для взаимного обмена маршрутной информацией между протоколами RIP и OSPF. Маршрутизатор M2 в этой сети одновременно выполняет функции пограничного для области 200 и всей автономной системы.

Протокол маршрутизации EIGRP

Протокол маршрутизации EIGRP (Enhanced IGRP) был разработан специалистами компании Cisco и представляет дальнейшее развитие принципов, положенных в основу протокола IGRP. Благодаря своим высоким эксплуа-

тационным характеристикам протокол EIGRP может быть эффективно использован для обеспечения маршрутизации в сложных современных комплексных сетях. Характерными особенностями протокола маршрутизации EIGRP являются:

- обеспечение маршрутизации для внеклассовых сетей IP;
- построение и обслуживание нескольких параллельных маршрутов с различной пропускной способностью;
- эффективное использование пропускной способности каналов при передаче маршрутной информации;
- обеспечение маршрутизации для нескольких различных протоколов сетевого уровня одновременно (IP, IPX, Apple Talk).

Формально протокол EIGRP относится к алгоритмам маршрутизации Distance-Vector, однако он обладает рядом преимуществ, свойственных протоколам маршрутизации Link-State, и поэтому может быть отнесен к особому типу протоколов маршрутизации — к гибридным протоколам.

Особенности гибридных протоколов маршрутизации

Каждый из известных типов протоколов маршрутизации Link-State и Distance-Vector имеет свои положительные и отрицательные свойства. К достоинствам алгоритмов маршрутизации Distance-Vector несомненно можно отнести простоту их реализации, тогда как алгоритмы Link-State обеспечивают лучшие эксплуатационные характеристики. Недостатком протоколов маршрутизации Distance-Vector является неэффективное использование пропускной способности каналов при передаче маршрутной информации, алгоритмы Link-State плохо приспособлены для построения маршрутов в сетях с нестабильными каналами. Гибридные протоколы маршрутизации, к которым относится EIGRP, сочетают в себе положительные свойства протоколов Link-State и Distance-Vector и максимально свободны от их недостатков.

Основные проблемы любого протокола маршрутизации всегда связаны с нестабильностью маршрутов. В любой системе всегда существует хотя бы один нестабильно работающий элемент или элемент, выходящий из строя в самый неподходящий момент. Чем больше система, тем выше вероятность отказа или сбоя. Отказы и сбои в системах передачи данных приводят к пропаданию маршрутов. Задача алгоритма маршрутизации в данном случае состоит в том, чтобы как можно скорее найти обходной маршрут взамен вышедшего из строя. Именно в такие моменты некоторые алгоритмы маршрутизации проявляют себя не лучшим образом — поиск резервного маршрута и переключение на него может выполняться непозволительно долго. В худшем случае неисправность канала может вообще привести к возникновению циклического маршрута, вследствие чего часть переданной через сеть информации будет безвозвратно потеряна.

Основу протокола маршрутизации EIGRP составляют алгоритмы и процедуры, используемые для исключения возможности появления маршрутных петель, а также для увеличения скорости обновления маршрутной информации при изменении состояния системы (сходимости протокола маршрутизации). Для повышения скорости сходимости протокола EIGRP, в частности, используются механизм передачи частичных обновлений и алгоритм DUAL — Diffuse Update Algorithm (Алгоритм диффузных обновлений).

Алгоритм DUAL

Теоретические основы алгоритма DUAL были заложены в работах доктора J. J. Garcia Luna Aceves (компания SRI International). Одна из основных идей алгоритма DUAL проста до гениальности. Действительно, трудно не согласиться с тем, что искать запасные маршруты лучше тогда, когда система функционирует нормально, а не тогда, когда основной маршрут уже вышел из строя. Естественно, для того, чтобы искать и обслуживать резервные каналы в дополнение к основным, маршрутизатор должен выделять существенно больше своих ресурсов (объем ОЗУ, время процессора). Однако эти затраты в подавляющем большинстве случаев себя оправдывают, поскольку применение алгоритма DUAL обеспечивает существенное уменьшение времени сходимости протокола маршрутизации за счет практически мгновенного переключения на резервный маршрут при выходе из строя основного. Кроме того, легко предположить, что, если маршрутизатор будет передавать маршрутную информацию не по принципу "кто не спрятался — я не виноват", а в ответ на конкретный запрос, полученный от соседнего маршрутизатора, резко сократится доля недостоверной и ненужной информации, передаваемой по сети.

Протокол маршрутизации EIGRP, так же, как и протокол OSPF, предполагает наличие партнерских отношений между соседними маршрутизаторами для организации обмена маршрутной информацией между ними. Для представления маршрута к каждой сети из числа непосредственных соседей данного маршрутизатора могут быть выбраны два маршрутизатора-партнера, а именно Successor (последователь) и Feasible Successor (вероятный последователь). Выбор ответственных маршрутизаторов производится таким образом, чтобы, с одной стороны, обеспечить нахождение кратчайшего маршрута, а с другой стороны, избежать появления маршрутных петель. В качестве критерия оптимальности маршрута используется сумма стоимостей составляющих его каналов. В протоколе EIGRP критерий маршрута называют *дистанцией* (Distance). Каждому из представляемых маршрутов соответствуют два значения дистанции:

- Advertised Distance — представляемая дистанция;
- Feasible Distance — ожидаемая дистанция.

Advertised Distance (представляемая дистанция) представляет собой метрику маршрута, представляемого одним из партнеров данного маршрутизатора

до некоторой искомой сети N. При вычислении Advertised Distance не учитывается стоимость последнего участка, соединяющего маршрутизатор с партнером, представляющим данный маршрут.

Ожидаемая дистанция соответствует значению Advertised Distance для данного маршрута, увеличенному на стоимость последнего участка маршрута — от представляющего маршрутизатора до конечного.

Для того чтобы получить метрики маршрутов до сети N, маршрутизатор направляет своим партнерам соответствующие запросы. В ответ каждый из опрошенных маршрутизаторов направляет характеристику имеющегося у него маршрута до искомой сети. После получения всех ответов маршрутизатор производит выбор последователя и вероятного последователя для сети N.

Последователем для сети N становится маршрутизатор, представивший минимальное значение Advertised Distance. Этот маршрутизатор в дальнейшем используется в качестве Next Hop для доставки пакетов в сеть N.

Вероятным последователем для сети N становится тот маршрутизатор из числа непосредственных соседей, которому соответствует значение Advertised Distance меньшее, чем значение Feasible Distance маршрута, проходящего через маршрутизатор Successor (последователь). Маршрут, проходящий через Feasible Successor (вероятный последователь), используется в системе в качестве резервного. В том случае, если основной маршрут по каким-либо причинам не может быть использован для передачи данных, маршрутизатор должен произвести переключение на резервный маршрут. В том случае, если резервный маршрут для сети N отсутствует, маршрутизатор протокола EIGRP должен направить запрос к своим непосредственным соседям относительно этого маршрута. Если у соседа есть маршрут к этой сети, он направляет в ответ на запрос частичное обновление таблицы маршрутизации. В том случае, если искомая сеть отсутствует в его таблице маршрутизации, сосед должен передать запрос своим соседям — и т. д.

Относительная сложность алгоритма, используемого для определения резервного пути, объясняется необходимостью построения маршрута, не содержащего петель.

На рис. 4.17 приведен вариант сети, в котором при изменении маршрута используется маршрутизатор — вероятный последователь. В соответствии с правилом алгоритма DUAL маршрутизатор M4 определяет в качестве своего последователя для доставки в сеть N маршрутизатор M2. В данном случае для сети N может быть также определен маршрутизатор вероятный последователь. Поскольку значение представляемой дистанции для сети N у маршрутизатора M2 меньше, чем ожидаемая дистанция последователя M4, маршрутизатор M3 становится вероятным последователем M4 для этой сети. Результаты расчетов представляемых и ожидаемых дистанций для данного примера приведены в табл. 4.3.

Таблица 4.3. *Результаты расчета дистанций для маршрутизатора M4*

Сеть	Партнер	Представляемая дистанция	Ожидаемая дистанция
N	M2	3	5
N	M3	4	6

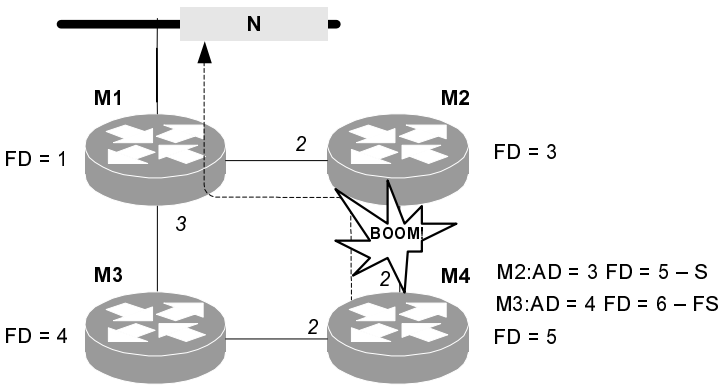


Рис. 4.17. Использование вероятного последователя протокола маршрутизации EIGRP

Примечание

На рисунках приняты следующие обозначения: AD — Advertised Distance, FD — Feasible Distance, S — Successor, FS — Feasible Successor.

Поскольку маршрут, представляемый вероятным последователем, гарантированно не содержит петель, переключение на него осуществляется после выхода из строя основного маршрута без каких-либо дополнительных действий со стороны маршрутизатора M4. В соответствии с терминами алгоритма EIGRP, такое состояние маршрутизатора называется пассивным. В таком состоянии маршрутизатор находится тогда, когда сеть стабильна, или когда возникающие в сети изменения его не затрагивают. Таким образом, при аварии канала, соединяющего M2 и M4, последний установит в качестве последователя для сети N маршрутизатор M3 с ожидаемой дистанцией 6 и разошлет своим партнерам соответствующее частичное обновление таблицы маршрутизации, не выходя при этом из пассивного состояния.

На рис. 4.18 приведен вариант сети, в котором при изменении маршрута вероятный последователь не может быть использован. Так же, как и в предыдущем случае, в соответствии с правилом алгоритма DUAL маршрутизатор M4 определяет в качестве своего последователя для доставки в сеть N

маршрутизатор M2. Однако в данном случае для сети N не может быть определен вероятный последователь. Поскольку в этом примере значение представляемой дистанции для сети N у маршрутизатора M2 больше, чем ожидаемая дистанция последователя M4, маршрутизатор M3 не может стать вероятным последователем M4 для этой сети. Результаты расчетов представляемых и ожидаемых дистанций для данного примера приведены в табл. 4.4. Обратите внимание, что в этом примере маршрутизатор M5 обладает таким же значением представляемой дистанции, однако не имеет собственного маршрута в сеть N.

Таблица 4.4. *Результаты расчета дистанций для маршрутизатора M4*

Сеть	Партнер	Представляемая дистанция	Ожидаемая дистанция
N	M2	3	5
N	M3	6	8
N	M5	6	7

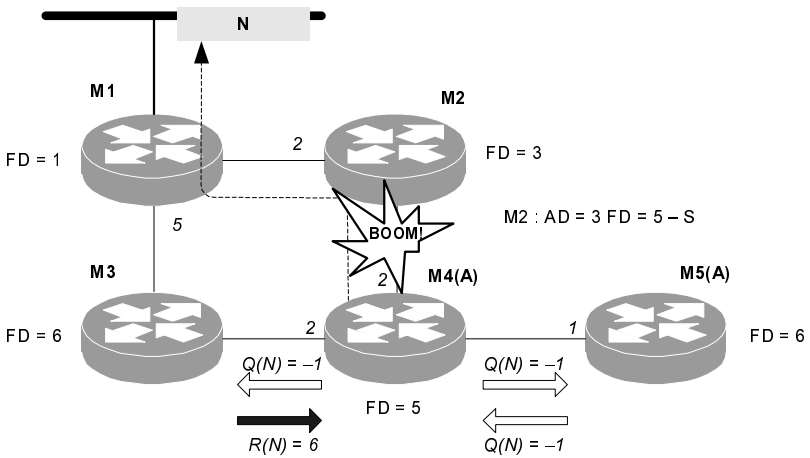


Рис. 4.18. Активное состояние маршрутизаторов протокола EIGRP

В случае аварии канала, соединяющего M2 и M4, последний перейдет в активное состояние, поскольку в его таблице маршрутизации теперь уже нет маршрута в сеть N. В данном случае маршрутизатор M4 должен уведомить партнеров о потере своего маршрута в сеть N, а также направить им запрос относительно нового маршрута в эту сеть. После получения уведомления от M4 маршрутизатор M3 останется в пассивном состоянии, поскольку его маршрут в сеть N никак не изменился. Поэтому в ответ на запрос от M4 маршрутизатор M3 отправит свое значение ожидаемой дистанции для сети N. В отличие от M3 маршрутизатор M5 после получения уведомления

от M4 перейдет в активное состояние. Это вызвано тем, что он также теперь не имеет маршрута в сеть N. Поэтому маршрутизатор M5 также посылает запрос своим партнерам — в том числе и M4. Поскольку маршрутизатор M4 все-таки получил от M3 новое значение маршрута, он считает свою задачу выполненной и производит соответствующие действия — корректирует таблицу маршрутизации, рассылает обновления и переходит в пассивное состояние.

Обратите внимание, что в приведенном примере произошло автоматическое отсечение ошибочного маршрута, представленного маршрутизатором M5. Очевидно, что именно такие маршрутизаторы являются потенциальными поставщиками циклических маршрутов.

Таким образом, использование алгоритма DUAL обеспечивает исключение петель на стадии построения нового маршрута.

Обеспечение быстрой сходимости и совместимости протоколов EIGRP и IGRP

Алгоритм протокола EIGRP обладает рядом дополнительных возможностей по обеспечению высокой скорости установления маршрута при изменении состояния сети и совместимости с протоколом IGRP.

Формат метрики протокола EGRP.

Так же, как и протокол IGRP, протокол маршрутизации EIGRP использует композитную метрику для оценки качества маршрута. Единственное отличие заключается в том, что в сообщениях протокола EIGRP поле метрики маршрута занимает 32 бита против 24 бит алгоритма IGRP. Компонентами композитной метрики протокола EIGRP являются характеристики маршрута с 1 по 4:

1. **B** — Bandwidth — минимальное значение пропускной способности среди каналов маршрута, определяемое соотношением: $B =$ = (10000000/<минимальное значение, установленное на интерфейсе (Кбит/с)> $\times$ 256;
2. **R** — Reliability — показатель надежности маршрута. При вычислении этого значения используется соотношение числа принятых и переданных сообщений протокола;
3. **D** — Delay — суммарная задержка распространения сигнала от источника до пункта назначения маршрута, вычисляемая по формуле: $D =$ <сумма задержек всех каналов по маршруту (мкс) > $\times$ 256;
4. **L** — Loading — соотношение загрузка/пропускная способность на наиболее загруженном участке маршрута;
5. **MTU** — минимальное значение maximum transfer unit у каналов, образующих маршрут;

6. **FD** — Feasible Distance — ожидаемая дистанция;
7. **AD** — Advertised Distance — представляемая дистанция.

При формировании композитной метрики M маршрута протокола EIGRP значения каждого из компонентов умножаются на весовой коэффициент K_i и суммируются.

$$M = [K_1 \times B + (K_2 \times B)/(256 - L) + K_3 \times D] \times [K_5/(R + K_4)].$$

Примечание

Во избежание появления маршрутных петель, категорически не рекомендуется изменять установленные по умолчанию значения коэффициентов K_i.

Реальное значение композитной метрики M маршрута протокола EIGRP рассчитывается по упрощенной формуле:

$$M = B + D.$$

Значения B и D в данном случае определяются по соотношениям, приведенным в пунктах 1 и 3 соответственно.

Поскольку рассчитанная подобным образом метрика маршрута протокола EIGRP ровно в 256 раз превышает значение метрики этого же маршрута, полученной протоколом IGRP, метрики этих протоколов могут быть легко преобразованы одна в другую. Эта особенность протокола EIGRP позволяет обеспечивать постепенность перехода на этот протокол маршрутизации сетей, использующих протокол IGRP.

Сокращение объема служебной информации

Одной из важнейших характеристик протокола маршрутизации является его эффективность, определяемая относительной величиной служебного трафика алгоритма маршрутизации. Наименьшей эффективностью обладают классические алгоритмы Distance-Vector, вследствие того, что они постоянно передают содержимое маршрутных таблиц, величина которых увеличивается пропорционально размеру системы.

Для уменьшения объемов служебной информации протокола маршрутизации, которая передается по каналам передачи данных, маршрутизаторы EIGRP используют механизм частичных обновлений, используемый в некоторых других протоколах маршрутизации (например, RIP II или OSPF). При этом частичные обновления обычно передаются в ответ на запрос конкретного соседа в режиме Unicast, что также способствует экономии сетевых ресурсов в сетях с множественным доступом. Такой принцип информационного взаимодействия довольно часто используется в современных протоколах маршрутизации, поскольку он позволяет существенно повысить эффективность использования канала передачи данных при передаче маршрутной информации и обеспечивает требуемый уровень безопасности и надежности информационного обмена.

При отправке некоторых сообщений протокола EIGRP в сеть с множественным доступом может быть также использован режим групповой адресации. В этом режиме адресации маршрутизаторы протокола EIGRP используют адрес 224.0.0.10.

Для передачи сообщений протокола маршрутизации EIGRP используются IP-пакеты со значением кода протокола равным 88.

Специфическим для протокола EIGRP является алгоритм использования резервных маршрутов. При наличии резервного маршрута маршрутизатор находится в пассивном состоянии, в случае аварии основного маршрута переключение на резервный маршрут произойдет мгновенно, автоматически и без дополнительного информационного обмена.

Отношения между маршрутизаторами. Назначение и типы сообщений протокола EIGRP

В отличие от классических протоколов Distance-Vector, алгоритм маршрутизации EIGRP не практикует периодическую передачу маршрутных таблиц между соседними маршрутизаторами. Вместо этого маршрутизаторы протокола EIGRP для передачи маршрутной информации используют несколько сообщений, позволяющих им строить специальные таблицы, отображающие текущее состояние сети, и обновлять их содержание. Для обеспечения большей достоверности информационного обмена большинство сообщений протокола EIGRP передается в режиме с подтверждением их получения.

Для построения маршрутов маршрутизаторы протокола EIGRP используют две разновидности маршрутных таблиц:

- таблица партнеров;
- топологическая таблица.

В партнерской таблице маршрутизатор протокола EIGRP хранит информацию о своих непосредственно подключенных соседях по процессу EIGRP. Эту информацию маршрутизатор собирает путем передачи приветственных сообщений протокола EIGRP HELLO и получения ответных сообщений. Таблица партнеров содержит идентификатор соседа (обычно это IP-адрес интерфейса, через который сосед производит отправку сообщений), текущий последовательный номер последнего принятого от соседа сообщения и значение таймера активности.

Последовательный номер присваивается каждому сообщению протокола EIGRP для повышения достоверности информационного обмена. Таймер активности используется для определения текущего операционного статуса

соседа. Исходное значение таймера активности обычно в три раза превышает период посылки регулярных сообщений протокола EIGRP HELLO.

Примечание

Непродуманное изменение частоты передачи сообщений HELLO может приводить к выключению одного или нескольких маршрутизаторов из информационного обмена протокола EIGRP.

При получении любого очередного сообщения от соседа маршрутизатор перезапускает таймер его активности. Если таймер активности соседа становится равным 0, маршрутизатор констатирует, что у него стало одним соседом меньше и выполняет, при необходимости, процедуры по изменению топологической таблицы и таблицы маршрутизации.

В топологической таблице маршрутизатор хранит информацию о маршрутах, полученную от соседей. Для каждой сети N в этой таблице содержится идентификатор соседа, направившего информацию о маршруте, и характеристики этого маршрута: ожидаемая дистанция, представляемая дистанция. В зависимости от соотношения представленных характеристик маршрутов один или несколько маршрутизаторов выбираются последователями для сети N. Один из маршрутизаторов может быть выбран вероятным последователем для сети N при выполнении соответствующего условия алгоритма DUAL. Все маршрутизаторы, выбранные последователями, соответствующим образом помечаются в топологической таблице. На основании содержимого топологической таблицы маршрутизатор протокола EIGRP строит таблицу маршрутизации, при этом в качестве next hop для сети N используется один или несколько маршрутизаторов из числа последователей.

После включения в сеть маршрутизатор протокола EIGRP устанавливает партнерские отношения с соседними маршрутизаторами и поддерживает эти отношения, что обеспечивает постоянную актуальность содержимого его топологической таблицы.

Сообщения HELLO/ACK

Вместо того чтобы постоянно перекачивать содержимое таблицы маршрутизации, маршрутизаторы EIGRP периодически рассылают короткие сообщения HELLO. Сообщения данного типа используются для определения операционного статуса соседствующих маршрутизаторов. Сообщение HELLO содержит косвенную информацию: "Я в порядке. Все ранее переданные мной маршруты действительны. Новых маршрутов нет". В отличие от некоторых других сообщений протокола EIGRP при получении сообщения HELLO маршрутизатор не обязан передавать подтверждение его отправителю. В качестве сообщения подтверждения — ACK — может быть использовано одно из очередных сообщений HELLO.

Сообщения Query и Reply

В тех случаях, когда маршрутизатор протокола EIGRP потерял или не имел маршрут до какой-либо сети N, он должен передать запрос (Query) об этом маршруте своим соседям. В зависимости от того, есть ли у соседа информация о данном маршруте, он либо сам отвечает на запрос, формируя сообщение Reply, либо передает аналогичный запрос своим соседям. Маршрутизатор, передавший запрос, переходит в активное состояние в отношении сети N. В этом состоянии он будет находиться до тех пор, пока не получит новый маршрут в сеть N от одного или нескольких своих соседей или пока не получит от каждого из них уведомление о недостижимости этой сети. После выполнения указанных действий маршрутизатор переходит в пассивное состояние по отношению к сети N.

Сообщения Query и Reply протокола EIGRP передаются в режиме с подтверждением получения — в ответ на переданное сообщение отправитель должен получить сообщение ACK (Acknowledgement).

Сообщения Update

Сообщения этого типа используются для передачи полной или частичной информации о маршрутах. Обычно сообщения Update передаются в режиме Unicast при появлении нового маршрутизатора в сети. Сообщение Update может быть сформировано также при изменении стоимости существующего маршрута. В последнем случае для передачи сообщения Update через сеть с множественным доступом используется режим Multicast. Сообщения Update также передаются в режиме с подтверждением получения — в ответ на переданное сообщение отправитель должен получить сообщение ACK от каждого из адресатов сообщения.

Сообщения Requests

Сообщения типа Requests протокола маршрутизации EIGRP используются сервером маршрутов для получения специфической информации от одного или нескольких соседних маршрутизаторов. Сообщения Update передаются в режиме без подтверждения получения.

Пример практического применения алгоритма протокола EIGRP

Гибкий и мощный протокол маршрутизации одновременно является относительно простым в использовании, что позволяет ему успешно конкурировать с таким протоколом, как OSPF. Ниже приведен пример конфигурации маршрутизатора, у которого в интерфейсе S5/0 используется протокол маршрутизации EIGRP, в то время как в интерфейсе E0 — протокол RIP II. Включение протокола EIGRP в последовательном интерфейсе в данном случае позволяет повысить эффективность использования его пропускной способности.

Пример конфигурационного файла маршрутизатора M

```
...
interface S5/0
ip address 192.168.80.1 255.255.255.252
...
interface E0
ip address 172.24.24.1 255.255.255.0
...
router eigrp 100
redistribute rip
distribute-list S5-OUT out S5/0
distribute-list S5-IN in S5/0
network 192.168.80.0
...
router rip
version 2
network 172.24.0.0
redistribute eigrp 100 metric 4
...
```

Здесь команды `redistribute` используются для взаимного обмена маршрутной информацией между протоколами RIP и EIGRP. Команды `distribute-list` используются для фильтрации маршрутов экспортируемых и импортируемых через интерфейс `S5/0`.

Протокол маршрутизации EGP

В силу ряда исторических причин на ранней стадии развития сети Интернет наибольшее распространение имела радиальная схема построения сети, называемая Core Architecture (централизованная архитектура).

Характерной особенностью сетей Core Architecture являлось наличие ядра (Core), к которому подключались все периферийные информационные системы. Ядро системы образовывали несколько маршрутизаторов, выполняющие задачи определения маршрута в сети.

Узлом (Node) сети Core Architecture мог быть как отдельный узел, так и совокупность узлов или сетей, объединенных по принципу территориальной или функциональной принадлежности. Информационное взаимодействие в этих сетях организовывалось через центральную структуру — ядро (Core). Каждый из центральных узлов был связан с одним или несколькими периферийными узлами и с несколькими маршрутизирующими узлами центральной структуры.

Маршрутизация в сетях Core Architecture выполнялась по двум различным схемам — одна из этих схем применялась для определения маршрута в периферийных узлах, другая применялась в центральных узлах. Периферийные узлы использовали статическую маршрутизацию, причем один из сформированных таким образом маршрутов обязательно должен был указывать на один из центральных ближайших узлов.

Центральные узлы накапливали информацию о непосредственно подключенных к ним узлах и сетях и имели возможность передавать ее другим узлам, входящим в центральную структуру. Таким образом, любой из узлов центральной структуры мог обеспечить выполнение решения задачи маршрутизации. Использование центральной структуры, содержащей несколько маршрутизирующих узлов, обеспечивало большую гибкость и надежность всей системы в целом.

Основным достоинством сетей Core Architecture являлась простота их реализации. В тех случаях, когда данная логическая структура соответствует принципам организации информационного обмена в системе, задача маршрутизации может быть успешно выполнена одним или несколькими центральными узлами.

Основным недостатком сетей, построенных по принципу Core Architecture, является трудность наращивания таких сетей и недостаточная гибкость маршрутизации. При появлении нового периферийного узла или сети в такой системе должны быть обязательно выполнены изменения существующих настроек маршрутизаторов центральной системы. Маршрут между двумя любыми периферийными узлами обязательно должен был проходить через центральную систему независимо от реального расстояния между периферийными узлами.

Выше было отмечено, что сети Core Architecture обладают рядом существенных недостатков, главным из которых является ограниченная возможность масштабирования.

Примечание

Масштабированием принято называть наращивание возможностей информационных систем по мере необходимости, осуществляемое в процессе их использования.

Со временем этот недостаток начинал оказывать все более существенное сдерживающее влияние на развитие глобальных информационных сетей. Появлялись и развивались частные сети, структура сети Интернет становилась более динамичной. Идея, положенная в основу сетей Core Architecture, уже не могла быть использована в условиях, когда развитые частные сети имели множественный доступ к глобальным сетям. Альтернативной концепцией построения логической структуры сетей по отношению к Core Architecture является концепция автономных систем.

Назначение и особенности протоколов внешней маршрутизации

Концепция автономных систем предполагает разбиение сети на отдельные области, в пределах которых управление процессами определения маршрута производится автономно с использованием независимых процессов маршрутизации. По сути, концепция автономных систем является логическим развитием идеи, положенной в основу сетей Core Architecture. Отличие заключается в том, что вместо локальных единичных сетей, фигурирующих в сетях Core Architecture, современные сети состоят из иерархических систем, включающих десятки локальных сетей. На рис. 4.19 представлена структурная схема организации информационного взаимодействия в автономных системах.

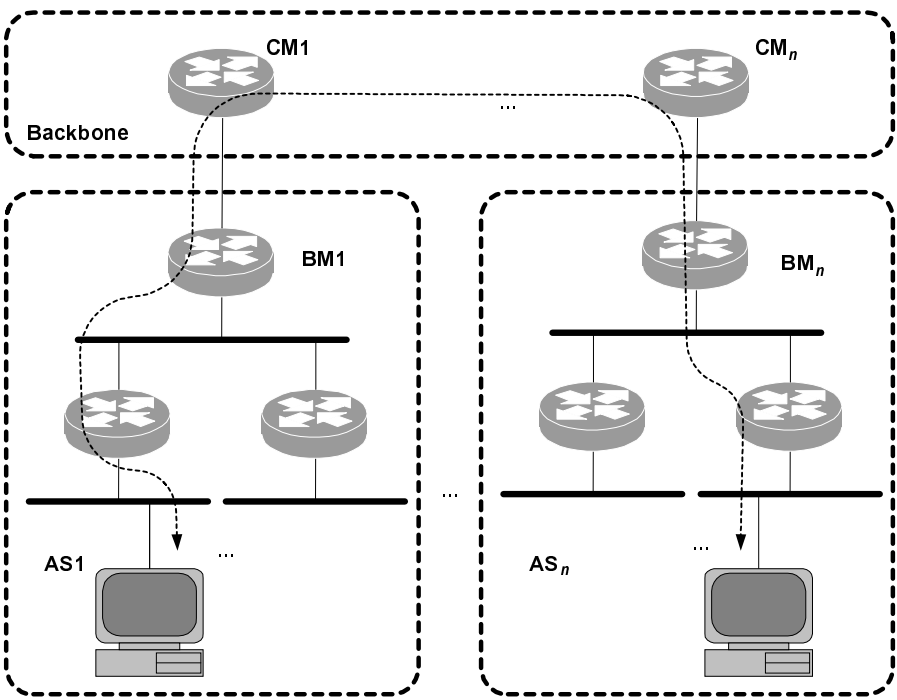


Рис. 4.19. Схема организации взаимодействия в автономных системах

В данном случае каждая из автономных систем AS_N может объединять в себе несколько локальных сетей. Взаимодействие между локальными сетями в пределах автономной системы производится в соответствии с правилами определения маршрута, принятыми в данной автономной системе. Информационное взаимодействие между компонентами, принадлежащими различным автономным системам, выполняется через специальную область,

предназначенную для интеграции системы в целом. В соответствии с терминологией, принятой в данной концепции, такая область называется магистральной (Backbone). Функция, выполняемая магистральной областью, аналогична функции, возлагаемой на центральные маршрутизаторы в системах Core Architecture. Автономная система может иметь несколько физических каналов для обеспечения информационного взаимодействия с компонентами других автономных систем через Backbone.

Процесс определения маршрута в современных информационных сетях, таким образом, распадается на два процесса:

- определение маршрута внутри автономной системы;
- определение маршрута между автономными системами.

Выше было отмечено, что первая задача решается индивидуальными средствами каждой автономной системы. Как правило, этими средствами являются один или несколько протоколов маршрутизации класса IGP (Interior Gateway Protocol). Эти средства должны обеспечивать управление маршрутами и выполнение информационного обмена в пределах автономной системы и, кроме того, обеспечивать информационный обмен с компонентами других автономных систем. Для решения последней задачи каждая автономная система информирует остальные системы через область Backbone Area о состоянии существующих в этой системе маршрутов. Для передачи этой информации внутри автономной системы организуются один или несколько специальных коммуникационных узлов. Задача этих узлов заключается в обеспечении двухстороннего информационного взаимодействия данной автономной системы с остальным миром. Функции коммуникационных узлов в реальных системах выполняют специальные пограничные маршрутизаторы, называемые exterior/edge router (внешний пограничный маршрутизатор). Пограничные маршрутизаторы всех автономных систем совместно с маршрутизаторами центральной области участвуют в выполнении специальных алгоритмов маршрутизации, которые используются для определения маршрута передачи данных между автономными системами.

Примечание

Алгоритмы маршрутизации, которые используются для определения маршрута за пределами автономной системы, относятся к классу exterior gateway protocol — EGP.

Алгоритмы маршрутизации, которые используются для определения маршрута внутри автономной системы, относятся к классу interior gateway protocol — IGP.

В настоящее время в сети Интернет используются два протокола маршрутизации класса Exterior Gateway Protocol:

- протокол внешнего шлюза EGP (Exterior Gateway Protocol) (в данном случае наименование протокола в точности совпадает с обозначением его класса);

- протокол пограничного шлюза BGP (Border Gateway Protocol).

Основное отличие протоколов внешней маршрутизации от протоколов IGP заключается в стратегии определения качества маршрута. Протоколы внешней маршрутизации принципиально не могут использовать схемы определения метрики маршрута, применяемые в алгоритмах внутренней маршрутизации. Основная причина заключается не столько в том, что различные автономные системы зачастую представляют несовместимые метрики для проходящих через них участков маршрута, а и в том, что эти метрики жестко привязаны к конкретной точке внутри каждой автономной системы. Следовательно, при наличии в автономной системе нескольких точек подключения задача построения оптимального внешнего маршрута существенно усложняется. Эту сложную задачу можно решать самыми различными способами. Протокол EGP, в частности, накладывает значительные ограничения на структуру сети, в которой он используется, и к тому же решает задачу в упрощенном виде — определяет только факт досягаемости некоторой сети.

Протокол EGP был первым протоколом внешней маршрутизации, использованным в сети Интернет, и в силу этого он, естественно, недостаточно совершенен и имеет массу недостатков. Однако многие принципы информационного взаимодействия, использованные в этом протоколе, с успехом применяются и в современных протоколах маршрутизации.

Взаимодействие маршрутизаторов протокола EGP, назначение и типы сообщений

Протокол EGP предполагает решение нескольких задач:

- установление партнерских отношений между маршрутизаторами (Neighbor Acquisition);
- определение текущего статуса партнера (Neighbor Reachability);
- определение досягаемости сетей (Network Reachability).

В процессе своего функционирования маршрутизаторы EGP обмениваются специальными сообщениями, которые помогают им устанавливать и поддерживать партнерские отношения и обмениваться маршрутной информацией. Ниже приведены основные типы сообщений протокола EGP:

- сообщения Neighbor Acquisition Messages;
- сообщения Neighbor Reachability Messages;
- сообщения Poll Request Messages;
- сообщения Routing Update Messages.
- сообщения Error Responce/Indication

Сообщения установления партнерских отношений Neighbor Acquisition

Сообщения этого типа маршрутизатор EGP передает в том случае, когда собирается установить партнерские отношения с одним из соседних

маршрутизаторов. Структура сообщения Neighbor Acquisition приведена на рис. 4.20.

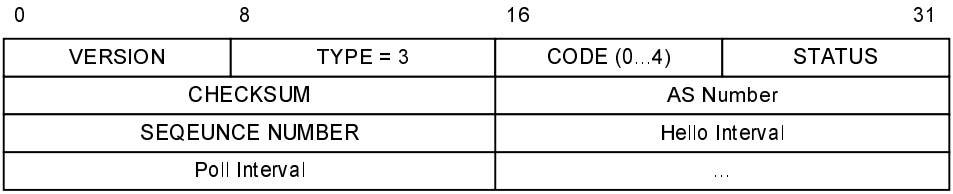


Рис. 4.20. Структура сообщения Neighbor Acquisition

В поле **VERSION** размещается номер версии протокола EGP, реализованной на маршрутизаторе — источнике сообщения.

Содержимое поля **TYPE** определяет тип сообщения EGP. Сообщениям Neighbor Acquisition соответствует значение 3.

Значение поля **CODE** определяет код конкретного сообщения Neighbor Acquisition. Соответствие между сообщениями и их кодами приведено в табл. 4.6.

Таблица 4.6. *Сообщения установления партнерских отношений EGP*

Код	Команда	Содержание
0	Request command	Запрос на установление партнерских отношений
1	Confirm response	Подтверждение установления партнерских отношений
2	Refuse response	Отказ установления партнерских отношений
3	Cease	Запрос на расторжение партнерских отношений
4	Cease-Ack response	Подтверждение расторжения партнерских отношений

В зависимости от выполнения или невыполнения условий обеспечения взаимодействия, в ответ на сообщение Request command маршрутизатор может получить подтверждение установления партнерских отношений Confirm response или отказ от их установления Refuse response. В том случае, если маршрутизатор по своей инициативе хочет расторгнуть уже установленные партнерские отношения, он должен передать сообщение Cease. Отношения не будут считаться расторгнутыми до получения сообщения о подтверждении их расторжения — Cease-Ack response.

Поле **STATUS** является общим для всех типов сообщений протокола EGP. При необходимости маршрутизатор EGP, являющийся источником сообще-

ния, может размещать в этом поле диагностическую информацию о своем текущем состоянии. В табл. 4.7 приведены возможные значения поля **STATUS** и соответствующие им диагностические сообщения.

Таблица 4.7. *Диагностические сообщения EGP*

Код	Сообщение	Содержание
0	Unspecified	Неопределенное
1	Active mode	Активный режим
2	Passive mode	Пассивный режим
3	Insufficient resources	Недостаточно ресурсов
4	Administratively prohibited	Запрещено
5	Going down	Выключено
6	Parameter problem	Несоответствие значений параметров
7	Protocol violation	Нарушение протокола

Содержание поля **AS Number** устанавливается в соответствии с номером автономной системы источника сообщения протокола EGP.

В поле **SEQEUNCE NUMBER** размещается текущий последовательный номер сообщения протокола EGP, сформированного данным источником. Последовательный номер присваивается передаваемому сообщению протокола EGP для того, чтобы приемник смог установить факт потери переданного сообщения и принять соответствующие меры для его восстановления.

В поле **Hello Interval** размещается значение временного интервала, отделяющего два последовательных сообщения Hello.

Примечание

Для поддержания партнерских отношений между маршрутизаторами протокола EGP достаточно, чтобы хотя бы один из них находился в активном (Active) состоянии и формировал сообщения Hello. Обычно оба маршрутизатора протокола EGP, установившие партнерские отношения, находятся в активном состоянии.

В поле **Poll Interval** размещается значение временного интервала, который определяет частоту передачи обновлений маршрутной информации. В поле **CHECKSUM** размещается 16-разрядная контрольная сумма сообщения протокола EGP. В начале вычислений ее значения поле **CHECKSUM** считается равным 0.

Сообщения состояния партнера Neighbor Reachability

Сообщениями этого типа маршрутизаторы протокола EGP обмениваются для поддержания партнерских отношений. Однако использование этих сообщений не является обязательным, поскольку для поддержания партнерских отношений и установления состояния партнера могут быть использованы также сообщения обновления маршрутной информации. Маршрутизатор протокола EGP может использовать содержимое поля **STATUS** для определения состояния соседа вместо того, чтобы периодически опрашивать его. Администратором может быть выбран один из способов поддержания партнерских отношений, обеспечивающий наиболее эффективное использование пропускной способности канала передачи данных. Структура сообщения Neighbor Reachability приведена на рис. 4.21.

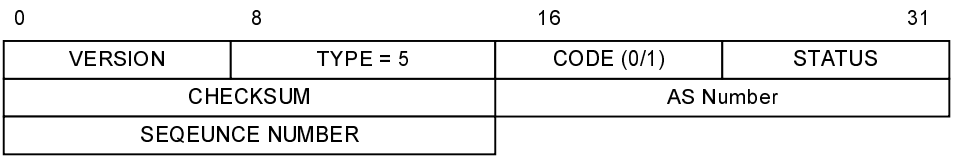


Рис. 4.21. Структура сообщения Neighbor Reachability

Содержимое поля **STATUS** сообщения Neighbor Reachability определяет состояние партнера следующим образом: 1 — соответствует включенному состоянию, 2 — выключенному. Значение 0 соответствует неопределенному состоянию.

Содержимое поля **CODE** определяет тип сообщения: значение 0 — соответствует запросу "Hello command", значение 1 — соответствует ответу "Я слышу Вас" — I-H-U (I-Hear-yoU) response.

Некоторые из ответов I-H-U response могут быть потеряны из-за ошибок, которые периодически возникают в каналах передачи данных. Для того чтобы уменьшить влияние этих ошибок на выполнение процедуры поддерживания партнерских отношений, маршрутизатор может использовать специальный алгоритм с запаздыванием. В этом случае сосед считается "мертвым", если он посылает менее N ответов I-H-U на каждые M переданных сообщений Hello.

Сообщения проверки состояния маршрута Poll Request

Сообщения этого типа маршрутизатор передает в том случае, когда хочет обновить содержимое маршрутной информации, полученной от партнера. В качестве параметра сообщения указывается адрес опорной сети **IP SOURCE NETWORK**, относительно которой должны быть рассчитаны метрики

маршрутов в ответном сообщении Routing Update. Структура сообщения Poll Request приведена на рис. 4.22.

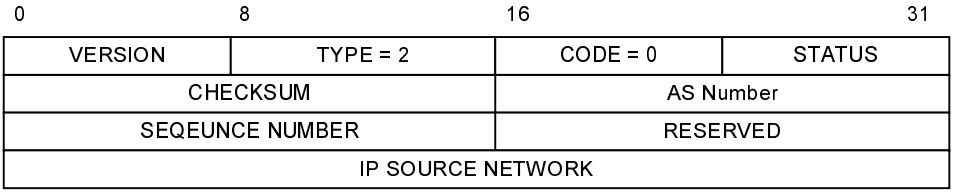


Рис. 4.22. Структура сообщения Poll Request

Сообщения обновления маршрутной информации Routing Update

Сообщения этого типа маршрутизатор передает в ответ на полученное сообщение Poll Request. В сообщении Routing Update содержится информация о внутренних маршрутах данной автономной системы. Маршруты, содержащиеся в сообщении, могут быть представлены маршрутизатору протокола EGP несколькими внутренними маршрутизаторами, расположенными в опорной сети, адрес которой указывается в поле **IP SOURCE NETWORK** сообщения **Pool Request**. Маршруты, представляемые каждым из этих маршрутизаторов, в сообщении Routing Update упорядочены по значению их дистанции. Структура сообщения Routing Update приведена на рис. 4.23.

В поле **STATUS** сообщений Routing Update содержится информация о состоянии маршрутизатора протокола EGP. В полях **#INT.GW** и **#EXT.GW** размещается количество внутренних и внешних шлюзов (маршрутизаторов), от которых были получены маршруты до сетей, представленных в данном сообщении, поскольку протокол EGP не обеспечивает возможность установить различие между двумя этими типами маршрутизаторов. В том случае, если необходимо точное знание типа представленного маршрута, целесообразно формировать отдельные сообщения Routing Update для каждого из этих типов шлюзов. Так же, как и в сообщении Poll Request, в данном сообщении в поле **IP SOURCE NETWORK** размещается адрес сети, по отношению к которой вычисляются метрики представляемых маршрутов.

Как уже было выше отмечено, описание метрики маршрута в протоколе EGP указывается относительно общей сети, которая называется Source Network. Это является существенным отличием данного протокола от протоколов класса IGP. В качестве примера использования Source Network рассмотрим сеть, представленную на рис. 4.24.

0	8	16	31
VERSION	TYPE = 1	CODE = 0	STATUS
CHECKSUM		AS Number	
SEQEUNCE NUMBER		# INT.GW	# EXT.GW
IP SOURCE NETWORK			
ROUTER 1 ADDRESS (HOST)			
# DISTANCES			
$DISTANCE_{11}$	# NET at D_{11}		
NET_1 at D_{11}			
...			
NET_i at D_{11}			
$DISTANCE_{1m}$	# NET at D_{1m}		
NET_1 at D_{1m}			
...			
NET_2 at D_{1m}			
ROUTER 2 ADDRESS (HOST)			
# DISTANCES			
...			
ROUTER n ADDRESS (HOST)			
...			
NET_j at D_{nm}			

Рис. 4.23. Структура сообщения Routing Update

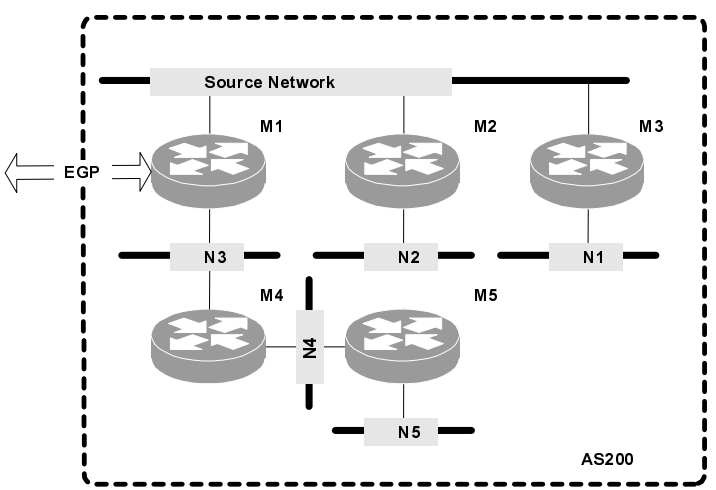


Рис. 4.24. Представление внутренних маршрутов протоколом EGP

В представленном примере автономная система AS200 использует маршрутизатор M1 для представления маршрутов к своим внутренним сетям другим автономным системам. Дистанции внутренних сетей в автономной системе AS200, полученные в соответствии с принципом формирования информации о маршрутах, принятым в протоколе EGP, представлены в табл. 4.8.

Таблица 4.8. *Дистанции внутренних сетей EGP*

Маршрутизатор	Сеть	Дистанция
M1	N3	1
M1	N5	2
M1	N4	3
M2	N2	1
M3	N1	1

Информация о маршрутах, передаваемых сообщением Routing Update (см. рис. 4.23), сгруппирована по двум признакам: по адресу маршрутизатора, представившего маршрут, и по размеру дистанции для данного маршрута. В поле **# DISTANCES** помещается информация о количестве групп маршрутов, проходящих через маршрутизатор, адрес которого указан в поле **ROUTER n ADDRESS.** Каждая из групп в данном случае образуется из сетей, имеющих одинаковый размер дистанции.

Описание каждой группы начинается с указания размера дистанции для данной группы **DISTANCE**$_{ni}$, вслед за которым указывается число сетей в данной группе **#NET at D**$_{ni}$. В последующих полях размещаются собственно адреса сетей.

Сообщение Error Response/Indication

Маршрутизатор протокола EGP использует сообщения этого типа для того, чтобы предупредить партнера о возникновении нештатной ситуации. Структура сообщения Error Response/Indication приведена на рис. 4.25.

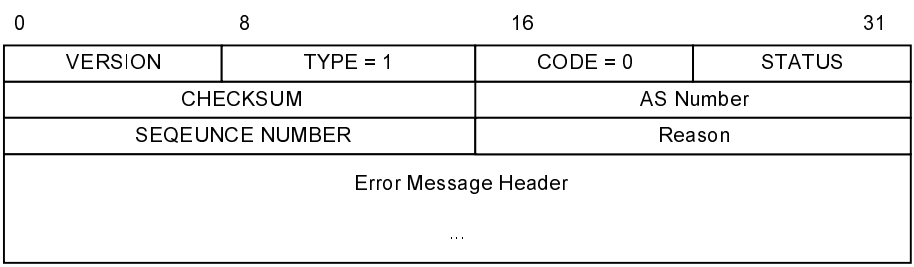


Рис. 4.25. Структура сообщения Error Response/Indication

В поле **Reason** в данном случае размещается код, определяющий тип нештатной ситуации. Соответствие между типом ситуаций и их кодами приведено в табл. 4.9.

Таблица 4.9. *Типы нештатных ситуаций протокола EGP*

Код поля Reason	Ситуация	Содержание
0	Undefined	Непредвиденная ситуация
1	Bad EGP header format	Неправильный формат заголовка EGP
2	Bad EGP data field format	Неправильный формат поля данных EGP
3	Reachability info unavailable	Информация о достижимости недоступна
4	Excessive polling rate	Превышение допустимой частоты обновлений
5	No response	Нет ответа

Ограничения и недостатки протокола EGP

Как уже было отмечено выше, протоколу EGP свойственен ряд существенных недостатков.

- Протокол маршрутизации EGP представляет только один путь до каждой сети. Это делает невозможным использование процедур динамического перераспределения нагрузки между параллельными каналами.
- Протокол маршрутизации EGP не поддерживает внеклассовые сети.
- Протокол маршрутизации EGP представляет метрики маршрутов, рассчитываемых относительно некоторой общей сети, а не относительно интерфейсов маршрутизатора.

Совокупность этих недостатков ставит под сомнение возможность успешного применения протокола EGP в современных сетях. Неоднократно предпринимаемые попытки доработать протокол EGP в соответствии с современными требованиями к протоколам данного типа (EGP2, EGP3), к сожалению, успехом не увенчались.

Протокол маршрутизации BGP

Протокол маршрутизации BGP (Border Gateway Protocol) является более современным протоколом внешней маршрутизации автономных систем, чем протокол EGP. Оба этих протокола построены по примерно одинаковой

схеме, однако протокол BGP имеет ряд существенных преимуществ по отношению к EGP.

При разработке этого протокола маршрутизации был использован опыт, полученный при создании и применении протокола EGP.

Основной задачей протокола BGP является распространение информации о достижимости сетей между автономными системами. Для решения этой задачи один или несколько маршрутизаторов автономной системы собирают информацию о достижимости внутренних сетей данной системы и накапливают ее в специальных базах внутренних сетей. Эти базы используются в качестве одного из источников для формирования базы маршрутов, для представления соседним маршрутизаторам — партнерам по протоколу BGP. В эту базу помещается также информация, полученная от маршрутизаторов-партнеров аналогичным образом.

Способы распространения маршрутной информации протокола BGP

Алгоритм протокола BGP относится к классу улучшенных алгоритмов маршрутизации Distance-Vector. В отличие от своего классического прообраза, такие алгоритмы маршрутизации не используют периодическую передачу информации о маршрутах. Вместо этого улучшенные алгоритмы маршрутизации класса Distance-Vector передают соседям только изменившуюся часть маршрутной таблицы. Для обеспечения актуальности маршрутной информации используется периодический опрос состояния соседних маршрутизаторов. В качестве примера рассмотрим сеть, представленную на рис. 4.26. Пограничные маршрутизаторы M11, M21 и M31 выполняют в данном случае трансляцию маршрутной информации своих автономных систем и ретрансляцию маршрутной информации, полученной от соседних автономных систем. Такие маршрутизаторы называют трансляторами (Speaker) BGP. Один раз после первоначального включения такой маршрутизатор передает соседям полную информацию обо всех внутренних сетях своей автономной системы. Все последующие изменения маршрутной информации транслятор BGP оформляет в виде последовательно пронумерованных обновлений исходной информации. Маршрутизатор M11 путем анализа ответов от маршрутизаторов M21 и M31 на свой периодический запрос может убедиться в том, что эти маршрутизаторы располагают последней версией переданной им информации о системе AS100. В том случае, если маршрутизатор M31 не получит своевременного ответа на свой запрос от маршрутизатора M21, он должен исключить внутренние сети AS200 из состава сетей, представляемых им в автономную систему AS100.

Еще одной важной особенностью протокола BGP является возможность построения маршрутов, проходящих через несколько автономных систем, а также возможность использования последовательности прохождения в качестве

критерия оптимальности маршрута. Благодаря этой особенности протокол BGP может использоваться в информационных сетях, имеющих произвольную структуру.

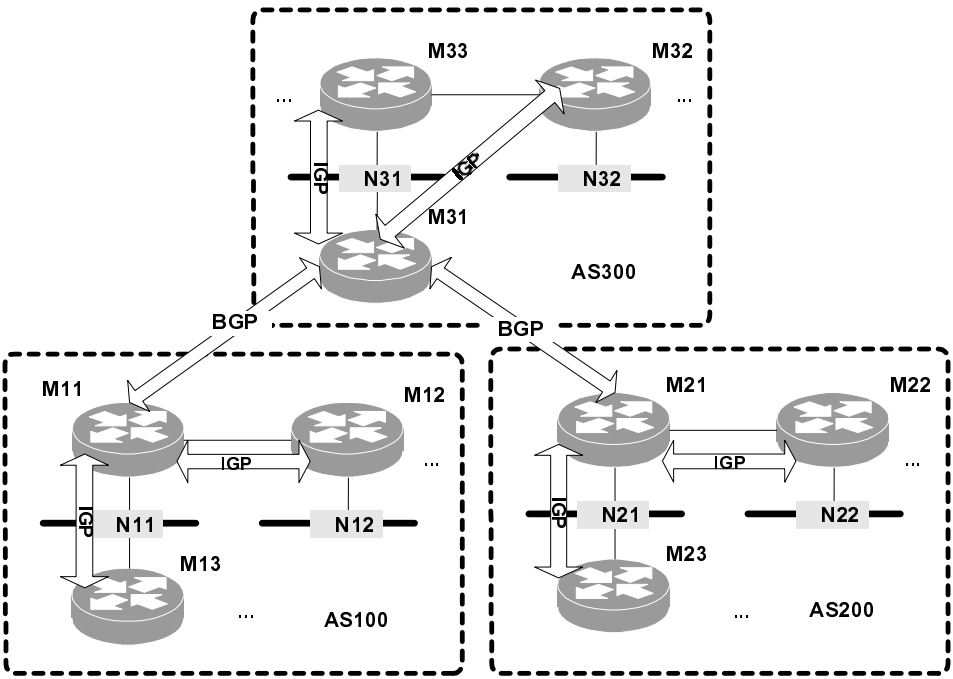


Рис. 4.26. Передача маршрутной информации с использованием протокола BGP

Примечание

Следует отметить, что предшественник протокола BGP, протокол EGP, мог успешно применяться только в сетях, имеющих радиальную архитектуру, поскольку не обладал необходимыми возможностями по обнаружению и устранению циклических маршрутов.

Задача обнаружения циклов легко решается протоколом BGP благодаря использованию принципиально иного подхода к оценке маршрутов. В дополнение к полученной от протокола IGP метрике, протокол BGP снабжает маршрут в представляемую сеть совокупностью собственных атрибутов. В противоположность используемой в алгоритмах протокола IGP метрике маршрута, которая оценивает его объективные технические характеристики, практически все атрибуты, используемые протоколом BGP, представляют субъективные оценки маршрута. Атрибуты протокола BGP могут быть совершенно по-разному трактоваться различными автономными системами, что обеспечивает возможность реализации в каждой из этих автономных

систем своей независимой политики определения маршрута — маршрутной политики.

Атрибуты маршрутов

Выше было отмечено, что маршрутизатор протокола BGP снабжает маршрут совокупностью специально определяемых параметров — атрибутов. Далее атрибутом протокола BGP мы будем называть характеристику маршрута, формируемую для обеспечения реализации политики маршрутизации. По характеру своего формирования и использования атрибуты маршрутов протокола BGP делятся на 4 группы:

- популярные обязательные (Well-known mandatory);
- популярные необязательные (Well-known discretionary);
- специфические передаваемые (Optional transitive);
- специфические непередаваемые (Optional non-transitive).

Популярные атрибуты должны интерпретироваться во всех реализациях протокола маршрутизации BGP для обеспечения взаимной совместимости этих реализаций. Кроме того, некоторые популярные атрибуты (Well-known mandatory) обязательно должны присутствовать в представляемом маршруте. Маршрутизатор протокола BGP должен передать все популярные атрибуты полученного маршрута при его представлении в другие автономные системы. При этом некоторые из данных атрибутов, в соответствии с алгоритмом их формирования, должны быть модифицированы.

В дополнение к популярным атрибутам маршрут протокола BGP может также содержать и специфические атрибуты. Интерпретация этих атрибутов не является обязательной для всех реализаций протокола BGP. Для обозначения того, что не все маршрутизаторы на маршруте были способны интерпретировать специфический передаваемый атрибут, используются специальные признаки в заголовке сообщения протокола BGP.

В случае, если маршрутизатор не может интерпретировать непередаваемый специфический атрибут принятого маршрута, он может тихо уничтожить такой атрибут и не транслировать его своим соседям при передаче этого маршрута.

Основной акцент при разработке протокола BGP был сделан на возможность построения и реализации с его помощью политик маршрутизации для автономной системы.

Атрибут ORIGIN

Атрибут ORIGIN (Первоисточник) относится к популярным обязательным атрибутам протокола BGP. Формирование этого атрибута производится пограничным маршрутизатором той автономной системы, где размещена представляемая сеть. Атрибут ORIGIN должен размещаться во всех сообще-

ниях BGP, содержащих информацию о представляемой сети. Назначение этого атрибута состоит в том, чтобы определить, каким образом информация о сети попала протоколу BGP. Каждому из возможных вариантов поступления этой информации соответствует одно из возможных значений, представленных в табл. 4.10.

Таблица 4.10. *Первоисточники маршрутной информации протокола BGP*

ORIGIN	Кодовое обозначение	Первоисточник маршрутной информации
0	IGP	Информация о достижимости сети сформирована внутри AS
1	EGP	Информация о достижимости сети представлена в AS протоколом EGP
2	Incomplete	Для получения информации о достижимости сети использован другой источник информации

Атрибут AS_PATH

Атрибут AS_PATH (путь через AS) также относится к популярным обязательным атрибутам протокола BGP и обычно представляет перечень номеров автономных систем пересекаемых при достижении представляемой сети. Этот перечень может состоять из нескольких сегментов одного из двух возможных типов:

- Type=1 — неупорядоченный набор номеров автономных систем;
- Type=2 — номера автономных систем упорядочены в соответствии с очередностью их прохождения по маршруту.

Когда пограничный маршрутизатор представляет какой-либо маршрут своему соседу из той же автономной системы, в которой он сам находится, значение этого атрибута не должно изменяться. Исключение представляет тот случай, когда соседу по автономной системе направляется самодельный маршрут. В такой ситуации пограничный маршрутизатор протокола BGP должен включить в сообщение специальный (пустой) атрибут AS_PATH. Во всех остальных случаях маршрутизатор протокола BGP должен добавить идентификатор собственной автономной системы слева к существующему перечню номеров маршрута. Причем, в зависимости от того, к какому типу относился последний сегмент, маршрутизатор либо дополняет последний упорядоченный сегмент (Type=2), либо создает новый (Type=1).

Атрибут NEXT_HOP

Атрибут NEXT_HOP (следующий узел) относится к популярным обязательным атрибутам протокола BGP. Атрибут NEXT_HOP определяет IP-адрес пограничного маршрутизатора, используемого в качестве узла Next Hop для

достижения представляемой сети. При формировании представления сети пограничный маршрутизатор протокола BGP может использовать адрес любого BGP-маршрутизатора из числа соседних с ним по автономной системе. При выборе внутреннего маршрутизатора учитывается степень его близости к представляемой сети.

При представлении маршрута, полученного из другой автономной системы, пограничный маршрутизатор, предающий сообщения протокола BGP должен придерживаться следующего правила: никогда не направлять маршрут с атрибутом NEXT_HOP в направлении узла с соответствующим IP-адресом.

Примечание

В данном случае реализуется принцип, подобный рассмотренному ранее для протоколов IGP принципу Split Horizon.

Атрибут MULTI_EXIT_DISC

Атрибут MULTI_EXIT_DISC (определитель выхода) относится к специфическим непередаваемым атрибутам протокола BGP и представляет собой целое неотрицательное число, задаваемое четырехбайтовым двоичным кодом. При наличии в данной автономной системе нескольких маршрутов для доступа во внешнюю автономную систему AS_N значение этого атрибута определяет степень предпочтения использования каждого из сравниваемых маршрутов (чем меньше значение MULTI_EXIT_DISC, тем выше приоритет маршрута). В соответствии со статусом непередаваемого атрибута, значение MULTI_EXIT_DISC не должно передаваться внешним партнерам по протоколу BGP при представлении полученного маршрута.

Атрибут LOCAL_PREF

Атрибут LOCAL_PREF (локальное предпочтение) относится к специфическим непередаваемым атрибутам протокола BGP. Так же, как и атрибут MULTI_EXIT_DISC, LOCAL_PREF представляется целым неотрицательным числом, задаваемым четырехбайтовым двоичным кодом. Назначение атрибута LOCAL_PREF состоит в определении степени предпочтения внешнего маршрута для данной автономной системы. Атрибут LOCAL_PREF включается во все сообщения о маршрутах, которыми обмениваются между собой маршрутизаторы протокола BGP внутри автономной системы. Сравнивая значения атрибутов маршрутов, представленных маршрутизаторами-партнерами, маршрутизатор протокола BGP определяет среди них один наиболее предпочтительный. Единая схема формирования и анализа значений атрибутов LOCAL_PREF позволяет таким образом реализовать единство внутренней политики маршрутизации данной автономной системы. Отличие атрибута LOCAL_PREF от ранее рассмотренного MULTI_EXIT_DISC состоит в том, что атрибут LOCAL_PREF позволяет

задавать различную степень предпочтения маршрутам одной внешней автономной системы.

Атрибут ATOMIC_AGGREGATE.

Атрибут ATOMIC_AGGREGATE (суммирование маршрутов) относится к числу популярных необязательных атрибутов BGP. Пограничный маршрутизатор-транслятор протокола BGP должен включить этот атрибут в передаваемое сообщение в том случае, если он произвел операцию суммирования маршрутов, то есть собрал несколько маршрутов в сети с перекрывающимся адресным пространством в один маршрут с наиболее коротким префиксом. Назначение этого атрибута, таким образом, состоит в том, чтобы указать на наличие внутренней структуры у представляемого маршрута. Однажды установленное для маршрута значение атрибута ATOMIC_AGGREGATE не должно изменяться при его последующих представлениях другими маршрутизаторами протокола BGP.

Атрибут AGGREGATOR

Атрибут AGGREGATOR (суммирующий узел) имеет длину 6 байтов и относится к числу специфических передаваемых атрибутов протокола BGP. Назначение этого атрибута состоит в указании маршрутизатора, который произвел операцию суммирования маршрутов. Два байта атрибута AGGREGATOR содержат номер автономной системы, где было произведено суммирование маршрутов. В последующих 4 байтах размещается собственно IP-адрес маршрутизатора, который эту операцию выполнил.

Понятие и способы реализации политики маршрутизации

Политикой маршрутизации (ПМ) далее мы будем называть иерархическую совокупность критериев, используемых для выбора маршрута между автономными системами. Политика определяет степень важности для каждого из критериев. Следует учитывать, что критерии, используемые протоколом маршрутизации BGP, предназначены для сравнения не столько технических, сколько административных характеристик маршрута. Благодаря использованию ПМ пограничный маршрутизатор автономной системы может строить внешние маршруты, имеющие заданные характеристики на участках, находящихся за пределами этой автономной системы.

Внешний характер политики маршрутизации, реализуемой с использованием протокола BGP, заключается, однако, не в том, что маршрутизатор протокола BGP навязывает свою волю по определению маршрута в некоторую сеть N пограничным маршрутизаторам других автономных систем. Очевидно, что такой способ построения маршрута напрямую противоречил бы принятой в Интернете концепции "Hop-by-Hop". Если маршрутизатор M в силу особенности своей ПМ не желает представлять проходящий через его

автономную систему маршрут до некоторой сети N, то никакие внешние силы не в состоянии заставить его это сделать. Задача протокола внешней маршрутизации BGP состоит в том, чтобы из всех возможных маршрутов в сеть N выбрать тот, который наиболее полно соответствует предъявляемым к нему требованиям, сформулированным в ПМ. Многочисленность и разнообразие атрибутов маршрута протокола BGP позволяют достаточно точно определить, насколько полно выполняются эти требования на всем протяжении каждого из рассматриваемых маршрутов.

Процедура определения оптимального маршрута

Воплощением политики маршрутизации, реализуемой маршрутизаторами протокола BGP автономной системы, является процедура определения оптимального маршрута. Назначение этой процедуры состоит в том, чтобы из маршрутов, полученных от маршрутизаторов-партнеров по протоколу BGP и хранящихся во входной информационной базе Adj-RIB-In (Routing Information Base-Input — база для размещения входной информации маршрутизации), выбрать только те, которые соответствуют текущей ПМ. Выбранные маршруты используются для формирования таблицы маршрутизации и для представления другим маршрутизаторам протокола BGP. Для выбора представляемых маршрутов также могут применяться правила, образующие специальную политику их представления.

Примечание

Политика выбора применяется к маршруту по индивидуальному принципу, при ее реализации никоим образом не учитываются, например, такие факторы, как наличие или отсутствие конкурирующих маршрутов.

Процедура определения оптимального маршрута обеспечивает выполнение трех основных функций:

- выбор маршрутов для их последующего представления маршрутизаторам-трансляторам протокола BGP, размещенным в этой же автономной системе;
- выбор маршрутов для их последующего представления маршрутизаторам-трансляторам протокола BGP, размещенным во внешних автономных системах;
- суммирование маршрутов и уменьшение объемов маршрутной информации.

Описываемая процедура состоит из трех последовательных этапов. На первом этапе вычисляются степени предпочтения для каждого из маршрутов, полученных от маршрутизаторов-партнеров по протоколу BGP. На основании рассчитанных значений на втором этапе производится выбор маршрутов, имеющих наибольшие значения предпочтений, для размещения последних в локальной базе маршрутов. На последнем этапе выполняется

выбор маршрутов для представления их партнерам по протоколу BGP и комплекс операций по уменьшению объемов маршрутной информации.

При определении степени предпочтения того или иного маршрута может быть использована следующая последовательность рассмотрения его атрибутов:

- маршрут с максимальным значением атрибута LOCAL_PREF;
- маршрут с наименьшим значением атрибута AS_PATH;
- маршрут с наименьшим значением атрибута ORIGIN

и т. д.

При прочих равных условиях предпочтение в конечном итоге будет отдано маршруту, представленному соседом, имеющим наименьший IP-адрес.

Особенности протокола BGP

Протокол BGP может быть использован для определения различных типов маршрутов:

- Inter-autonomous system routing — маршрут, соединяющий данную автономную систему с одной или несколькими другими автономными системами;
- Intra-autonomous system routing — маршрут внутри автономной системы;
- Pass-through autonomous system — маршрут, проходящий через автономную систему, не участвующую (может быть) в обмене маршрутной информацией по протоколу BGP.

Для обеспечения информационного обмена между маршрутизаторами протокола BGP используются специальные сообщения, передаваемые с использованием транспортного протокола TCP.

Так же, как и некоторые другие протоколы маршрутизации, протокол BGP предполагает установление партнерских отношений между маршрутизаторами, однако, в отличие от многих других протоколов маршрутизации, маршрутизаторы протокола BGP не пытаются устанавливать партнерские отношения с незнакомыми соседями, рассылая по всем направлениям сообщения типа "Здравствуйте все!". Маршрутизаторы протокола BGP могут установить партнерские отношения только с теми из своих соседей, которые были явно указаны администратором.

В ряде случаев может оказаться целесообразным размещение нескольких пограничных маршрутизаторов в пределах одной автономной системы. Чаще всего это случается, когда автономная система связана с несколькими другими системами одновременно. Для обмена маршрутной информацией между маршрутизаторами, принадлежащими одной автономной системе, используется внутренний протокол BGP (Internal BGP — IBGP). Чтобы отличать от этого варианта протокола собственно протокол BGP в его класси-

ческом понимании, для его обозначения используют аббревиатуру EBGP (External BGP).

Реализация протокола BGP в варианте IBGP имеет существенные особенности, заключающиеся в дополнительных правилах распространения маршрутной информации внутри автономной системы. Рассмотрим вариант построения сети, представленный на рис. 4.27.

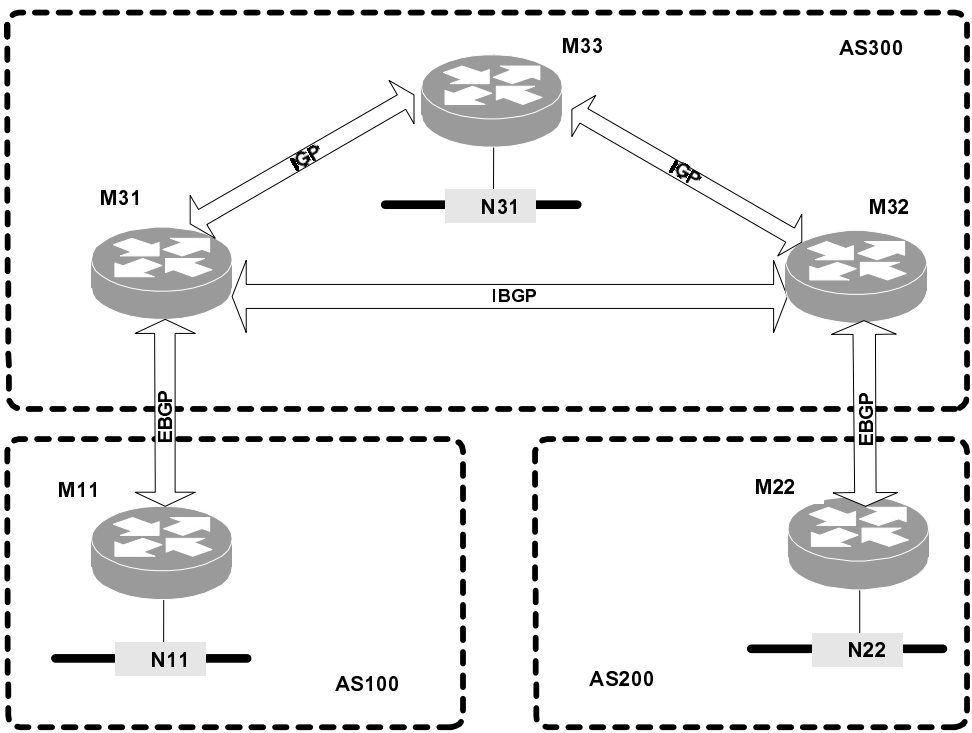


Рис. 4.27. Синхронизация представления маршрутной информации протокола BGP

В рассмотренном варианте построения сети используются внешний EBGP и внутренний IBGP протоколы BGP. В частности, сеть N11 представляется маршрутизатором M11 автономной системы AS100 с использованием протокола EBGP. Обмен маршрутной информацией между маршрутизаторами M31 и M32, принадлежащими к одной автономной системе AS300, производится по протоколу IBGP. В этом случае может оказаться так, что информация о достижимости сети N11 будет представлена в автономную систему AS200 маршрутизатором M32 раньше, чем появится соответствующий внутренний маршрут от одного из протоколов IGP. Это неминуемо приведет к потере пакетов, направленных по несуществующему маршруту из автономной системы AS200 в AS100. Для предотвращения подобных ситуаций

при совместном использовании внешнего и внутреннего вариантов протокола BGP используется следующее правило синхронизации: маршрут, полученный по протоколу IBGP, не может быть представлен во внешнюю систему раньше, чем представляющий маршрутизатор примет информацию о достижимости этой сети от протокола IGP. Описанная ситуация не возникает в том случае, если для передачи маршрутной информации внутри автономной системы используется только протокол BGP.

В последней (четвертой) версии протокола реализован набор механизмов, позволяющих осуществлять маршрутизацию для внеклассовых IP сетей, выполнять суммирование маршрутов с использованием принципов CIDR (Classless Inter-Domain Routing). Протокол BGP для определения маршрута использует префикс сети, который состоит из двух компонентов:

- длина маски сети — 1 байт;
- собственно номер сети — 14 байтов.

Типы и форматы сообщений протокола BGP

Для обеспечения информационного взаимодействия между маршрутизаторами протокола BGP передаются следующие типы сообщений:

- OPEN — сообщение о начале сеанса BGP;
- KEEP ALIVE — сообщение проверки состояния соседа;
- UPDATE — сообщение об изменении содержания таблицы маршрутов автономной системы;
- NOTIFICATION — сообщение о возникновении аварийной ситуации.

Маршрутизатор протокола BGP может обслуживать несколько различных баз маршрутов — Routing Information Bases (RIB), а именно: Adj-RIBs-In, Loc-RIB и Adj-RIBs-Out. В базе Adj-RIBs-In содержатся маршруты, полученные от соседних внешних маршрутизаторов. В базе Loc-RIB маршрутизатор протокола BGP размещает маршруты, предназначенные для внутреннего использования. В базе Adj-RIBs-Out размещаются маршруты, предназначенные для представления соседним маршрутизаторам протокола BGP.

Примечание

Маршрутизатор протокола BGP может при необходимости дезавуировать представленный ранее маршрут. Для этого может быть использовано несколько различных способов — например, с использованием поля **WITHDRAWN ROUTES** в сообщении UPDATE или путем разрывания сессии.

Формат сообщения протокола BGP

Каждое сообщение протокола BGP состоит из заголовка и последующих специфических полей. Структура заголовка сообщения протокола BGP приведена на рис. 4.28.

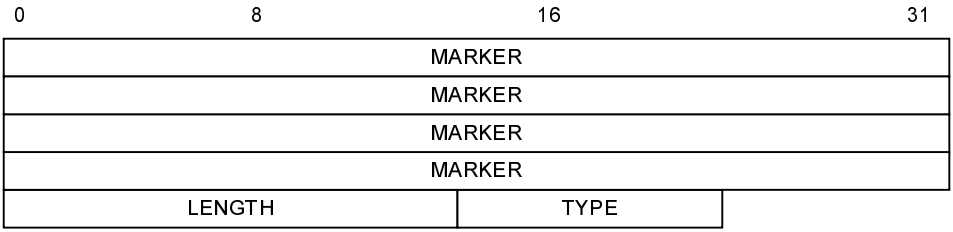


Рис. 4.28. Структура заголовка сообщения протокола BGP

В поле **LENGTH** помещается размер сообщения (вместе с заголовком), выраженный в байтах. Значение поля **TYPE** определяет тип сообщения протокола BGP. Возможные типы сообщений и их назначение приведены в табл. 4.11.

Таблица 4.11. *Типы и назначение сообщений протокола BGP*

TYPE	Тип сообщения	Назначение сообщения
1	OPEN	Сообщение о начале сеанса
2	UPDATE	Сообщение проверки состояния соседа
3	NOTIFICATION	Сообщение об изменении содержания Loc-RIB
4	KEEPALIVE	Сообщение о возникновении аварийной ситуации

В поле **MARKER** может быть помещена информация, необходимая для выполнения процедуры аутентификации партнера. Если установление подлинности абонента не требуется, маркер формируется значениями — все "1".

Сообщение OPEN

Сообщение OPEN является первым сообщением, которое передает маршрутизатор протокола BGP своему партнеру после установления с ним соединения по протоколу TCP. Структура сообщения OPEN приведена на рис. 4.29.

В поле **VERSION** размещается информация о номере версии протокола BGP. Как уже было отмечено, в настоящий момент используется четвертая версия протокола BGP. В поле **My Autonomous System** размещается номер автономной системы, в которой находится источник данного сообщения. Значение поля **Hold Time** определяет величину интервала времени, в течение которого отправитель будет ожидать получения ответного сообщения в подтверждение установления сессии протокола BGP. В поле **BGP Identifier** размещается идентификатор отправителя сообщения — в качестве этого идентификатора может быть использован IP-адрес одного из интерфейсов

маршрутизатора. В поле **Optional Parameters** (необязательные параметры), в частности, может быть размещена информация, которая используется для выполнения процедуры аутентификации партнера.

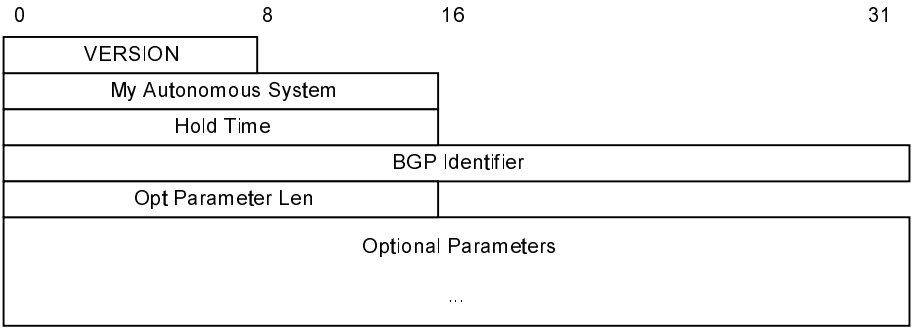


Рис. 4.29. Структура сообщения OPEN протокола BGP

Сообщение UPDATE

Сообщение UPDATE используется для представления маршрутов маршрутизаторам-партнерам протокола BGP. Это сообщение также может быть использовано для отзыва представленных ранее маршрутов, которые перестали существовать. Сообщение UPDATE состоит из стандартного заголовка сообщения протокола BGP и пяти последующих фрагментов:

- Unfeasible Routes Length — количество удаляемых маршрутов (2 байта);
- Withdrawn Routes — удаляемые маршруты (переменная длина);
- Total Path Attribute Length — длина поля атрибутов (2 байта);
- Path Attributes — атрибуты маршрута (переменная длина);
- Network Layer Reachability Information — информация о представляемой сети (переменная длина).

Фрагменты Unfeasible Routes Length и Withdrawn Routes определяют уничтожаемые маршруты. Фрагменты Withdrawn Routes содержат описания отзываемых маршрутов. Если данное сообщение UPDATE не содержит уничтожаемых маршрутов, фрагмент Unfeasible Routes Length должен иметь значение 0. Фрагмент Total Path Attribute Length содержит выраженную в байтах длину фрагмента Path Attributes. Атрибут каждого маршрута представляет собой совокупность трех значений: <attribute Type, attribute length, attribute value>. Содержимое этих полей используется для задания таких атрибутов маршрута — AS_PATH, NEXT_HOP и т. д., подробно описанных в предыдущих разделах настоящей главы. Содержание фрагмента Network Layer Reachability Information определяет представляемую сеть. Для представления используется пара значений <length, prefix>, где значение length

представляет собой выраженную в битах длину префикса сети, а в prefix — собственно префикс, т. е. биты сетевого адреса.

Сообщение KEEPALIVE

Маршрутизаторы протокола BGP периодически обмениваются сообщениями KEEPALIVE для того, чтобы подтвердить свою работоспособность и, следовательно, актуальность маршрутной информации, представленной своим партнерам. Период формирования сообщений KEEPALIVE обычно составляет третью часть значения интервала времени Hold Time — максимального интервала времени, в течение которого маршрутизатор протокола BGP, не передающий сообщения KEEPALIVE, может считаться работоспособным. Если значение Hold Time установлено равным 0, сообщения KEEPALIVE вообще не должны формироваться. Собственно сообщение KEEPALIVE состоит из заголовка протокола BGP и имеет длину 19 байтов.

Сообщение NOTIFICATION

Сообщение NOTIFICATION используется для указания на возникновение аварийной ситуации при выполнении процедур протокола BGP. Структура сообщения NOTIFICATION приведена на рис. 4.30.

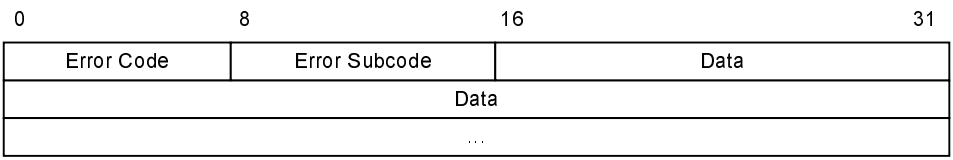


Рис. 4.30. Структура сообщения NOTIFICATION

В поле **Error Code** (Код ошибки) размещается кодировка, определяющая тип диагностического сообщения NOTIFICATION. Возможные типы диагностических сообщений и их назначение приведены в табл. 4.12.

Таблица 4.12. *Типы и назначение сообщений NOTIFICATION*

Error Code	Тип сообщения	Диагностическое сообщение
1	Header Error	Неверный формат сообщения или соединение протокола BGP не синхронизировано.
2	OPEN Error	Невозможно установить соединение протокола BGP вследствие технических или административных ограничений
3	UPDATE Error	Ошибочный атрибут маршрута принятого сообщения или возникновение маршрутной петли

Таблица 4.12 (окончание)

Error Code	Тип сообщения	Диагностическое сообщение
4	Hold Timer Expired	Истекло установленное время ожидания регулярного сообщения NOTIFICATION
5	Finite State Machine Error	Внутренняя ошибка протокола BGP
6	Cease	Запрос расторжения партнерских отношений

Пример практического использования алгоритма протокола BGP

В качестве примера практического использования протокола BGP рассмотрим сеть, представленную на рис. 4.31.

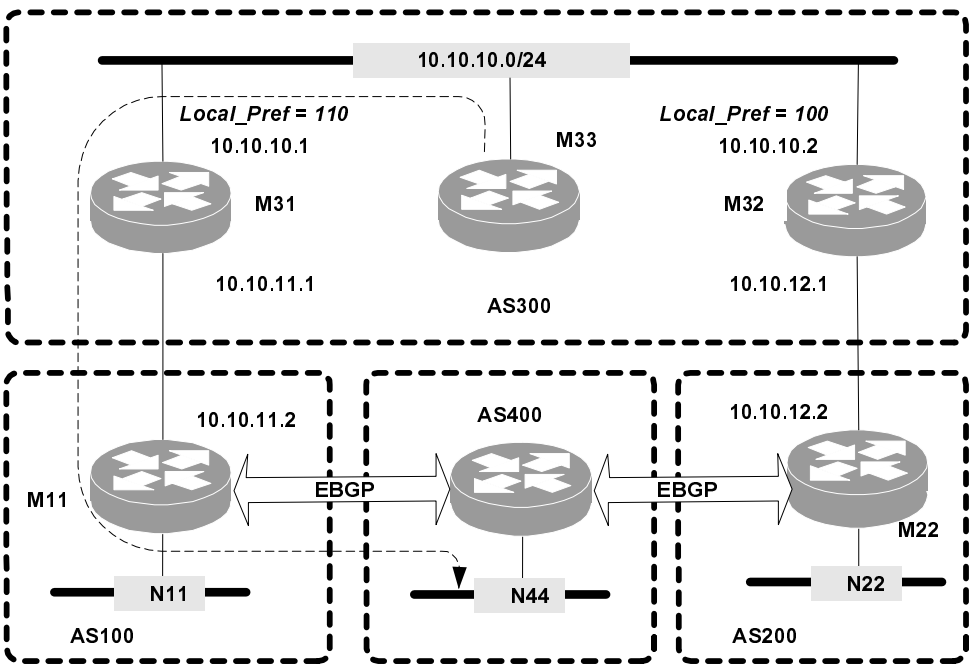


Рис. 4.31. Пример практического использования протокола BGP

В представленном варианте построения сети маршрутизаторы M31 и M32 представляют в автономную систему AS300 альтернативные маршруты в сеть N44. Независимо от соотношения технических характеристик внешних

каналов AS300, предпочтение в данном случае будет отдано маршруту, представленному маршрутизатором M31, поскольку ему соответствует большее значение атрибута Local_Preference, установленное администратором системы. Ниже представлены фрагменты конфигурационных файлов маршрутизаторов M31 и M32.

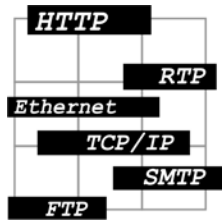
Фрагмент конфигурационного файла маршрутизатора M31

```
...
router bgp 300
neighbor 10.10.10.2 remote-as 300
neighbor 10.10.11.2 remote-as 100
bgp default local- preference 110
...
```

Фрагмент конфигурационного файла маршрутизатора M32

```
...
router bgp 300
neighbor 10.10.10.1 remote-as 300
neighbor 10.10.12.2 remote-as 200
...
```

Глава 5



Транспортные протоколы Интернета

Протоколы транспортного уровня предназначены для обеспечения непосредственного информационного обмена между двумя пользовательскими процессами (приложениями). Существует две разновидности протоколов транспортного уровня — сегментирующие протоколы и протоколы доставки дейтаграмм.

Сегментирующие протоколы транспортного уровня разбивают исходное сообщение на блоки данных транспортного уровня — сегменты. Основной функцией таких протоколов транспортного уровня является обеспечение доставки этих сегментов до объекта назначения и восстановление сообщения. Для выполнения этой задачи протокол транспортного уровня устанавливает и разрывает соединение между процессами пользователя, управляет скоростью передачи сегментов и обеспечивает достоверность передачи и восстановления сообщения специальными контрольными процедурами. Функционирование протокола транспортного уровня состоит из трех фаз:

- фаза установления соединения;
- фаза передачи сегментов;
- фаза разрыва соединения.

Протоколы доставки дейтаграмм не сегментируют сообщение, а отправляют его одним блоком, который называется "дейтаграмма". Очевидно, что в данном случае процедуры установления и разрыва соединения, управления потоком не требуются. Протоколы доставки дейтаграмм просты для реализации, однако не могут быть использованы для гарантированной и достоверной доставки сообщений.

Поскольку транспортное соединение может использоваться несколькими приложениями одновременно, должен существовать механизм, обеспечивающий идентификацию блоков данных транспортного уровня, сформированных различными приложениями. В семействе протоколов транспортного уровня сети Интернет для этой цели используются идентификаторы номеров виртуальных портов получателя и источника данных. Каждому приложению ставится в соответствие номер порта, который должен быть уникальным для данного объекта сетевого уровня. Таким образом, для

организации информационного взаимодействия на транспортном уровне должны быть указаны сетевой адрес абонента и номер порта приложения. В данном случае порт является виртуальным интерфейсом транспортного уровня. Взаимодействие процессов пользователя с протоколом транспортного уровня через порт может производиться по двум основным схемам:

- синхронизация процесса;
- буферизация поступающих данных.

При использовании первой схемы поступление данных от внешней системы в порт вызывает прерывание выполнения исполняемого процесса и немедленную обработку принятых данных. Возможность применения второй схемы обеспечивается использованием буферов промежуточного хранения принимаемых данных. При этом приложение взаимодействует с портом транспортного протокола в асинхронном режиме.

В настоящее время в сети Интернет используются два протокола транспортного уровня:

- UDP (User Datagram Protocol) — протокол дейтаграмм пользователя;
- TCP (Transmission Control Protocol) — протокол управления передачей.

Протокол UDP относится к транспортным протоколам доставки дейтаграмм и применяется в том случае, если к процессу доставки не предъявляется повышенных требований в части обеспечения надежности, гарантированности доставки и управления темпом информационного обмена.

Протокол TCP является основным транспортным протоколом сети Интернет и обеспечивает возможность управления информационным потоком и гарантированную доставку блоков данных приложений пользователя.

Особое место в сети Интернет занимает протокол RTP (Real-Time Transport Protocol). Этот транспортный протокол предназначен для обеспечения информационного взаимодействия приложений, требующих от процесса доставки обязательного выполнения совокупности установленных временных ограничений. Потребность в транспортном протоколе такого рода была вызвана активным использованием вычислительных сетей для передачи голосового и видео трафика.

Протокол UDP

Протокол UDP (User Datagram Protocol) предназначен для обмена дейтаграммами между приложениями через сеть Интернет. В качестве протокола нижнего уровня протокол UDP использует IP.

Протокол UDP предоставляет прикладным программам возможность обмениваться данными, используя для этого минимальное количество управляющих параметров. Как уже было отмечено, протокол UDP не обеспечивает

достоверность при доставке данных, защиту от дублирования и потери данных при их передаче. Достоинствами протокола UDP являются простота его реализации и экономичность. За исключением номеров портов отправителя и получателя данных, протокол UDP практически ничего не добавляет к исходной IP-дейтаграмме, полученной от протокола сетевого уровня.

Блоки данных, передаваемые протоколом UDP, называются *дейтаграммами*. Дейтаграммы имеют переменную длину и состоят из заголовка сообщения UDP header и поля полезной нагрузки UDP Data. На рис. 5.1 приведена структура блока данных протокола UDP.

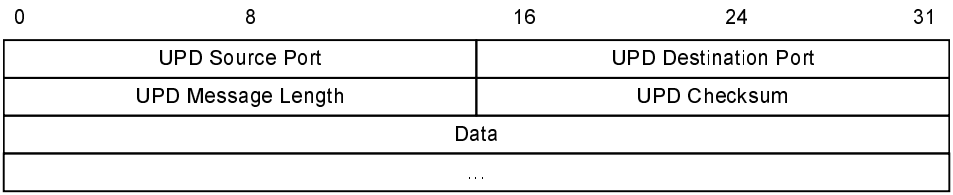


Рис. 5.1. Структура блока данных протокола UDP

Поле **UDP Source Port** (16 бит). Порт отправителя. В этом поле может быть помещен номер порта, с которого был отправлен пакет, в тех случаях, когда это имеет значение (например, когда отправитель ожидает ответа). Если это поле не используется, оно должно быть заполнено нулями. В табл. 5.1 приведен перечень портов, используемых наиболее популярными приложениями.

Таблица 5.1. *Перечень портов популярных приложений Интернета*

Порт	Обозначение	Приложение
25	smtp	Simple Mail Transfer Protocol
69	tftp	Trivial File Transfer Protocol
80	www-http	World Wide Web HTTP Protocol
79	finger	Finger Protocol
65	tacacs-ds	TACACS-Database Service Protocol
21	ftp	File Transfer Protocol
23	telnet	Telnet Protocol
110	pop3	Post Office Protocol — Version 3
123	ntp	Network Time Protocol
143	imap2	Interim Mail Access Protocol v2

Таблица 5.1 *(окончание)*

Порт	Обозначение	Приложение
161	snmp	Simple Network Management Protocol
179	bgp	Border Gateway Protocol
220	imap3	Interactive Mail Access Protocol v3

Поле **UDP Destination Port** (16 бит). Порт назначения — это порт приложения, на который должна быть доставлена данная дейтаграмма.

В поле **UDP Message Length** (16 бит) помещается значение, соответствующее длине данной дейтаграммы. Длина дейтаграммы в данном случае вычисляется в байтах, включая заголовок и данные. Минимальное значение **UDP Message Length** равно 8.

В поле **UDP Checksum** (16 бит) помещается контрольная сумма дейтаграммы протокола UDP. Контрольная сумма дейтаграммы протокола UDP представляет собой побитное дополнение 16-битной суммы 16-битных слов. При вычислении контрольной суммы используются следующие поля дейтаграммы:

- данные пакета — полезная нагрузка дейтаграммы;
- заголовок UDP-пакета — номера портов источника и приемника;
- псевдо-заголовок — IP-адрес источника и приемника.

Примечание

Псевдо-заголовок формально не является частью дейтаграммы протокола UDP. В данном случае его назначение состоит в том, чтобы исключить возможность ошибки при направлении дейтаграммы с сетевого уровня на транспортный уровень.

Протокол UDP полезен в таких ситуациях, когда мощные механизмы обеспечения надежности протокола TCP не требуются или будут только помехой для решения определенного рода задач, например, для передачи потокового трафика.

Преимущество протокола UDP состоит в том, что он практически не требует предварительных установок и согласования параметров для организации информационного обмена между приложениями. Этот протокол используется такими приложениями, как протоколы TFTP (Trivial File Transfer Protocol, простой протокол передачи файлов), SNMP (Simple Network Management Protocol) а также при построении систем аутентификации.

Протокол TCP

Существует достаточно много причин, которые мешают передаваемому по сети пакету успешно достигать узла назначения. В частности, такими причинами могут быть ошибки протоколов маршрутизации, нарушение функционирования каналов передачи данных, перегрузки в узлах сети. Эти причины приводят к тому, что часть информационных блоков передаваемого сообщения теряется, а другие блоки передаются одновременно по нескольким маршрутам. При передаче по сети информационные блоки могут многократно дублироваться и поступать в пункт назначения не в том порядке, в котором они были переданы. Таким образом, если не будут использованы специальные методы для обеспечения гарантированной доставки, принятое сообщение может существенным образом отличаться от того сообщения, которое было передано отправителем.

Задача восстановления исходного вида переданного сообщения может быть возложена на протоколы верхних уровней, например, прикладного уровня. Однако ввиду универсальности самой задачи обеспечения надежного информационного обмена целесообразно решать ее именно на транспортном уровне.

Способы обеспечения надежного информационного обмена

Надежный информационный обмен должен обеспечивать:

- потоковый режим информационного обмена;
- организацию виртуальных соединений;
- согласование скорости информационного потока и пропускной способности канала передачи данных;
- возможность одновременного приема и передачи информации.

Все эти требования могут быть удовлетворены транспортным протоколом, функционирующим по принципу установления соединения между взаимодействующими сторонами.

Надежное транспортное соединение позволяет обеспечить такой режим информационного взаимодействия, когда приемник получает абсолютно ту же последовательность байтов, которая была передана отправителем. В этом случае считается, что транспортное соединение обеспечивает потоковый обмен.

Надежный информационный обмен на транспортном уровне можно представить в виде виртуального логического соединения. На начальной стадии взаимодействия одна из сторон инициирует установление соединения, используя при этом по мере необходимости процедуры аутентификации.

В том случае, если все необходимые условия для установления соединения выполнены, обе стороны считают, что виртуальное соединение установлено, и могут переходить к фазе непосредственного обмена данными. В процессе информационного обмена через установленное соединение обе стороны контролируют качество этого обмена и при возникновении неразрешимых проблем могут инициировать процесс разрыва соединения и формировать соответствующие сообщения для протоколов верхних уровней.

Использование буферов позволяет согласовать скорость информационного обмена в канале передачи данных со значением скорости формирования данных приложением пользователя. В том случае, если скорость передачи данных в канале намного превышает скорость формирования данных приложением, целесообразно накапливать передаваемые данные в буфере и передавать их затем в виде блока. Такой способ позволит повысить эффективность использования канала передачи данных, поскольку обеспечивает существенное снижение накладных расходов.

Для обеспечения требования доставки трафика, чувствительного к временным задержкам, в дополнение к буферу может быть использован дополнительный механизм "push" — поршень. Использование этого механизма обеспечивает форсирование передачи содержимого буфера в тот момент, когда в него попадают данные, которые должны быть переданы как можно скорее.

По возможности транспортное соединение должно обеспечивать режим одновременной передачи и приема данных взаимодействующими приложениями. Помимо увеличения результирующей скорости информационного обмена, этот режим обеспечивает также повышение эффективности использования канала передачи данных.

Взаимодействие с приложениями

Протокол TCP (Transmission Control Protocol) был спроектирован в качестве связующего протокола для обеспечения информационного взаимодействия в сети Интернет. Протокол TCP, с одной стороны, взаимодействует с прикладным протоколом пользовательского приложения, а с другой — с протоколом, обеспечивающим "низкоуровневые" функции: маршрутизацию и адресацию пакетов, которые, как правило, выполняет IP.

В отличие от протокола UDP, протокол TCP обеспечивает надежную и гарантированную доставку передаваемой информации, поскольку является протоколом транспортного уровня, ориентированным на установление соединения (connection-oriented). При передаче данных протокол TCP использует схему позитивного подтверждения с повторной передачей (positive acknowledgement with retransmission). Применение подобной схемы обеспечивает динамическое управление скоростью информационного обмена, что позволяет успешно применять протокол TCP на самых разнообразных каналах передачи данных.

В операционной системе протокол TCP реализуется в виде отдельного системного модуля (драйвера), через который, как правило, проходят все вызовы функций протокола. Интерфейс между прикладным процессом и протоколом TCP представляет собой библиотеку вызовов, такую же, как библиотека системных вызовов, например, для работы с файлами. Вы можете открыть или закрыть соединение (как открыть или закрыть файл) и отправить или принять данные из установленного соединения (аналогично операциям чтения и записи файла). Вызовы протокола TCP могут взаимодействовать с прикладным приложением в асинхронном режиме.

Схема взаимодействия пользовательского приложения с протоколом TCP, в общих чертах, состоит в следующем. Для передачи данных пользовательскому процессу надо вызвать соответствующую функцию протокола TCP с указанием буфера передаваемых данных. Протокол TCP упаковывает эти данные в сегменты своего стека и вызывает функцию передачи протокола нижнего уровня, в нашем случае — протокола IP.

На приемной стороне получатель группирует поступившие от своего протокола сетевого уровня сегменты, проверяет целостность размещенных в них данных, передает сегменты пользовательскому процессу, после чего уведомляет отправителя об их получении.

При взаимодействии с протоколом TCP процесс пользователя может выполнять следующие процедуры:

- открыть (OPEN) или закрыть (CLOSE) соединение;
- отправить (SEND) или принять (RECEIVE) данные;
- уточнить состояние соединения (STATUS).

Эти процедуры подобны вызовам функций операционной системы из пользовательской программы, таким как открытие, чтение или закрытие файла.

Модель OSI не устанавливает дополнительных требований к процедурам организации взаимодействия протокола TCP с протоколами верхних уровней. Должен быть, однако, определен механизм, обеспечивающий асинхронную передачу информации от одного уровня к другому. Результатом работы этого механизма является инкапсуляция блока данных протокола более высокого уровня в блок данных протокола более низкого уровня. Реализуется этот механизм через интерфейс вызовов между протоколами TCP и IP. В результате работы этого механизма каждый TCP-пакет вкладывается в "конверт" протокола нижнего уровня, например, IP. Получившаяся таким образом дейтаграмма содержит в себе TCP-пакет, точно так же, как TCP-пакет содержит пользовательские данные.

Схема информационного взаимодействия транспортных протоколов Интернета представлена на рис. 5.2.

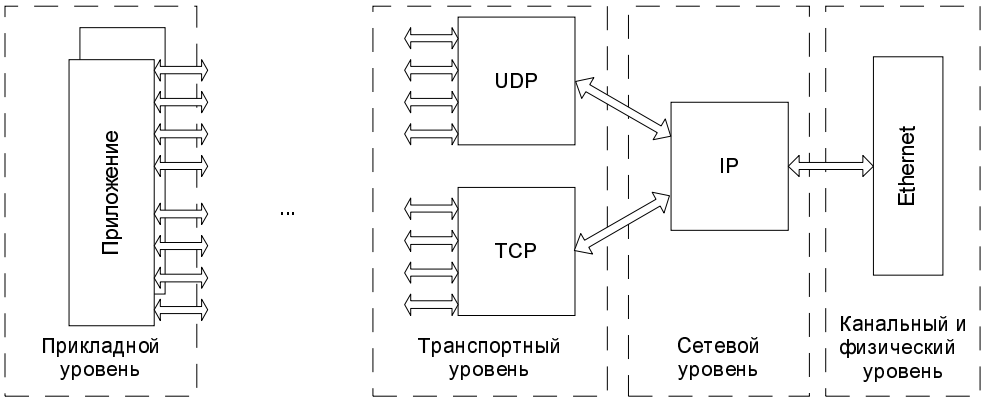


Рис. 5.2. Схема информационного взаимодействия транспортных протоколов Интернета

На представленной схеме прямоугольники обозначают обработку данных, а линии, соединяющие прямоугольники, — пути передачи данных.

Мультиплексирование информационных потоков, механизм гнезд (sockets)

Для установления соединения между двумя процессами на различных компьютерах сети необходимо знать не только Интернет-адреса компьютеров, но и номер TCP-порта, который процесс использует на данном компьютере. В совокупности с Интернет-адресом компьютера TCP-порт образует гнездо. Пара гнезд уникально идентифицирует каждое соединение или поток данных в сети Интернет, а порт обеспечивает независимость каждого TCP-канала на данном компьютере.

Типичным прикладным процессом, использующим протокол TCP, является приложение FTP (File Transfer Protocol, Протокол передачи файлов). Стек протоколов в этом случае будет выглядеть следующим образом: FTP/TCP/IP/ETHERNET. При использовании протокола UDP (User Datagram Protocol, Протокол дейтаграмм пользователя) данные передаются между прикладным процессом и модулем протокола UDP. Например, протокол SNMP (Simple Network Management Protocol, Простой протокол управления сетью) пользуется транспортными услугами протокола UDP. Его стек протоколов выглядит так: SNMP/UDP/IP/ETHERNET.

Одно гнездо на компьютере может быть задействовано в соединениях с несколькими гнездами на удаленных компьютерах. Кроме того, через одно и то же гнездо поток данных может передаваться в обоих направлениях, таким образом, механизм гнезд уникально идентифицирует каждый поток данных и позволяет на одном компьютере одновременно работать нескольким

приложениям. Эта функция называется мультиплексированием соединений транспортного уровня.

Модули протоколов TCP и UDP, а также, драйвер Ethernet являются мультиплексорами типа $n \times 1$. Функционируя в качестве мультиплексоров, они переключают несколько входов на один выход. Эти модули выполняют также функции демультиплексоров $1 \times n$. Действуя, в качестве демультиплексоров, они переключают входной поток на один из выходов в соответствии со значением, размещенным в заголовке блока данных протокола верхнего уровня. Когда Ethernet-кадр обрабатывается драйвером сетевого интерфейса Ethernet, он может быть направлен либо в модуль протокола ARP, либо в модуль протокола IP. В данном случае значение, размещенное в поле **Type/Code** заголовка кадра, указывает, куда именно должен быть направлен блок данных, переносимый эти Ethernet-кадром.

Когда IP-пакет принимается модулем протокола IP, то содержащиеся в нем данные могут быть переданы либо модулю протокола TCP, либо — UDP, что определяется значением поля **Protocol** заголовка этого пакета. Когда сегмент TCP принимается модулем TCP, выбор прикладной программы, которой должен быть передан блок данных, осуществляется на основе значения поля **Destination Port** заголовка сегмента.

Мультиплексирование данных в обратную сторону осуществляется довольно просто, поскольку из каждого модуля существует только один путь вниз. Модуль каждого протокола добавляет к блоку данных протокола верхнего уровня свой заголовок, на основании которого узел, принявший этот блок данных, может выполнить демультиплексирование.

Назначение портов приложениям на каждом узле сети происходит независимо. Протокол TCP может самостоятельно выбирать порт, с которым будет работать приложение, или приложение само укажет, с каким портом на данном компьютере оно будет работать. Однако, как правило, приложения-сервисы используют одни и те же номера портов, которые уже стали общеизвестными, например, такие как HTTP, FTP, SMTP и др., для того, чтобы к данному процессу на компьютере можно было присоединиться, указывая только адрес машины. Например, Интернет-браузер, если ему не указать дополнительно, ищет по указанному адресу приложение, работающее с портом 80, — это наиболее распространенный порт для серверов WWW.

Кроме того, машина может быть снабжена несколькими сетевыми интерфейсами, тогда она должна осуществлять мультиплексирование типа $n \times m$, т. е. между несколькими прикладными программами и сетевыми интерфейсами.

Установление соединения и передача данных

Соединение протокола TCP устанавливается путем вызова функции OPEN с аргументами в виде номера локального порта и гнезда (IP-адрес + порт)

удаленного процесса. Функция OPEN вызывается в том случае, когда процесс намерен передавать информацию (активный OPEN), или в том случае, когда процесс ожидает поступления информации (пассивный OPEN). Функция возвращает значение идентификатора соединения, на который пользовательский процесс может ссылаться при последующих вызовах. Идентификатор соединения указывает на структуру данных, в которой хранятся переменные и информация данного TCP-соединения. Эта структура данных называется TCB (Transmission Control Block, Блок управления соединением).

В структуре TCB хранятся такие параметры соединения, как адреса локального и удаленного гнезд, указатели на полученные и отправляемые пользовательские данные, указатели на очередь блоков для повторной отправки, номер текущего сегмента и т. д., то есть хранится вся информация, используемая данным соединением.

Как уже было отмечено выше, соединение может быть открыто в активном или пассивном режиме. Пассивный режим обозначает, что связанный с этим соединением процесс ожидает поступления специального сигнала открытия соединения и не будет пытаться открыть канал самостоятельно. Иными словами, канал, открытый одной стороной в пассивном режиме, ожидает инициирующего сигнала от своего партнера. Этот режим используется процессами, которые предоставляют свой сервис через заранее известный номер своего порта (например, HTTP, SMTP и т. д.).

Процесс может вызвать функцию пассивного открытия канала, ждать прихода сигнала активного открытия канала от другого процесса, и только после его получения соединение будет установлено. Соединение также устанавливается, если два процесса активно откроют канал навстречу друг другу. Эта гибкость в организации соединения особенно важна в распределенных сетях, когда компьютеры работают асинхронно.

Обеспечение гарантированной доставки данных

Протокол TCP обеспечивает информационный обмен в ситуациях, когда передаваемые по сети пакеты повреждаются, теряются, дублируются или поступают на приемник с нарушением первоначального порядка следования. Такая возможность достигается присвоением порядкового номера каждому передаваемому блоку данных и использования механизма подтверждения доставки передаваемых блоков данных.

Когда протокол TCP передает очередной сегмент, его копия помещается в очередь повтора передачи и затем запускается таймер ожидания подтверждения. Когда система получает подтверждение в виде TCP-сегмента, содержащего управляющий флаг ACK, означающий что этот сегмент получен, то она удаляет его копию из очереди. Сегмент подтверждения содержит номер полученного сегмента, на основании которого и происходит контроль

доставки данных адресату. Если подтверждение не поступило до истечения срока таймера, сегмент отправляется повторно. Уведомление протокола TCP о получении сегмента еще не означает, что он был доставлен конечному пользователю. Оно только означает, что протокол TCP выполнил возложенные на него функции.

При использовании протокола TCP каждому байту данных присваивается порядковый номер, поэтому, в каком бы порядке сегменты не достигали точки назначения, размещенные в них байты исходного сообщения всегда будут собраны в изначально заданной последовательности. Порядковый номер первого байта данных в передаваемом сегменте называется порядковым номером сегмента. Нумерация проводится "с головы состава", т. е. от заголовка пакета. TCP-сегмент содержит номер подтверждения (acknowledgment number), который представляет собой номер последнего полученного сегмента. Иными словами, номер обозначает: "до сих пор я все получил".

На рис. 5.3 представлена схема организации передачи и приема данных с квитированием.

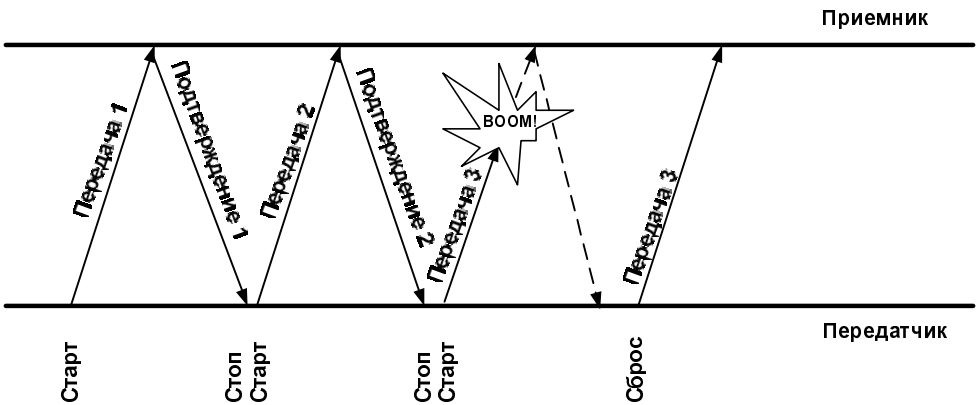


Рис. 5.3. Схема организации передачи и приема данных с квитированием

Здесь стоит сказать несколько слов о механизме предотвращения появления в сети пакетов с одинаковыми номерами. Они могут появиться, например, при установлении и быстром сбросе соединения или при сбросе соединения и его быстром восстановлении, т. е. когда номер испорченного пакета может сразу использоваться новым пакетом данных. Механизм предотвращения подобных ситуаций основан на генерировании случайного стартового номера байта сообщения.

Так, при установлении нового соединения генерируется 32-битное число ISN (Initial Sequence Number). Генератор использует 32 младших разряда машинного таймера, значения которого меняются каждые 4 микросекунды (полный цикл — 4,55 часа). Кроме того, каждая дейтаграмма в сети имеет ограниченное время жизни TTL, которое заведомо меньше периода повторения

счетчика. Таким образом, в сети гарантируется невозможность появления сегментов пакетов с одинаковыми последовательными номерами.

Поврежденные пакеты отсеиваются механизмом проверки контрольной суммы данных, которая размещается в каждом передаваемом пакете.

Процедура установления соединения использует специальный флаг синхронизации — SYN и состоит из трех этапов обмена сообщениями — так называемое "трехэтапное квитирование". Использования именно трех квитирующих сообщений всегда достаточно, чтобы однозначно синхронизировать потоки данных.

Инициализация соединения начинается с обмена пакетами, которые отправляются при открытии канала пользователем, например, командой OPEN, и содержат флаг SYN и начальный порядковый номер.

Соединение считается установленным, когда в обоих направлениях синхронизируются последовательности нумерации передаваемых блоков данных, т. е. обе стороны "знают", какой блок данных и с каким номером поступит с противоположного конца соединения. Схема организации квитирования в протоколе TCP при создании и разрыве соединения приведена на рис. 5.4.

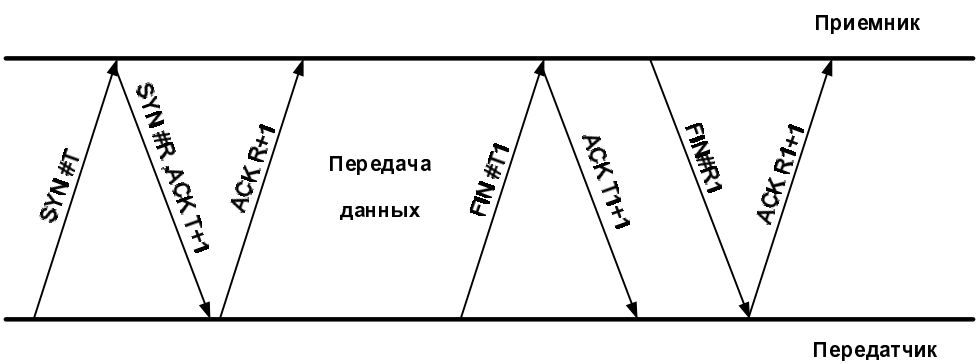


Рис. 5.4. Схема организации квитирования при создании и разрыве соединения в протоколе TCP

Соединение закрывается, когда стороны обмениваются пакетами, содержащими команду `FIN` (конец). При этом все ресурсы системы, занятые этим соединением, должны быть освобождены.

Пользователь, отправляющий пакет вызовом команды `SEND`, указывает, передаются ли в сегменте или в предшествующих сегментах данные немедленно, или при передаче будет использоваться механизм буферизации данных, тогда данные будут отправляться в соответствии с установками системы.

Примечание

Под словом "пользователь" здесь и далее будет пониматься пользовательский процесс или приложение, которое работает с протоколом TCP.

В том случае, если модуль TCP получает блок данных с флагом PUSH, вся буферизированная информация, вне зависимости от того, сформирована она одним вызовом команды SEND или несколькими, будет немедленно отправлена получателю.

Назначение PUSH-функции и PUSH-флага состоит только в "проталкивании" данных к пользователю, минуя механизм кэширования. Команда PUSH не производит никаких дополнительных группировок или других действий над данными.

Управление темпом информационного обмена

Протокол TCP позволяет получателю регулировать передаваемый отправителем поток данных. При передаче флага подтверждения получения пакета (ACK) в TCP-сегменте получатель информации передает размер "окна" — блока данных, который может быть передан отправителем без получения подтверждения от получателя. Иными словами — получатель передает размер свободного места в своем буфере, куда записываются принятые и ожидающие дальнейшей обработки блоки данных. Из буфера данные обрабатываются и передаются соответствующим процессам. Использование буфера позволяет избегать "пробок" при передаче данных между системами, имеющими различную производительность.

Размер "окна" задается количеством байтов, отсчитываемых от номера, заданного в поле **Acknowledgment Number** последнего полученного сообщения. Нулевой размер окна для отправителя аналогичен получению команды о приостановлении передачи. Необходимо заметить, что в этом случае отправитель продолжает посылать пустые сегменты и может возобновить передачу своих данных только после изменения размера "окна".

Безусловно, большой размер "окна" позволяет передавать данные быстрее, поскольку отправителю пакета не нужно ждать сигнала готовности к приему от получателя. Однако в случае сбоя передачи, соответственно, возрастет объем данных, которые нужно будет отправить заново. При небольшом же размере "окна" потерянные сегменты данных можно восстановить с минимальными затратами.

Механизм контроля потока данных позволяет протоколу TCP влиять на изменение размера "окна" передачи. Это, в свою очередь, помогает существенно оптимизировать скорость достоверного обмена данными между процессами в сетях Интернет.

Протокол TCP прозрачен для параметов секретности и приоритета пакета, определяемых средствами протокола IP, что позволяет выполнять их обработку для обеспечения информационной безопасности и заданного уровня качества обслуживания. В спецификации протокола TCP термин "секретность и изолированность" обозначает безопасность соединения, изолиро-

ванность потоков данных и другие ограничения, которые обеспечивает IP. Кроме того, протокол TCP создает условия для осуществления некоторых дополнительных механизмов, которые поддерживаются специальным аппаратным и программным обеспечением.

Протокол TCP имеет в своем распоряжении механизм работы с флагом важности пакетов — **URG** (см. ниже). Этот механизм позволяет отправителю "настоятельно сообщать" получателю о том, что тот или иной пакет содержит срочную информацию, и позволяет получателю установить момент, когда вся срочная информация будет им получена. Такой механизм уведомления, т. е. манипулирование с URG-флагом TCP сегмента, используется, например, при обработке асинхронных событий.

Формат сегмента протокола TCP

Для обеспечения достоверной и управляемой передачи данных протокол TCP разбивает входящий информационный поток на блоки — сегменты. Каждый из сегментов состоит из заголовка — TCP-header и поля передаваемых данных.

В заголовке сегмента размещается информация, необходимая для обеспечения функции гарантированной доставки данных, а также признак кода сегмента, размер представляемого окна, адреса портов источника и назначения.

Структура заголовка сегмента TCP представлена на рис. 5.5.

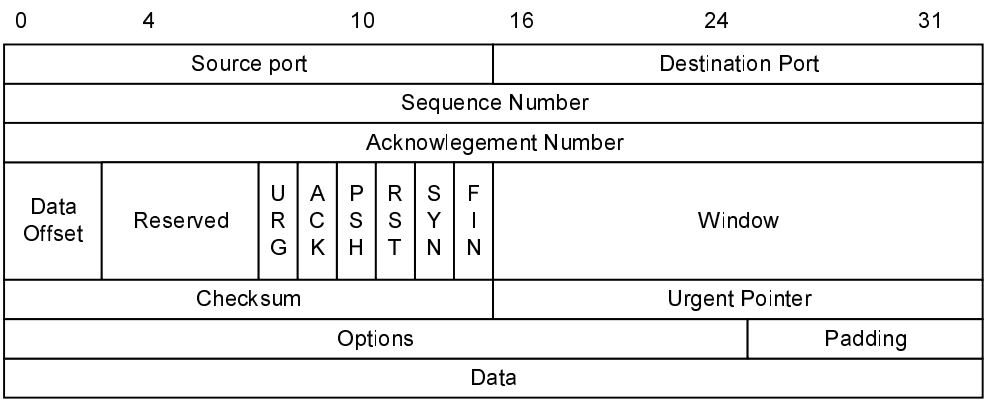


Рис. 5.5. Структура заголовка сегмента протокола TCP

Поле **Source Port** (16 бит). Порт отправителя.

Поле **Destination Port** (16 бит). Порт получателя.

Поле **Sequence Number** (32 бита). Номер сегмента. Номер первого байта данных в этом сегменте (за исключением пакета, где присутствует флаг **SYN**). Если в пакете присутствует флаг **SYN**, то номер данного пакета становится

номером начала последовательности (ISN (Initial Sequence Number)) и номером первого байта данных становится номер ISN+1.

Поле **Acknowledgment Number** (32 бита). Поле номера подтверждения. Если сегмент содержит установленный контрольный бит **ACK**, то это поле содержит номер следующего сегмента данных отправителя, который ожидает получатель. При установленном соединении сегмент подтверждения отправляется всегда.

Поле **Data Offset** (4 бита). Поле величины смещения данных. Оно содержит количество 32-битных слов заголовка TCP-сегмента. Это число определяет смещение расположения данных в пакете.

Поле **Reserved** (6 бит). Резервное поле. Поле зарезервировано.

Флаги управления (слева направо):

- **URG** — флаг срочности;
- **ACK** — флаг пакета, содержащего подтверждение получения;
- **PSH** — флаг форсированной отправки;
- **RST** — переустановка соединения;
- **SYN** — синхронизация чисел последовательности;
- **FIN** — флаг окончания передачи со стороны отправителя.

В поле **Window** (16 бит) помещается размер "окна" — количество байт данных, которое отправитель может принять без уведомления о получении, отсчитанное от номера байта, указанного в поле **Acknowledgment Number**.

В поле **Checksum** (16 бит) помещается контрольная сумма сегмента. Если сегмент содержит нечетное число байт заголовка и данных, последний байт дополняется справа нулями. При вычислении контрольной суммы поле контрольной суммы полагается равным нулю.

Поле **Urgent Pointer** (16 бит) предназначено для размещения указателя срочных данных. Это поле содержит значение счетчика пакетов, начиная с которого следуют пакеты повышенной срочности. Значение поля **Urgent Pointer** анализируется только в сегментах с установленным флагом **URG**.

Поле дополнительных параметров **Options** имеет переменную длину. Описание назначения этого поля выходит за рамки настоящей главы.

Поле **Padding** (Заполнение) имеет переменную длину. Заполнение заголовка TCP-сегмента нулями при необходимости используется для выравнивания его длины до кратной 32-битному слову.

Особенности практической реализации протокола TCP

При использовании протокола TCP на линиях, пропускная способность которых отличалась в различных направлениях, пользователи могли наблюдать

возникновение ситуации, получившей название "синдром неоптимального окна" — SWS (Silly Window Syndrome). В такой ситуации одно из взаимодействующих приложений, назовем его "A", может передавать данные с существенно большей скоростью, чем другое — "B". Если изначально сторона "B" установила размер своего окна равным величине своего буфера, вполне может получиться так, что сторона "A" заполнит весь буфер до того, как получит первое уведомление об изменении размера окна. Исчерпав лимит байтов, установленный для передачи, сторона "A" перейдет в режим ожидания подтверждения. Когда сторона "B" начнет обработку поступивших данных, она сможет освободить некоторую часть буфера и передаст уведомление о соответствующем изменении размера приемного окна. Сторона "A" быстро заполнит освободившееся место в буфере и опять перейдет в режим ожидания. Наиболее неприятным следствием возникновения такой ситуации будет то, что канал передачи данных в направлении от "A" к "B" будет использоваться крайне неэффективно, поскольку сегменты протокола TCP будут применяться для переноса небольших объемов данных (до 1 байта). Соотношение долей полезной нагрузки и служебной информации в данном случае будет крайне неудачным. Для того чтобы избежать возникновения синдрома SWS, в практической реализации протокола TCP используются несколько различных способов.

Чтобы предотвратить возникновение синдрома SWS, приемной стороне достаточно передавать представления только для больших изменений размеров окна. Это означает, что сообщение ACK с новым значением размера окна передается не сразу после того, как появится свободное место во входном буфере, а только после того, как размер этого свободного места будет достаточен для приема минимального установленного объема передаваемой информации. Например, в качестве такого минимального объема может быть использован половинный объем приемного буфера. Некоторые реализации протокола TCP могут также использовать в качестве величины окна максимальную длину передаваемого сегмента.

Другим способом, который также может быть использован на приемной стороне для предотвращения возникновения эффекта синдрома SWS, является процедура задержки подтверждений. Этот метод довольно прост для реализации и в тоже время достаточно эффективен. Действительно, задержка ответа на некоторое постоянное время позволит избежать синдрома SWS и одновременно повысить эффективность использования канала передачи данных — поскольку на все сегменты, которые поступят на приемник в течение интервала задержки, будет сформировано только одно подтверждение. Использование фиксированной задержки подтверждения рекомендовано стандартом для предотвращения синдрома SWS. Следует, однако, иметь в виду, что выбор слишком большого времени задержки может привести к повторной передаче сегмента.

Для предотвращения появления синдрома SWS передающая сторона может использовать алгоритм Нейгла (Nagle). Суть этого алгоритма заключается

в том, что первая порция информации передается немедленно после попадания в буфер, все последующие дожидаются, пока в буфере накопится достаточный для передачи объем данных.

Протокол RTP

Стремительный рост Интернета предъявляет новые требования к скорости и объемам передачи данных. И для того чтобы удовлетворить все эти запросы, одного увеличения емкости сети недостаточно, необходимы разумные и эффективные методы управления трафиком и уровнем загруженности каналов передачи данных.

Изначально сети TCP/IP предоставляли всем приложениям только простейшую услугу по доставке данных по мере возможности. Однако со временем потребности пользователей изменились. Организации, потратившие значительные средства на передачу данных между локальными сетями, сталкиваются теперь с тем, что построенные структуры не способны эффективно поддерживать новые мультимедийные приложения реального времени.

В приложениях реального времени отправитель генерирует поток данных с постоянной скоростью, и предполагается, что приложение на стороне получателя должно принимать эти данные с той же самой скоростью. Такие приложения включают, например, аудио- и видеоконференции, живое видео, удаленную диагностику в медицине, компьютерную телефонию, распределенное интерактивное моделирование, игры, мониторинг в реальном времени и др.

Протокол TCP является наиболее широко используемым протоколом транспортного уровня в сети Интернет. Однако, несмотря на то, что этот протокол является достаточно универсальным и позволяет поддерживать множество разнообразных распределенных приложений, он совершенно не подходит для обеспечения транспортировки потокового трафика приложений реального времени.

Использование протокола TCP в качестве транспортного протокола для этих приложений невозможно по нескольким причинам.

Протокол TCP предполагает организацию соединения между двумя конечными точками, следовательно, не может быть использован для многоадресной передачи, когда количество передатчиков и приемников заранее неизвестно.

Повторная передача протоколом TCP потерянных сегментов неминуемо будет приводить к дополнительным нарушениям информационного обмена, поскольку эти сегменты будут прибывать в тот момент, когда приложение реального времени уже их не ждет.

Протокол TCP не обладает механизмом привязки информации о синхронизации передаваемых данных к сегментам, который является обязательным для приложений реального времени.

Другой популярный протокол транспортного уровня сети Интернет — UDP не имеет некоторых из перечисленных выше недостатков протокола TCP, однако также не обладает возможностью предоставления информации о временной синхронизации передаваемых данных.

Несмотря на то, что каждое приложение реального времени может иметь свои собственные механизмы для обеспечения передачи данных в реальном масштабе времени, они имеют много общих черт, что делает использование единого универсального протокола весьма желательным.

В типичной среде реального времени отправитель генерирует блоки данных с постоянной скоростью. Они отправляются через одинаковые интервалы времени, проходят через сеть и принимаются получателем, воспроизводящим данные в реальном масштабе времени по их получении. Однако ввиду изменения времени задержки при передаче блоков данных по сети, они могут прибывать к получателю через нерегулярные интервалы времени. Поэтому для восстановления исходного вида сигнала реального времени необходимо, чтобы каждый блок данных, передаваемый по сети, содержал метку времени. При наличии такой метки получатель сможет определить взаимное расположение во времени получаемых порций и воспроизвести поступающие данные хотя и с небольшой задержкой, но в том же временном масштабе, в котором они были сформированы отправителем.

Эту задачу и призван решить специальный транспортный протокол реального времени — RTP (Real Time Transport Protocol), который способен передавать данные с привязкой к моменту времени их формирования.

Принципы построения протокола RTP

Выше было отмечено, что основная особенность потокового голосового и видеотрафика состоит в наличии жесткой привязки каждого передаваемого фрагмента данных к определенному моменту времени. Именно поэтому для обозначения подобных разновидностей трафика используют обобщенный термин — трафик реального времени. Данные трафика реального времени передаются, как правило, небольшими порциями (chunks). В том случае, если для выполнения оцифровки входящего аналогового потока используются стандартные прямые методы, порции данных формируются и должны передаваться с постоянным интервалом времени. Для повышения эффективности использования канала передачи данных при выполнении преобразования аналогового потока могут быть также использованы дополнительные алгоритмы сжатия. В этом случае порции данных формируются с переменным интервалом времени. В те моменты времени, когда источник аналогового трафика проявляет активность, эти интервалы уменьшаются,

при понижении активности источника они увеличиваются. Однако в обоих случаях, для того чтобы выполнить адекватное восстановление исходного вида аналогового сигнала на приемной стороне, соответствующее приложение приемника должно получать информацию о взаимном временном расположении полученных порций входных данных.

Строго говоря, протокол RTP не является отдельным транспортным протоколом, а представляет собой лишь универсальную надстройку для стандартного протокола транспортного уровня (в рассматриваемом варианте — для протокола UDP).

Следует особо отметить, что протокол RTP не поддерживает каких-либо механизмов гарантированной доставки блоков данных, обеспечения достоверности их передачи или контроля надежности соединения. При необходимости все эти функции могут быть возложены на протокол транспортного уровня.

Функции, выполняемые протоколом RTP, разделены между двумя функционально обособленными протоколами:

- RTP (Real Time Transport Protocol) — собственно транспортный протокол реального времени;
- RTCP (RTP Control Protocol) — управляющий протокол RTP.

Назначение протокола RTP состоит в непосредственной передаче блоков данных трафика реального времени, снабженных метками времени.

Управляющий протокол RTCP обеспечивает оперативный обмен управляющей информацией между узлами, выполняющими прием и передачу потоковых данных в соответствии с правилами протокола RTP.

Поскольку подавляющее большинство сетевых приложений, формирующих трафик реального времени, предназначено для использования в режиме многоадресной передачи (Multicast), комплекс протоколов RTP также имеет ряд механизмов и функций, учитывающих особенности информационного обмена в этом режиме.

В частности, сеансом RTP (RTP session) принято называть отношения между группой участников информационного обмена с использованием протокола RTP. Для каждого из участников такой сеанс определяется совокупностью трех значений.

1. IP-адресом узла назначения.
2. Номером порта протокола транспортного уровня, используемого для передачи сообщений протокола RTP.
3. Номером порта протокола транспортного уровня, используемого для передачи сообщений протокола RTCP.

В том случае, если для информационного обмена в сеансе используется групповая адресация, число участников может быть переменным, причем у всех участников сеанса значение первого параметра будет одинаковым.

Примечание

При проведении мультимедийных конференций (например, голос + видео) для каждого из информационных потоков должен быть использован отдельный сеанс.

Если для организации сеанса используется обычный режим Unicast, число участников должно быть равно 2, каждый из которых в качестве IP-адреса узла назначения будет использовать адрес своего партнера.

Любой из узлов сети может одновременно принимать участие в нескольких групповых и одноадресных сессиях. Для разделения информационных потоков в подобной ситуации используются различные адреса назначения.

Протокол RTP обеспечивает выполнение двух основных функций объединения и преобразования трафика:

- Mixing — объединение;
- Translation — преобразование.

При объединении порций, принадлежащих разным потокам одной сессии, обеспечиваются дополнительные возможности повышения качества воспроизведения передаваемого потокового сигнала и экономия сетевых ресурсов.

В качестве примера, иллюстрирующего необходимость подобных преобразований, рассмотрим схему, представленную на рис. 5.6.

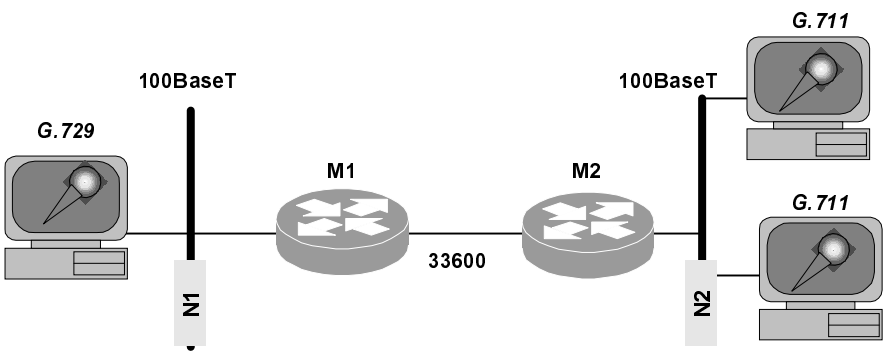


Рис. 5.6. Пример выполнения объединения и преобразования потоковых сигналов

Предположим, что в представленной сети организована аудиоконференция, в которой участвуют узлы сетей N1 и N2. Проблема в данном случае заключается в том, что эти сети связаны низкоскоростным каналом, который не позволяет применять кодирующие схемы, обеспечивающие высокое качество воспроизведения голосового сигнала всем участникам сессии. Подобная проблема может быть решена двумя путями:

- использованием одинаковых низкоскоростных кодеков всеми участниками сессии;

- преобразованием (трансляцией) формы представления аналоговых сигналов на маршрутизаторах.

Совершенно очевидно, что второй метод является гораздо более предпочтительным, поскольку позволяет обеспечить максимальное качество передачи голосового сигнала в условиях ограниченной пропускной способности используемого канала передачи данных.

Функция смешивания позволяет объединить несколько однонаправленных информационных потоков в один и при необходимости выполнять преобразование порций информации, сформированных при использовании различных кодирующих схем.

В тех случаях, когда вследствие особенностей топологии сети информационный обмен с некоторыми из участников сессии должен быть выполнен с использованием особого режима адресации, возникает необходимость в применении функции трансляции.

Транслятор создает один исходящий пакет протокола RTP для каждого поступающего пакета. Транслятор может изменить формат данных в пакете или использовать иной комплект низкоуровневых протоколов для передачи данных из одного домена в другой.

Трансляция адресов, например, обязательно должна присутствовать в том случае, когда область распространения группового трафика сессии административно ограничивается некоторой зоной, за пределами которой оказывается часть участников этой сессии. На рис. 5.7 представлен пример сессии, участники которой размещены в областях, разделенных межсетевыми экранами, ограничивающими распространение группового трафика. В подобных случаях для организации передачи потокового трафика между этими областями может быть использован тоннель, построенный с использованием функции трансляции протокола RTP.

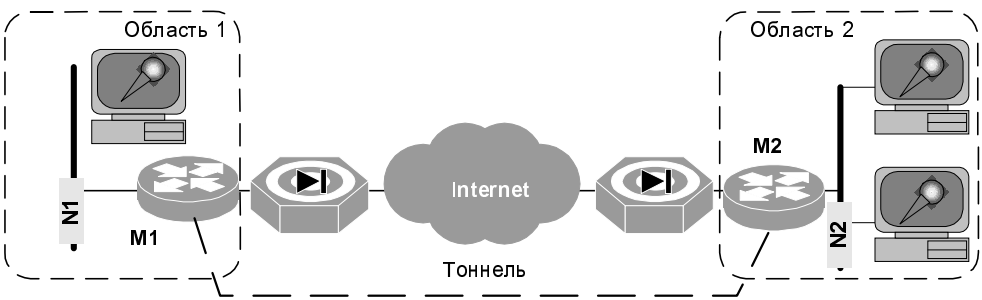


Рис. 5.7. Применение трансляции для передачи потокового трафика между несвязанными областями.

Очень важным для протокола RTP является понятие источника синхронизации (Synchronization source). *Источником синхронизации* называется участник

сессии, который формирует временные метки и прикрепляет их к порциям оцифрованного потокового трафика. В качестве источников синхронизации могут выступать как первичные источники трафика (например, видео- или аудиокодеки), так и вторичные источники (например, микшеры)

Примечание

Определения микшеров и кодеков и примеры их использования приведены в *гл. 7*.

Уже было отмечено, что микшеры выполняют функцию объединения нескольких потоков, принадлежащих к одной сессии в единый поток. При выполнении этого преобразования сам микшер становится источником синхронизации, поскольку он прикрепляет к сгруппированным порциям трафика, поступающего от источников заполнения (Contributing source) свои временные метки.

Для обеспечения однозначной идентификации каждого из участников сессии применяются сформированные случайным образом 32-разрядные числа. Следует отметить, что уникальность идентификатора должна быть обеспечена только в пределах данной сессии. Один узел может иметь разные идентификаторы для каждой из сессий, в которых он принимает участие.

Заголовок сообщения протокола RTP содержит следующие поля: идентификатор отправителя, указывающий, кто из участников генерирует данные, отметки о времени генерирования пакета, чтобы данные могли быть воспроизведены принимающей стороной с правильными интервалами, информация о порядке передачи, а также информация о характере содержимого пакета, например, о типе кодировки видеоданных. Наличие такой информации позволяет оценить величину начальной задержки и объема буфера передачи.

Таким образом, протокол RTP применяется только для передачи пользовательских данных — обычно с использованием многоадресного режима — между участниками сеанса. Все функции управления передачей данных, выполняемой протоколом RTP, осуществляются протоколом RTCP (Real Time Transport Control Protocol). Основной задачей протокола RTCP, таким образом, является организация и обслуживание сессий протокола RTP.

Для передачи управляющих сообщений протокола RTCP применяется обычно тот же самый базовый транспортный протокол, что и для протокола RTP (обычно UDP), однако при этом для протокола RTCP используется отличный от протокола RTP номер порта.

Протокол RTCP выполняет нескольких основных функций, таких как:

1. Обеспечение обратной связи между передатчиками и приемниками группового трафика для оперативного определения качества передаваемого сигнала. Эта функция относится к основным и обязательным

функциям протокола RTCP. Так как диагностические сообщения протокола RTCP передаются приемниками потокового трафика в режиме групповой адресации, все заинтересованные участники сеанса могут оценить качество передаваемого сигнала и степень загруженности сети. Диагностические сообщения протокола RTCP содержат информацию о проблемах, которые возникли при приеме потокового трафика, например, потеря пакетов или большая неравномерность передачи.

Обратная связь с получателями важна также для диагностирования ошибок при распространении потокового трафика по сети. Например, анализируя сообщения всех участников сеанса, администратор сети может определить, является ли возникшая проблема локальной проблемой одного участника или носит глобальный характер. Если приложение-отправитель или администратор сети приходит к выводу, что проблема характерна для системы в целом, допустим, по причине перегруженности одного из используемых каналов связи, они могут, например, сформировать предложения по изменению степени сжатия данных, передаваемых по этому каналу.

2. Идентификация источника передаваемых данных. Эта функция также относится к основным и обязательным функциям протокола RTCP. Сообщения протокола RTCP содержат, в отличие от случайного идентификатора источника синхронизации протокола RTP, стандартное текстовое описание отправителя и, следовательно, предоставляют персональную информацию об участнике сессии. Использование сообщений протокола RTCP, таким образом, обеспечивает однозначную идентификацию пользователя, участвующего в нескольких различных сеансах одновременно.

3. Оценка числа участников сеанса и, на основании этой оценки, определение оптимального периода формирования RTCP-сообщений. Очевидно, что число передаваемых сообщений протокола RTCP будет расти с увеличением числа участников сессии. При небольшом числе участников один пакет RTCP посылается максимум каждые 5 секунд. Цель определения оптимального периода формирования RTCP-сообщений состоит в том, чтобы трафик сообщений протокола RTCP не превышал установленной доли общего трафика сеанса.

Вспомогательной функцией протокола RTCP является передача управляющей информации сессии для отображения ее в пользовательском интерфейсе. Предполагается, что формирование и передача подобной информации будут целесообразными для обслуживания сессий, использующих упрощенные процедуры членства. Поскольку для участия в такой сессии, как правило, не требуется выполнения процедур опознавания пользователя, полезным может оказаться наличие специального канала для передачи дополнительной информации об ее участниках.

Сообщения протокола RTP

Как уже было выше отмечено, протокол RTP предназначен для передачи потоковых данных в реальном масштабе времени. Сообщения протокола RTP состоят из заголовка, в котором размещается управляющая информация, и поля полезной нагрузки, в которое помещаются блоки передаваемых данных. Заголовки сообщений протокола RTP содержат информацию о порядке следования и временном положении передаваемых порций оцифрованных потоковых данных, с тем чтобы обеспечить адекватное восстановление переданного сигнала на приемной стороне.

Каждое сообщение протокола RTP снабжается основным (фиксированным) заголовком, к которому, при необходимости, могут быть добавлены дополнительные поля, специфичные для конкретного приложения.

На рис. 5.8 представлена структура фиксированного заголовка сообщения протокола RTP, в котором содержится ряд полей, идентифицирующих такие элементы, как формат пакета, порядковый номер, источники, границы и тип полезной нагрузки.

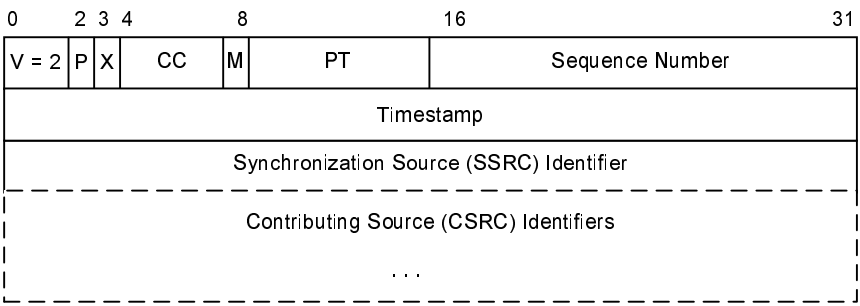


Рис. 5.8. Структура фиксированного заголовка сообщения протокола RTP

Поле **V** (2 бита). Поле версии протокола RTP. Текущая версия — вторая.

Признак **P** (1 бит). Признак заполнения. Этот признак сигнализирует о наличии заполняющих байтов в конце поля полезной нагрузки. Заполнение применяется, когда приложение требует, чтобы размер полезной нагрузки был кратен, например, 32 битам. Если в сообщении протокола RTP бит **P** установлен в 1, последний байт этого сообщения содержит число заполняющих байтов.

Признак **X** (1 бит). Признак расширения заголовка. Если в сообщении протокола RTP бит **X** установлен в 1, то за основным заголовком следует еще один дополнительный, используемый в экспериментальных расширениях протокола RTP.

Поле **CC** (4 бита). Поле количества использованных источников заполнения. В этом поле содержится количество источников потоковых данных, передаваемых в настоящем сообщении протокола RTP.

Признак **M** (1 бит). Признак маркера. Назначение этого признака определяется типом полезной нагрузки. Бит маркера обычно используется для указания границ потока данных. В случае видео он задает конец кадра. В случае голоса он задает начало речи после периода молчания.

Поле **PT** (7 бит). Поле типа полезной нагрузки. Это поле идентифицирует тип полезной нагрузки и формат данных, включая сжатие и шифрование. В стационарном состоянии отправитель использует только один тип полезной нагрузки в течение сеанса. Соответствие структуры поля полезной нагрузки и значения поля **PT** определятся профилем приложения.

Поле **Sequence Number** (16 бит). Поле порядкового номера сообщения, значение которого увеличивается на единицу при передаче очередного сообщения RTP. Наличие порядкового номера позволяет обнаружить потерю сообщений или определить порядок следования сообщений, имеющих одинаковую метку времени.

Примечание

Несколько последовательных сообщений могут иметь одну и ту же метку времени, если логически они порождены в один и тот же момент, как, например, сообщения, переносящие порции информации, принадлежащие одному и тому же видеокадру. Для обеспечения противодействия возможным атакам на приемники каждый источник начинает нумеровать пакеты с произвольного случайно выбранного номера.

Поле **Timestamp** (32 бита) предназначено для размещения метки времени. Это поле содержит линейно и монотонно возрастающее значение, соответствующее моменту времени, в который был создан первый байт данных полезной нагрузки. Точность, с которой формируется это значение, должна быть достаточной для обеспечения временной синхронизации для всех типов полезной нагрузки. Значение поля **Timestamp** определяется по локальным часам отправителя, причем стартовое значение этих локальных часов для каждой сессии одного и того же устройства определяется случайным образом.

В поле идентификатора источника синхронизации **Synchronization Source (SSRC) Identifier** помещается случайное число, предназначенное для однозначной идентификации этого источника. Уникальность этого идентификатора в пределах сессии обеспечивает возможность выявления и устранения петель, возникающих при передаче потокового трафика в сети.

Смеситель, используемый в качестве источника синхронизации, может объединять в одно сообщение протокола RTP порции данных, полученных от нескольких источников, и, при необходимости, изменять формат представления данных. Предположим, что новая система хочет принять участие в сеансе аудиоконференции, но ее канал до сети не имеет достаточной пропускной способности. В этом случае смеситель, который получает информационные потоки данного сеанса, объединяет их в один, который

и передает новому члену сеанса, используя при этом более экономичный способ кодирования. Заголовок сообщения протокола RTP, генерируемого смесителем, включает идентификаторы всех передатчиков, чьи данные присутствуют в этом сообщении.

В поле **Contributing Source (CSRC) Identifier** (источник входящих данных), занимающем 32 бита в заголовке сообщения протокола RTP, помещается идентификатор источника данных, представленных в данном сообщении. Количество таких полей в сообщении протокола RTP может быть от 0 до 15.

Сообщения протокола RTCP

Сообщения протокола RTCP предназначены для управления компонентами, выполняющими формирование и передачу группового трафика с использованием протокола RTP. Для решения этой задачи используются следующие типы сообщений протокола RTCP:

- SR (Sender Report) — отчет источника (передатчика);
- RR (Receiver Report) — отчет приемника;
- SDES (Source Description) — описание источника;
- BYE — выход из сессии;
- APP (Application) — функции приложения.

Сообщения SR формируются активными участниками сессии для передачи диагностической информации о передаваемых и принимаемых порциях потоковых данных.

Сообщения RR формируются пассивными участниками сессии для передачи диагностической информации о принимаемых порциях потоковых данных. Сообщения SDES содержат описания источников потоковых данных.

Сообщение BYE формируется одним участником сессии для информирования остальных участников о том, что данный узел перестал участвовать в сессии.

Сообщение APP предназначено для выполнения специфических функций конкретных приложений.

Каждое сообщение протокола RTCP представляет собой информационный блок, имеющий стандартный заголовок, похожий на заголовок сообщения протокола RTP. Особенность протокола RTCP заключается в том, что несколько различных типов сообщений этого протокола могут объединяться в одно композиционное сообщение, передаваемое единым блоком без использования каких-либо внутренних разделителей. При формировании композитного сообщения протокола RTCP должны быть соблюдены некоторые специальные правила. Сообщение BYE, например, может появиться только в завершающей части композитного сообщения, а сообщение SDES,

содержащее каноническое имя источника потокового трафика, должно включаться в композитное сообщение так часто, чтобы обеспечить достаточно быструю взаимную синхронизацию мультимедийных потоков.

Сообщения Sender Report

Формат сообщений Sender Report протокола RTCP представлен на рис. 5.9.

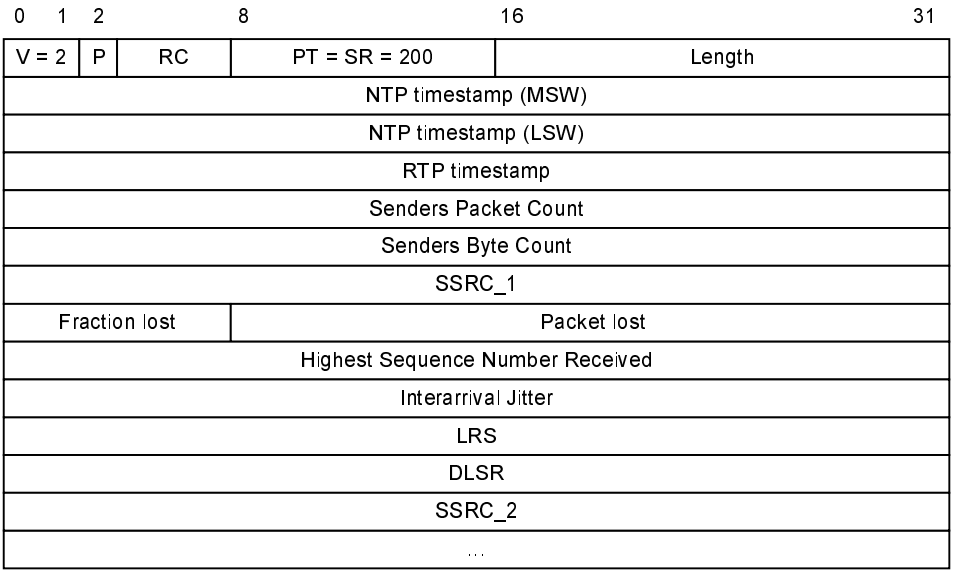


Рис. 5.9. Формат сообщения Sender Report протокола RTCP

В поле **V** (Version) (2 бита) размещается признак номера версии протокола RTCP (текущая версия 2).

В поле **P** (Padding) (1 бит) размещается признак, который сигнализирует о наличии заполняющих байтов в конце полезной нагрузки. Если в сообщении Sender Report бит **P** установлен в 1, последний байт этого сообщения содержит число заполняющих байтов.

В поле **RC** (Reception Report Count) (5 бит) сообщения Sender Report протокола RTCP находится количество блоков отчетов различных приемников, содержащихся в данном сообщении.

Поле **PT** (8 бит) содержит признак сообщения Sender Report — константу "200".

В поле **Length** (16 бит) размещается длина сообщения Sender Report, выраженная в 32-битных словах.

В поле **SSRC** (16 бит) сообщения Sender Report содержится идентификатор источника синхронизации, сформировавшего данное сообщение.

Поле **NTP timestamp** (64 бита) предназначено для размещения значения абсолютного времени в момент передачи настоящего сообщения. Это значение может быть использовано для определения задержки распространения данных по сети.

Поле **RTP timestamp** (32 бита) предназначено для размещения значения относительного времени в момент передачи настоящего сообщения. Значение, находящееся в данном поле соответствует тому же моменту времени, что и значение поля **NTP timestamp,** однако формируется с использованием случайного начального смещения.

В поле **Senders Packet Count** (32 бита) размещается общее количество сообщений протокола RTP, сформированных данным источником с момента получения текущего значения идентификатора SSRC.

В поле **Senders Byte Count** (32 бита) размещается общее количество байтов полезной нагрузки сообщений протокола RTP, сформированных данным источником с момента получения текущего значения идентификатора SSRC.

Следующая секция заголовка сообщения Sender Report содержит 24-байтовые блоки диагностической информации источников потоковых данных.

В поле **SSRC_n** (16 бит) сообщения Sender Report содержится идентификатор источника потоковых данных.

В поле **Fraction lost** (8 бит) помещается доля потерянных порций данного источника, вычисляемая относительно общего числа переданной им информации.

Поле **Packets lost** (24 бита) предназначено для размещения общего количества потерянных сообщений данного источника.

В младших 16 разрядах поля **Highest Sequence Number Received** (32 бита) помещается максимальный номер сообщения, принятого от данного источника, в старших разрядах данного поля помещается расширение этого значения соответствующим значением счетчика циклов.

В поле **Interarrival Jitter** (16 бит) сообщения Sender Report помещается среднее абсолютное значение изменения времени распространения информации данного источника.

В поле **LSR** (32 бита) данного сообщения помещается старшая часть последнего значения **NTP timestamp**, принятого от данного источника в его сообщении Sender Report.

В поле **DLSR** (32 бита) сообщения Sender Report размещается значение задержки с момента времени получения сообщения от данного источника до момента времени формирования настоящего сообщения.

Сообщения Receiver Report

Формат сообщений Receiver Report протокола RTCP представлен на рис. 5.10.

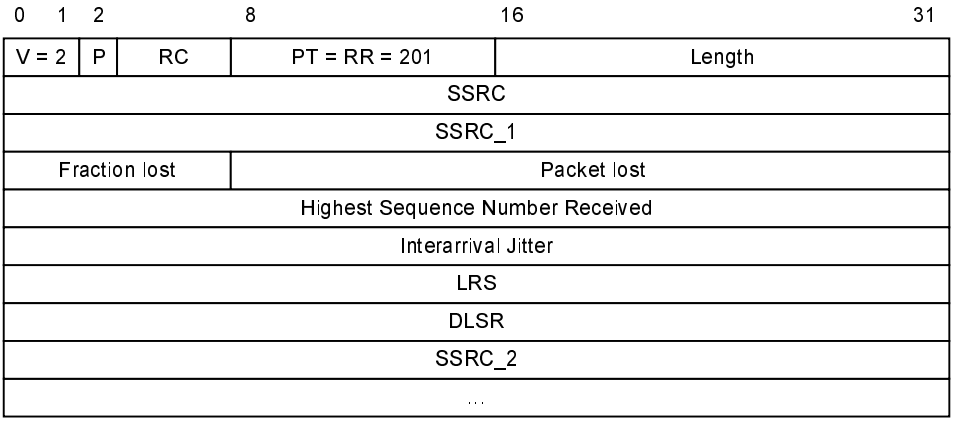


Рис. 5.10. Формат сообщения Receiver report протокола RTCP

В поле **V** (Version) (2 бита) размещается признак номера версии протокола RTCP (текущая версия 2).

В поле **P** (Padding) (1 бит) размещается признак, который сигнализирует о наличии заполняющих байтов в конце полезной нагрузки. Если в сообщении Receiver Report бит **P** установлен в 1, последний байт этого сообщения содержит число заполняющих байтов.

В поле **RC** (Reception Report Count) (5 бит) —сообщения Receiver Report протокола RTCP находится количество блоков отчетов различных приемников, содержащихся в данном сообщении.

Поле **PT** (8 бит) содержит признак сообщения Receiver Report — константу "201".

В поле **Length** (16 бит) размещается длина сообщения Receiver Report, выраженная в 32-битных словах.

В поле **SSRC** (16 бит) сообщения Receiver Report содержится идентификатор источника синхронизации, сформировавшего данное сообщение.

Следующая секция заголовка сообщения Receiver Report содержит 24-байтовые блоки диагностической информации источников потоковых данных.

В поле **SSRC_n** (16 бит) сообщения Receiver Report содержится идентификатор источника потоковых данных.

В поле **Fraction lost** (8 бит) помещается доля потерянных порций данного источника, вычисляемая относительно общего числа переданной им информации.

Поле **Packets lost** (24 бита) предназначено для размещения общего количества потерянных сообщений данного источника.

В младших 16 разрядах поля **Highest Sequence Number Received** (32 бит) помещается максимальный номер сообщения, принятого от данного источника. В старших разрядах данного поля помещается расширение этого значения соответствующим значением счетчика циклов.

В поле **Interarrival Jitter** (16 бит) сообщения Receiver Report помещается среднее абсолютное значение изменения времени распространения информации данного источника.

В поле **LSR** (32 бита) данного сообщения помещается старшая часть последнего значения **NTP timestamp**, принятого от данного источника в его сообщении Receiver Report.

В поле **DLSR** (32 бита) сообщения Receiver Report размещается значение задержки с момента времени получения сообщения от данного источника до момента времени формирования настоящего сообщения.

Сообщения Source Description

Формат сообщений Source Description протокола RTCP представлен на рис. 5.11.

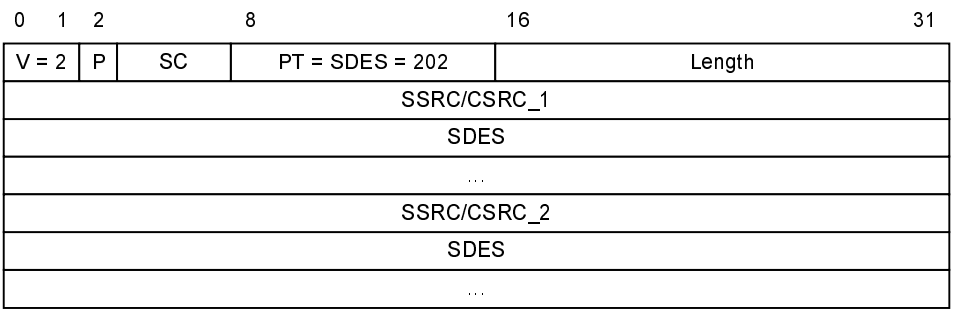


Рис. 5.11. Формат сообщения Source Description протокола RTCP

В поле **V** (Version) (2 бита) размещается признак номера версии протокола RTCP (текущая версия 2).

В поле **P** (Padding) (1 бит) размещается признак, который сигнализирует о наличии заполняющих байтов в конце полезной нагрузки. Если в сообщении Source Description бит **P** установлен в 1, последний байт этого сообщения содержит число заполняющих байтов.

В поле **SC** (Source Count) (5 бит) сообщения Source Description протокола RTCP находится количество блоков информации различных приемников, содержащихся в данном сообщении.

Поле **PT** (8 бит) содержит признак сообщения Source Description — константу "202".

В поле **Length** (16 бит) размещается длина сообщения Source Description выраженная в 32-битных словах.

Следующая секция заголовка сообщения Source Description содержит блоки диагностической информации источников потоковых данных.

В поле **SSRC/CSRC_n** (16 бит) сообщения Source Description содержится идентификатор источника потоковых данных.

Поле **SDES** имеет переменную длину и предназначено для размещения идентифицирующей информации источника потоковых данных.

В табл. 5.2 приведен перечень параметров, используемых для описания источников в сообщении Source Description.

Таблица 5.2. *Параметры описания источника сообщения Source Description*

Признак	Обозначение	Назначение
1	CNAME	Канонический идентификатор источника
2	NAME	Имя пользователя
3	EMAIL	Электронная почта
4	PHONE	Телефонный номер
5	LOC	Географическое размещение
6	TOOL	Тип используемого приложения
7	NOTE	Текущее состояние источника
8	PRIV	Расширение описания

Сообщения BYE

Формат сообщений BYE протокола RTCP представлен на рис. 5.12.

В поле **V** (Version) (2 бита) размещается признак номера версии протокола RTCP (текущая версия 2).

В поле **P** (Padding) (1 бит) размещается признак, который сигнализирует о наличии заполняющих байтов в конце полезной нагрузки. Если в сообщении BYE бит **P** установлен в 1, последний байт этого сообщения содержит число заполняющих байтов.

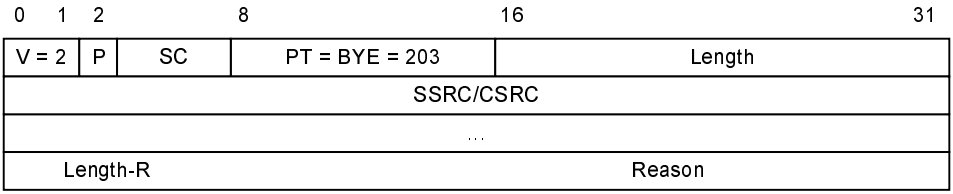


Рис. 5.12. Формат сообщения BYE протокола RTCP

В поле **SC** (Source Count) (5 бит) сообщения BYE протокола RTCP находится количество блоков информации различных приемников, содержащихся в данном сообщении.

Поле **PT** (8 бит) содержит признак сообщения BYE — константу "203".

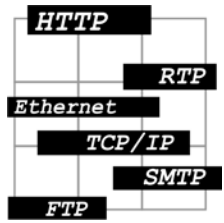
В поле **Length** (16 бит) размещается длина сообщения BYE, выраженная в 32-битных словах.

В поле **SSRC/CSRC** (16 бит) сообщения BYE содержится идентификатор источника потоковых данных.

В поле **Length-R** (8 бит) помещается длина поля переменной длины, предназначенного для размещения причины выхода данного источника из сеанса.

В поле **Reason** сообщения BYE протокола RTCP помещается причина выхода из сеанса данного источника.

Глава 6



Протоколы группового взаимодействия в сети Интернет

Особенности взаимодействия с использованием Multicast-адресов

В предыдущих главах было отмечено, что для обеспечения передачи данных в сети Интернет может быть использован один из трех основных режимов информационного обмена:

- Broadcast;
- Unicast;
- Multicast.

Режим Broadcast (безадресная широковещательная передача) предназначен для передачи данных всем абонентам сети и используется в основном в технологических целях — передача информации о маршрутах, определение адреса сервера и т. д. Распространение сообщений этого типа обычно ограничено локальным сегментом сети, поскольку в противном случае сеть Интернет была бы просто парализована сообщениями типа Broadcast.

Режим Unicast (одноадресная передача) используется в том случае, когда адрес назначения точно известен источнику информации. В этом случае в информационном обмене участвуют только два узла сети (если не считать маршрутизаторов) — передатчик и приемник. Этот режим информационного обмена традиционно является основным, на его долю приходится от 90 до 99% всего информационного обмена в сети TCP/IP.

Использование режима Multicast (групповая многоадресная передача) в вычислительных сетях целесообразно в тех случаях, когда передаваемая

информация предназначена для группы приемников. Такой режим может быть эффективно использован при организации группового вещания по принципу "один для многих". Использование режима многоадресной передачи в данном случае позволяет значительно более экономно использовать пропускную способность вычислительной сети, поскольку передается только одна порция информации для нескольких приемников. Это преимущество является особенно существенным в тех случаях, когда объем передаваемого трафика является большим (видеоконференции), или когда тип вещания не предполагает явного указания адреса абонента (видео- и аудиотрансляции). На рис. 6.1 приведена схема организации информационного обмена с использованием групповых адресов. В представленном примере несколько станций, расположенных в сетях N.N.N.0/24 и J.N.N.0/24, оказались заинтересованы в приеме трафика группы 224.0.0.18. В данном случае трафик для этой группы поставляется через специальные групповые маршрутизаторы ГМ1—ГМ3. Групповые маршрутизаторы используют специальные алгоритмы маршрутизации для того, чтобы определить необходимость ретрансляции группового трафика через те или иные свои интерфейсы. Поскольку, в отличие от источника Unicast, источник группового трафика формирует один поток данных, независимо от наличия или количества приемников, пропускная способность его канала не оказывает практически никакого влияния на темп информационного обмена. Увеличение числа приемников группового трафика может привести только к подключению новых ветвей сети, по которым будет распространяться этот трафик. Важно то, что интенсивность информационного потока на существующих ветвях при этом никак не меняется. На приведенном примере маршрутизатор ГМ3 формирует две копии блоков данных, получаемых от источника 224.0.0.18, поскольку потребители этого трафика подключены к двум его интерфейсам. Групповой маршрутизатор обязан передавать групповой трафик до тех пор, пока у него есть хотя бы один потребитель этого трафика. Один групповой маршрутизатор может быть использован для обеспечения информационного обмена нескольких Multicast-групп одновременно. Маршрутизаторы ГМ2 и ГМ3 обслуживают две группы, а ГМ1 — только одну, поскольку у последнего нет потребителей трафика 224.0.1.128.

Основным недостатком, свойственным режиму Multicast, является то, что при его использовании, естественно, возникают существенные затруднения в организации гарантированной доставки сообщений и управления темпом информационного обмена. Эти недостатки, однако, могут быть частично скомпенсированы благодаря использованию специальных режимов информационного обмена.

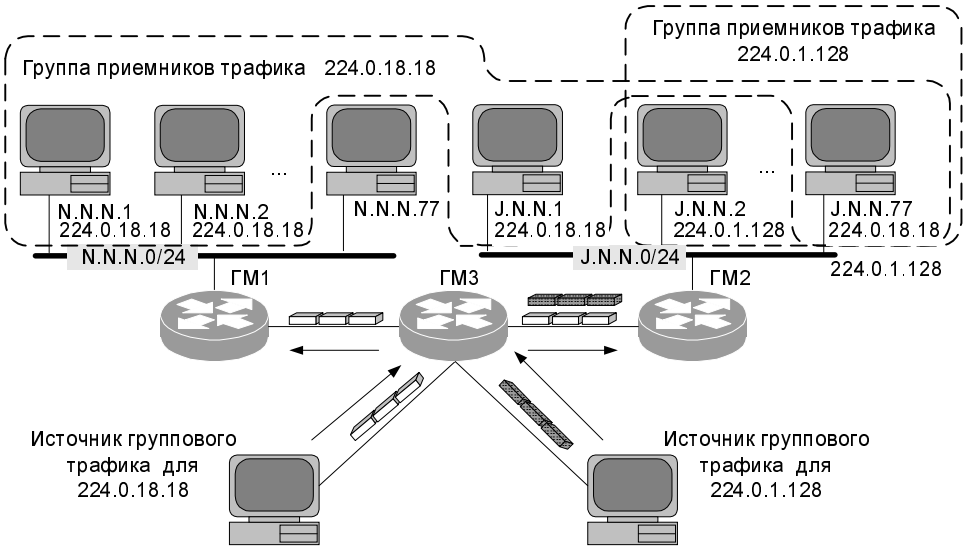


Рис. 6.1. Схема организации информационного обмена с использованием групповых адресов

Диапазоны групповых адресов TCP/IP

Для обеспечения режима групповой адресации в сети Интернет используется специальный класс адресов — класс "D". Групповые адреса находятся в диапазоне от 224.0.0.1 до 239.255.255.255. В отличие от адресов Unicast, групповые адреса назначаются не узлам, а группам узлов. Один и тот же групповой адрес имеют все узлы, входящие в такую группу.

Примечание

Обратите внимание, что передатчик — формирователь группового трафика — не обязательно должен являться членом той группы, для которой он его формирует. Групповые IP адреса могут быть использованы только в качестве адресов назначения.

Например, узлам P, Q и R назначен один групповой адрес — X. Тогда для того, чтобы отправить дейтаграмму всем трем узлам (P, Q и R), достаточно отправить ее по адресу X.

Обращение к группе узлов по одному IP-адресу можно использовать, например, при репликации баз данных, при организации конференций, параллельных вычислений или групповых компьютерных игр.

Для обеспечения организованного использования адресного пространства весь диапазон групповых адресов административно разделен организацией IANA на несколько зон адресов специального назначения.

В табл. 6.1 приведены диапазоны функциональных блоков групповых адресов, зарезервированных организацией IANA.

Таблица 6.1. *Адреса функциональных блоков групповых адресов*

Диапазон	Наименование	Назначение
224.0.0.0 — 224.0.0.255	Local Network Control Block	Адреса компонентов локальной сети
224.0.1.0 — 224.0.1.255	Internetwork Control Block	Адреса компонентов глобальной сети
224.0.2.0 — 224.0.255.0	AD-HOC Block	Блок специальных адресов
224.1.0.0 — 224.1.255.255	ST Multicast Groups [RFC-1190, KS14]	Блок адресов Internet Stream Protocol
224.2.0.0 — 224.2.255.255	SDP/SAP Block	Блок динамически назначаемых и технологических адресов
224.3.0.0 — 224.251.255.255	Reserved	Зарезервировано организацией IANA
225.0.0.0 — 231.255.255.255	Reserved	Зарезервировано организацией IANA
232.0.0.0 — 232.255.255.255	Source Specific Multicast	Блок адресов групп с ограниченным перечнем источников
233.0.0.0 — 233.255.255.255	GLOP Block — [RFC-3180]	Блок адресов, статически назначаемых по схеме GLOP
234.0.0.0 — 238.255.255.255	Reserved [IANA]	Зарезервировано IANA
239.0.0.0 — 239.255.255.255	Administratively Scoped [IANA, RFC-2365]	Блок адресов для административно ограниченных областей

Примечание

Для обозначения границ зон иногда применяется укороченная схема сетевой нотации. Так, например, диапазон групповых адресов 224.0.0.0 — 224.0.0.255 может быть представлен выражением 224.0.0.0/24.

Локальные групповые адреса протокола IP используются, как правило, для технологических целей. Различные протоколы Интернета используют адреса этого диапазона для обеспечения информационного обмена между своими компонентами вместо адресов Broadcast, что обеспечивает более эффективное использование ресурсов локальной сети. Например, адрес 224.0.0.9

используется маршрутизаторами протокола RIP II для передачи обновлений маршрутной информации.

Часть адресов из диапазона 224.0.0.0/24 относится к группе широко известных (well known), назначение некоторых из этих адресов приведено в табл. 6.2. Распространение IP-дейтаграмм с групповыми адресами локального диапазона ограничивается локальным сегментом сети, независимо от установленного значения TTL (нормальным значением TTL для дейтаграмм такого типа является 1).

В табл. 6.2 приведены групповые адреса некоторых компонентов локальных сетей.

Таблица 6.2. *Групповые адреса ряда компонентов локальных сетей*

Начальный адрес	Назначение
224.0.0.1	Узлы, которые используют групповой режим адресации
224.0.0.2	Маршрутизаторы, которые обеспечивают групповой режим адресации
224.0.0.3	Маршрутизаторы DVRMP
224.0.0.5	Маршрутизаторы OSPF
224.0.0.6	Назначенные (DR) маршрутизаторы OSPF
224.0.0.9	Маршрутизаторы RIP II
224.0.0.13	Маршрутизаторы PIM

Групповые адреса диапазона 224.0.1.0/24 предназначены для организации Multicast-групп, компонентами которых являются узлы глобальной сети. Распространение по сети Интернет трафика, направленного по этим адресам, может быть ограничено только временем существования дейтаграммы в сети (TTL).

Назначение адресов из этого диапазона производится статически, путем согласования и выпуска необходимых организационных документов. Глобальные групповые адреса могут быть использованы как для организации сетевого вещания, так и для корпоративных технологических целей. Например, адрес 224.0.1.40 зарезервирован корпорацией Cisco Systems для обеспечения поиска групповых маршрутизаторов (cisco-rp-discovery), а право использования группового вещания по адресу 224.0.1.128 закреплено за корпорацией CNN.

В специальный блок выделены традиционно определенные групповые адреса, принцип использования которых не совпадает с принципами использования адресов первой и второй групп. Групповые адреса AD-HOC Block применяются для обеспечения приложений, оперирующих с небольшим

количеством Multicast-групп. Назначение адресов этого диапазона, как правило, производится без санкции организации IANA.

В блок SDP/SAP (Session Description Protocol/Session Announcement Protocol) Block входят как технологические адреса этих протоколов, так и адреса, присваиваемые динамически источникам группового трафика. Протокол объявления сессий — SAP применяется источником группового трафика для того, чтобы занести свою сессию в список анонсируемых сессий. Благодаря тому, что информация об анонсируемых сессиях свободно распространяется по сети, любой желающий может получить перечень зарегистрированных активных источников при помощи протокола описания сессий — SDP.

Блок адресов групп с ограниченным перечнем источников предназначен для использования группами, которые применяют расширенную процедуру группового информационного взаимодействия. Приемники группового трафика — члены таких групп, в частности, могут явно определять перечень допустимых источников для поставки им группового трафика.

Адреса блока GLOP Block применяются при использовании специальной практики статического назначения групповых адресов. Уникальность групповых адресов в этом случае обеспечивается без участия организации IANA благодаря тому, что в групповой адрес включается уникальный номер автономной системы источника группового трафика. Номер автономной системы размещается во втором и третьем байте адреса. Младший байт при этом используется для назначения различным группам внутри системы.

Последний блок образуют динамически и статически назначаемые групповые адреса, предназначенные для использования внутри замкнутых областей.

Групповые адреса, которые имеют глобальное значение, присваиваются группам централизовано с использованием специальных динамических процедур. В соответствии с этими процедурами пространство групповых адресов распределяется по иерархическому принципу, когда домен нижнего уровня арендует у домена верхнего уровня блок адресов на ограниченный период времени. Однако вследствие глобального характера сети TCP/IP практическая реализация таких процедур представляется весьма затруднительной для практической реализации.

Передача группового трафика на канальном уровне

Назначение группового адреса узлу можно рассматривать как обобщение его персонального IP-адреса. В то время как индивидуальный IP-адрес узла отображается в адрес канального уровня этого узла, групповой Multicast IP-адрес должен отображаться в группу адресов канального уровня. Групповая адресация в сети с множественным доступом Ethernet — это, пожалуй, самый естественный способ построения Multicast IP-адресации при реализации доставки на канальном уровне.

Специальный диапазон адресов канального уровня сети Ethernet предназначен для формирования групповых Multicast-адресов. Групповые Ethernet-адреса канального уровня, предназначенные для отображения групповых адресов сети Интернет, занимают диапазон с 01-00-5E-00-00-00 по 01-00-5E-7F-FF-FF.

IP-адрес группы узлов ассоциируется с Ethernet Multicast-адресом следующим образом: 23 младших бита IP-адреса группы помещаются в 23 младших бита группового Ethernet-адреса. К сожалению, из-за того, что диапазон выделенных адресов канального уровня существенно меньше диапазона адресов сетевого уровня, такое отображение не может быть однозначным. Одному Multicast Ethernet-адресу можно поставить в соответствие 31 IP-группу.

Например, IP-адрес 224.1.10.111 соотносится с адресом 01-00-5E-01-0A-6F. Аналогичным образом с этим адресом канального уровня могут быть соотнесены групповые адреса 224.129.10.111, 225.1.10.111 и тому подобные.

Определение состава группы

Основным понятием в групповой адресации является понятие Multicast-группы узлов.

В дальнейшем *Multicast-группой* будет называться группа узлов, имеющих одинаковый IP-адрес класса D.

Multicast-группы не являются закрытыми, т. е. обращение к Multicast-группе узлов может быть сделано и не членом группы. Членство узлов в любой Multicast-группе — динамическое, т. е. узлы могут включаться и выходить из нее по собственному желанию. Multicast-группа не накладывает никаких ограничений на количество и расположение ее членов. Узел может быть членом одной или нескольких Multicast-групп одновременно.

Multicast-группы бывают двух типов — динамические и постоянные. Постоянные группы имеют фиксированные IP-адреса, и количество членов в такой Multicast-группе может быть нулевым. IP-адреса, которые не зарезервированы для использования постоянными Multicast-группами, могут использоваться динамическими группами. Динамическая Multicast-группа существует до тех пор, пока существует хотя бы один ее член.

Multicast-группа рассылки определяет сетевую группу, которая представляет собой набор сетей, содержащих членов данной Multicast-группы. Пакет, адресованный такой группе, дублируется в каждую сеть, содержащую членов Multicast-группы, затем внутри каждой из этих сетей он передается ее членам на канальном уровне.

Членом группы может стать любой узел H, который подключен к сети, после того как он выполнит процедуру установления своего членства в группе GA. Обычно процедура установления членства в группе заключается в передаче

групповому маршрутизатору ГМ запроса с указанием адреса группы GA. В том случае, если маршрутизатор уже принимает трафик, адресованный членам этой группы, никаких дополнительных действий не принимается. Такая ситуация возникает тогда, когда в том сегменте сети, в котором находится узел Н, уже имеются члены группы GA. В том случае, если узел Н является первым или единственным членом группы GA, маршрутизатор должен выполнить действия, которые обеспечивают передачу ему трафика, адресованного членам этой группы. Для того чтобы определить оптимальный путь в сети, по которому маршрутизатору ГМ должен быть передан трафик группы GA, используются специальные алгоритмы маршрутизации.

Маршрутизация и доставка группового трафика

При передаче данных с использованием групповой адресации основные усилия по организации информационного взаимодействия возлагаются на принимающую сторону. Передатчик — формирователь группового трафика не знает не только адресов станций, которые принимают его данные, но и то, сколько таких приемников всего и есть ли они вообще. Основные действия по доставке группового трафика до конечного абонента выполняются активными компонентами сети — групповыми маршрутизаторами.

При информационном обмене с использованием групповых адресов маршрутизатор должен поддерживать следующие дополнительные компоненты протокола IP.

1. Таблицу маршрутизации Multicast-группы, содержащую информацию о ближайшем шлюзе для каждой сети (она должна присутствовать в той или иной форме в работе любого из маршрутизаторов, вне зависимости от того, работает он с Multicast-дейтаграммой или нет).
2. Таблицу членов Multicast-группы сети, содержащую запись о каждой Multicast-группе в данной сети и о других сетях, которые содержат членов этой же группы (по сути, маршрутизация осуществляется на основании этой таблицы).
3. Таблицу членов Multicast-группы узла. Таблица содержит записи, по одной для каждой Multicast-группы, члены которой расположены в сети, непосредственно подсоединенной к данному узлу.
4. Источник отправляет многоадресную дейтаграмму, используя обычный способ отправки дейтаграмм. Multicast-дейтаграмма протокола IP, адресованная группе узлов, сначала отправляется на шлюз, присоединенный к сети отправителя. Шлюз по таблице членов Multicast-группы определяет адреса сетей, куда следует переправить эту Multicast-дейтаграмму. Если требуемая сеть подсоединена к шлюзу напрямую, он передает в эту сеть копию дейтаграммы, используя групповой адрес канального уровня. В том случае, если искомая сеть не входит в состав непосредственно

подключенных сетей для данного маршрутизатора, дейтаграмма снабжается заголовком, обеспечивающим ее доставку ближайшему к искомой сети групповому маршрутизатору.

Таблицы членов групп могут меняться. Для динамического формирования этих таблиц используется протокол IGMP (Internet Group Management Protocol — протокол управления группами Интернета).

Протокол IGMP

Маршрутизатор узнает о том, есть ли в подключенных к нему сегментах сети потребители конкретного группового трафика, с помощью специальных сообщений протокола IGMP (Internet Group Management Protocol), формируемых потенциальными потребителями группового трафика. В сетях, используемых для передачи группового трафика, протокол IGMP играет практически такую же роль, какую играет протокол ICMP в обычной сети Интернет — с учетом специфики соответствующих сетей. Основной функцией протокола IGMP является определение процедуры динамического членства в группе.

Процедуры протокола IGMP выполняются двумя типами объектов — маршрутизатором и узлом. Маршрутизатор выполняет функцию доставки группового трафика, узел является потребителем этого трафика. Предполагается, что узел исходно обладает информацией о том, какой тип группового трафика его интересует. Такая информация представляется в виде группового адреса назначения (Group Destination Address — GDA). Узел получает информацию об активных GDA, используя процедуры, рассмотрение которых выходит за рамки данной главы. Свою потребность в приеме группового трафика узел может обозначить по собственной инициативе либо в ответ на соответствующий запрос, сформированный групповым маршрутизатором. Как было отмечено выше, один узел может быть членом нескольких Multicast-групп одновременно. Доступ к источникам группового трафика при этом может выполняться с использованием услуг нескольких групповых маршрутизаторов, подключенных к сегменту сети, содержащему узел — приемник группового трафика. Вполне может оказаться, что несколько групповых маршрутизаторов способны обеспечить доставку трафика некоторой группы на конкурсной основе. На рис. 6.2 представлен вариант сети, в которой для доставки группового трафика могут быть использованы групповые маршрутизаторы ГМ1 и ГМ2. Для того чтобы избежать коллизий и повысить эффективность использования сетевых ресурсов, в подобных ситуациях используется специальная процедура определения активного группового маршрутизатора. Поскольку активный маршрутизатор формирует запросы в адрес членов Multicast-групп, в терминах протокола IGMP такой маршрутизатор называется Querier (Проситель).

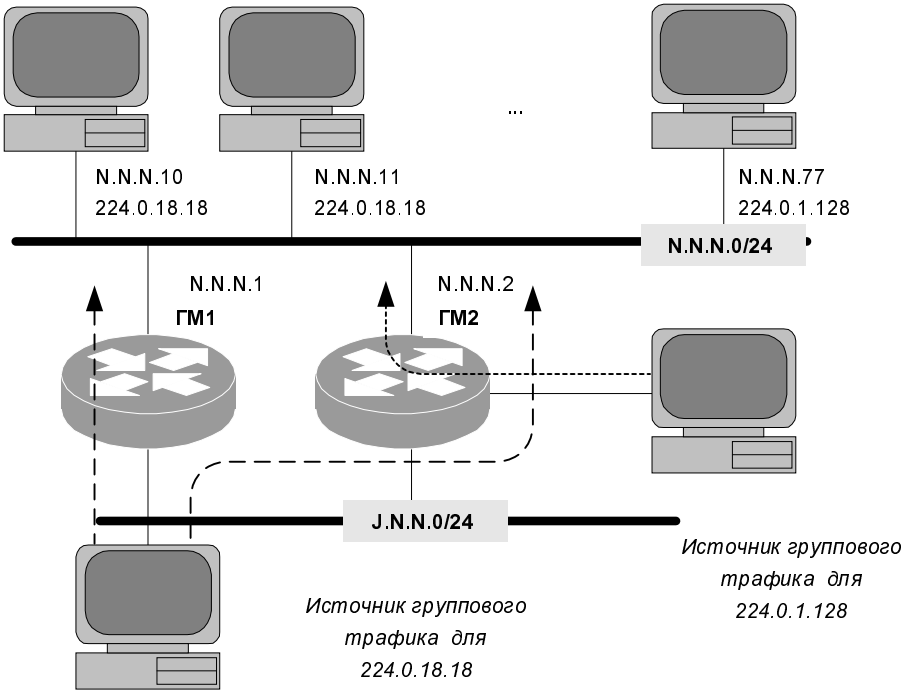


Рис. 6.2. Использование альтернативных групповых маршрутизаторов для доставки группового трафика

По большому счету, для конечного узла не имеет значения, какой именно из альтернативных маршрутизаторов будет использован для доступа к некоторому источнику. Поэтому определение активного маршрутизатора выполняется самими групповыми маршрутизаторами следующим образом:

- в момент первоначального включения каждый групповой маршрутизатор выполняет функции активного, рассылая запросы потребителям группового трафика;
- маршрутизатор прослушивает аналогичные запросы, передаваемые другими групповыми маршрутизаторами, подключенными к сегменту локальной сети;
- если маршрутизатор обнаруживает, что аналогичные запросы передает еще один маршрутизатор, имеющий меньший IP-адрес в данном сегменте, он автоматически переходит в пассивное состояние.

Таким образом, для сети, схема которой приведена на рис. 6.2, в соответствии с правилами определения активного маршрутизатора, активным маршрутизатором для группы 224.0.18.18 должен стать маршрутизатор ГМ1.

Активный групповой маршрутизатор начинает периодически передавать запросы (Query) потенциальным потребителям группового трафика и обрабатывать полученные от них ответы (Report). Для определения потребности в групповом трафике у абонентов подключенного к нему сегмента сети маршрутизатор формирует запросы двух видов:

- General Membership Query — общий запрос;
- Group-Specific Membership Query — специальный запрос.

Запрос General Membership Query предназначен для определения потребности в групповом трафике без указания адреса конкретной группы. На такой запрос должны отвечать все узлы — потенциальные потребители группового трафика с указанием группового адреса GDA. Такой запрос групповой маршрутизатор отправляет, используя в качестве адреса назначения групповой адрес 224.0.0.1 ("всем узлам, принимающим групповой трафик").

Запрос Group-Specific Membership Query предназначен для определения потребности в групповом трафике конкретной группы GDA. На такой запрос должны отвечать только те узлы, которые заинтересованы в приеме трафика именно этой группы.

Пакеты запросов General Membership Query и Group-Specific Membership Query отличаются только содержанием поля **Group Address**, а именно у общего запроса General Membership Query содержимое этого поля соответствует IP-адресу 0.0.0.0, а у специального запроса Group-Specific Membership Query в этом поле размещается конкретное значение адреса группы.

Теоретически при получении такого запроса все узлы, заинтересованные в приеме группового трафика, должны сформировать ответные сообщения (Response). В этих ответных сообщениях должны быть перечислены адреса групп, трафик которых узел желает принимать. При передаче ответного сообщения узел, заинтересованный в приеме группового трафика, использует в качестве адреса назначения групповой адрес 224.0.0.2 ("всем маршрутизаторам, передающим групповой трафик").

Следовательно, в том случае, если в сегменте локальной сети имеется *N* приемников трафика некоторой группы G, групповой маршрутизатор получит *N* ответов типа: "Да! Я хочу G!". Такая схема не обеспечивает эффективного использования пропускной способности сегмента сети по двум причинам. Во-первых, несколько одновременно сформированных ответов могут привести к возникновению коллизий, и, как следствие, к многочисленным повторным передачам. Во-вторых, групповому маршрутизатору важно знать лишь то, что в подключенном к нему сегменте локальной сети количество потребителей трафика группы G больше 0. Конкретное количество потребителей не интересует маршрутизатор потому, что он будет направлять в сегмент только одну копию каждого принятого от группы G блока данных.

Для обеспечения более эффективного использования пропускной способности сети при выполнении процедуры протокола IGMP может быть использован специальный алгоритм подавления дублирующих ответов узлов на запросы активного группового маршрутизатора.

Для реализации такой процедуры активный маршрутизатор снабжает каждый свой запрос параметром Max Response Time (Максимальное время для ответа). Получив такой запрос, узел, заинтересованный в приеме группового трафика, отвечает на него с задержкой, случайным образом распределенной в диапазоне (0, Max Response Time). В том случае, если до истечения вычисленного временного интервала узел услышит ответ "Да! Я хочу G!", переданный каким-нибудь другим узлом, данный узел не отвечает совсем.

В том случае, если узел заинтересован в приеме группового трафика от нескольких различных групповых адресов назначения GDA, он будет пытаться сформировать ответ по каждой из групп, членом которой он является или хочет стать. В этом случае работает та же процедура подавления дублирующих ответов, с тем отличием, что для формирования ответа по каждой из интересующих групп узел будет использовать отдельный независимый таймер.

Формат сообщения протокола IGMP

В настоящее время используются три версии протокола IGMP. Отличия между различными версиями протокола заключаются в номенклатуре и способе обработки формируемых сообщений. Далее будут рассматриваться сообщения и алгоритмы их обработки, принятые во второй версии протокола IGMP v2.

Все сообщения протокола IGMP инкапсулируются в дейтаграммы IP с полем "тип протокола", равным 2. Сообщения протокола IGMP могут иметь переменную длину и снабжаются специальным заголовком. Формат заголовка сообщения IGMP приведен на рис. 6.3.

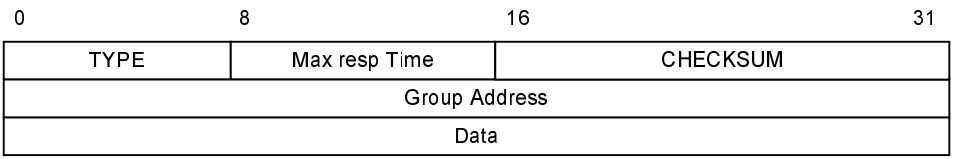


Рис. 6.3. Формат заголовка сообщения протокола IGMP

Как уже было отмечено, основными сообщениями протокола IGMP являются запросы (Query), которые передает маршрутизатор, и ответы (Report), формируемые узлами. Типы сообщений протокола IGMP приведены в табл. 6.3.

Таблица 6.3. *Типы сообщений протокола IGMP*

Значение поля Type	Источник	Тип сообщения IGMP	Назначение сообщения
0x11	Маршрутизатор	Membership Query	Запрос к членам группы
0x12	Узел	Membership Report IGMP v1	Ответ членов группы (версия 1)
0x16	Узел	Membership Report IGMP v2	Ответ членов группы (версия 2)
0x17	Узел	Leave Group	Отключение от группы

Код сообщения второй версии протокола IGMP сформирован таким образом, чтобы обеспечить совместимость форматов сообщений с первой версией протокола. В первой версии протокола IGMP используются только два типа сообщения Membership Query (код 1) и Membership Report IGMP (код 2). В первом полубайте первого байта сообщения протокола IGMP v1 размещается номер версии протокола — 1. Поэтому коды сообщений Membership Query и Membership Report протокола IGMP в первой версии протокола равны 11 и 12 соответственно.

Содержимое поля **Max Response Time** формируется только в сообщениях Membership Query. В это поле групповой маршрутизатор, использующий вторую версию протокола IGMP, помещает максимальную величину временного интервала задержки ответа на данное сообщение. Значение задержки выражается в десятых долях секунды. Поскольку маршрутизатор, применяющий протокол IGMP v1, не формирует поле **Max Response Time** (точнее, это поле при использовании IGMP v1 формируется равным 0), считается, что по умолчанию значение этого поля равно 100 (10 секунд). Во всех остальных сообщениях протокола IGMP поле **Max Response Time** не формируется и не анализируется.

Контрольная сумма сообщения протокола IGMP предназначена для обеспечения обнаружения искажений информации при передаче сообщения по сети. Контрольная сумма занимает два байта в заголовке сообщения и вычисляется суммированием всех полей исходного сообщения, за исключением собственно поля **CHECKSUM**. Если значение вычисленной приемником контрольной суммы не совпадает с переданным значением, сообщение протокола IGMP считается искаженным и уничтожается.

В поле **Group Address** размещаются групповые адреса Интернета. Содержание и назначение этого поля определяется типом сообщения протокола IGMP. При передаче сообщения General Membership Query в поле **Group Address** помещается код 0.0.0.0. При формировании специального запроса Group-Specific Membership Query в это поле должен быть помещен адрес

группы, для которой требуется установить наличие приемников. В сообщениях Membership Report, формируемых узлами — приемниками группового трафика, в поле **Group Address** соответственно должен быть размещен групповой адрес GDA той группы, в которой участвует или желает участвовать этот узел. В сообщении Leave Group в этом поле узел размещает адрес той группы, в которой он не желает больше участвовать.

Для обеспечения обратной совместимости с первой версией протокола сообщение протокола IGMP может содержать дополнительные поля и быть размером более 8 байтов. Хотя маршрутизаторы и узлы, применяющие вторую версию протокола, могут игнорировать содержимое этих полей, следует учитывать, что при вычислении контрольной суммы сообщения суммируются все поля сообщения — в том числе и дополнительные.

Особенности использования и обеспечение совместимости протокола IGMP версий v1 и v2

Основное отличие между первой и второй версией протокола IGMP заключается в том, что протокол IGMP v2 использует процедуру Leave Group для явного исключения из группы узла по его запросу. В противоположность этому, в протоколе IGMP v1 применяются только два типа сообщений Query и Report. Для определения текущего состава членов в группах маршрутизатор протокола IGMP v1 периодически посылает общие или специальные сообщения Query. Если в течение установленного временного интервала групповой маршрутизатор не получает ответные сообщения от членов какой-либо группы G, он делает вывод, что данная группа больше не имеет приемников в подключенном к нему сегменте локальной сети. В этом случае групповой маршрутизатор прекращает направлять трафик группы G в отмеченный сегмент или прекращает принимать этот трафик совсем — в том случае, если и на других его интерфейсах тоже нет потребителей группового трафика.

Очевидно, что использование процедуры умолчания для исключения из группы в протоколе IGMP v1 неизбежно приведет к тому, что в течение довольно длительного интервала времени сегмент сети будет наполняться никому уже не нужным трафиком. Поэтому протокол IGMP v2 использует процедуру явного исключения узла из Multicast-группы. При использовании такой процедуры узел, не заинтересованный более в приеме группового трафика группы G, обязан сформировать специальное сообщение Leave Group. Образно говоря, при использовании протокола IGMP v1 узел покидает свою группу, не прощаясь (по-английски), а узел, использующий протокол IGMP v2, должен попрощаться. Однако следует учитывать, что прощаться должен далеко не каждый узел, покидающий группу. В соответствии со спецификой информационного обмена в режиме Multicast, для группового маршрутизатора важным будет только сообщение Leave Group, сформи-

рованное последним узлом, покидающим группу. Таким образом, задача установления количества оставшихся членов в группе не является тривиальной и может быть решена двумя способами:

- ❒ узел определяет, что он является последним членом группы;
- ❒ групповой маршрутизатор определяет, что узел, сформировавший сообщение Leave Group, действительно был последним членом своей группы.

В первом случае узел должен прослушивать сообщения Report, формируемые другими узлами. Если в течение установленного интервала времени Max Response Time узел не услышит сообщений Report, сформированных другими узлами, у него есть некоторые основания полагать, что в его группе участников не так много. Маршрутизатор для определения полномочий узла, сформировавшего сообщение Leave Group, может использовать следующее правило: сообщение Leave Group принимается только от узла, который сформировал последнее сообщение Report для данной группы.

Однако использование обоих перечисленных методов в совокупности не может обеспечить полной гарантии того, что в подозреваемой группе больше не осталось приемников. Поэтому перед отключением группового трафика от некоторого сегмента маршрутизатор делает "контрольный выстрел" — передает в него несколько последовательных специальных сообщений Membership Query для подозреваемой группы G. В том случае, если эти запросы останутся без ответов, маршрутизатор выполняет процедуру отключения сегмента.

Легко заметить, что, обеспечить совместимость узлов, использующих различные версии протокола IGMP, достаточно затруднительно. Поэтому руководящие документы организации IETF (Internet Engineering Task Force) рекомендуют использование дополнительных процедур динамической взаимной адаптации компонентов протоколов IGMP v1 и IGMP v2.

Для обеспечения совместимости с групповыми маршрутизаторами протокола IGMP v1 узлы, применяющие вторую версию протокола, не должны использовать сообщений Membership Report протокола IGMP v2, поскольку такие сообщения игнорируются указанными маршрутизаторами.

Чтобы обеспечить совместимость с групповыми маршрутизаторами протокола IGMP v1, групповые маршрутизаторы, применяющие вторую версию протокола, должны быть административно переконфигурированы на IGMP v1.

Для обеспечения совместимости с узлами протокола IGMP v1 узлы, применяющие вторую версию протокола, должны прослушивать сообщения Membership Report протокола IGMP v1 наравне с сообщениями своей версии протокола.

Чтобы обеспечить совместимость с узлами протокола IGMP v1, маршрутизаторы, применяющие вторую версию протокола, должны игнорировать любые сообщения Leave Group, формируемые членами смешанной группы.

Дальнейшее развитие протокола IGMP

В дополнение к функциям текущих версий (IGMP v1 и IGMP v2), последующие версии протокола должны обеспечивать возможность использования дополнительных режимов информационного обмена с применением групповых адресов, например — Source Specific Multicast. Особенность этого режима заключается в том, что при его использовании узлы, заинтересованные в приеме группового трафика, могут сами определять адреса поставщиков этого трафика. Предполагается, что эта дополнительная возможность позволит обеспечить более высокий уровень информационной безопасности и более эффективное использование ресурсов вычислительных сетей.

Благодаря применению дополнительных сообщений новой версии протокола узел может определить как адреса желательных поставщиков группового трафика (белый список), так и адреса нежелательных поставщиков (черный список).

Таким образом, реализация всех возможностей, заложенных в первой и последующих версиях протокола, позволяет обеспечить эффективное использование всех преимуществ, которыми, без сомнения, обладает информационное взаимодействие с использованием групповых адресов.

Практическая реализация протокола IGMP в современных сетях

В современных сетях использование всех преимуществ информационного взаимодействия с применением групповых адресов невозможно без включения дополнительных функций в существующих сетевых инфраструктурах. Дело в том, что практически все современные локальные сети построены на коммутаторах, оперирующих на канальном уровне модели OSI (Open System Interconnect). Применение коммутаторов позволяет наиболее эффективно использовать пропускную способность сети, поскольку в данном случае для направления трафика в тот или иной порт коммутатор анализирует адреса назначения канального уровня. В соответствии с содержанием своей внутренней таблицы коммутации, коммутатор направляет блок данных канального уровня — кадр — только в тот порт, к которому подключен приемник, имеющий соответствующий адрес канального уровня (MAC-адрес). Построение внутренней таблицы коммутатор выполняет автоматически с помощью специального режима обучения. В этом режиме коммутатор анализирует поле адреса источника у всех кадров, поступающих в его порты, и устанавливает соответствие между номерами портов и MAC-адресами подключенных абонентов. Следовательно, при использовании коммутатора в сегменте сети возможно существование нескольких параллельных информационных потоков, благодаря чему и достигается повышение общей пропускной способности.

Совершенно очевидно, что схема информационного взаимодействия с использованием групповых адресов изначально достаточно плохо согласуется с современной концепцией построения сети. Это происходит потому, что режим информационного обмена Multicast в отличие от режима Unicast предполагает активность приемника, а не источника трафика. Действительно, используя стандартный режим обучения, коммутатор никоим образом не сможет установить потребность некоторого узла в групповом трафике, поскольку групповые адреса могут быть использованы только в качестве адресов назначения. Следовательно, коммутатор должен будет обрабатывать групповой трафик так же, как и широковещательный, то есть передавать его во все активные порты. Однако в отличие от широковещательного трафика, групповой трафик имеет большую интенсивность, поэтому даже наличие всего одного потребителя такого трафика в сети может привести к существенному уменьшению пропускной способности вследствие загрузки портов коммутатора невостребованным групповым трафиком.

Описанная проблема может быть решена тремя способами:

- статическим конфигурированием коммутатора;
- получением коммутатором информации о членах групп от группового маршрутизатора;
- посильным участием коммутатора в процедурах протокола IGMP.

Первый из описанных способов наиболее прост в реализации. В данном случае в таблице коммутации администратором формируются статические записи, наличие которых позволяет коммутатору обрабатывать групповой трафик так же, как одноадресный. При получении от группового маршрутизатора кадра для потребителей трафика некоторой группы G коммутатор направит его только в те порты, где этот кадр действительно ждут (по мнению администратора сети). Однако кажется маловероятным, что этот метод найдет сколько-нибудь заметное применение в современных сетях вследствие своей недостаточной гибкости.

Достаточно перспективным представляется использование процедур получения коммутатором информации о членах групп от группового маршрутизатора. При этом между коммутатором и групповым маршрутизатором устанавливаются специальные доверительные отношения, вследствие чего коммутатор оперативно и динамически получает от маршрутизатора необходимую информацию для корректировки содержимого таблицы коммутации. Несомненным достоинством такого метода следует признать его безусловную экономичность. Каждое из взаимодействующих при этом устройств — коммутатор и маршрутизатор — выполняют исключительно свои функции, благодаря чему и обеспечивается полная гармония. Примером подобного взаимодействия является протокол CGMP (Cisco Group Management Protocol) корпорации Cisco Systems.

Наиболее перспективным, однако, представляется третий способ решения указанной проблемы. В этом случае для обеспечения автоматического построения таблицы коммутации для обработки группового трафика коммутатор берет на себя часть функций устройства третьего уровня OSI. При этом коммутатор должен анализировать содержимое пакетов, выискивая среди них пакеты протокола IGMP. Именно поэтому подобные методы и получили свое название — *прослушивание (snooping)* протокола IGMP. Применение подобных методов, безусловно, приводит к резкому увеличению вычислительной нагрузки на коммутатор. Поэтому выполнение коммутатором описанных функций возможно только в том случае, если для этих целей будут использоваться специализированные интегральные микросхемы.

Протокол CGMP (Cisco Systems)

Протокол управления группами корпорации Cisco Systems CGMP (Cisco Group Management Protocol) предназначен для обеспечения эффективного распространения группового трафика в современных вычислительных сетях, построенных на коммутаторах.

В протоколе CGMP применяется следующая схема информационного обмена: узел — потребитель группового трафика — направляет сообщения протокола IGMP (Membership Report или Leave Group) групповому маршрутизатору, который преобразует эти сообщения в кадры протокола CGMP и направляет их коммутатору. Можно сказать, что маршрутизатор "разжевывает" сообщения протокола сетевого уровня IGMP и "скармливает" их коммутатору в том виде, в котором он их может принять.

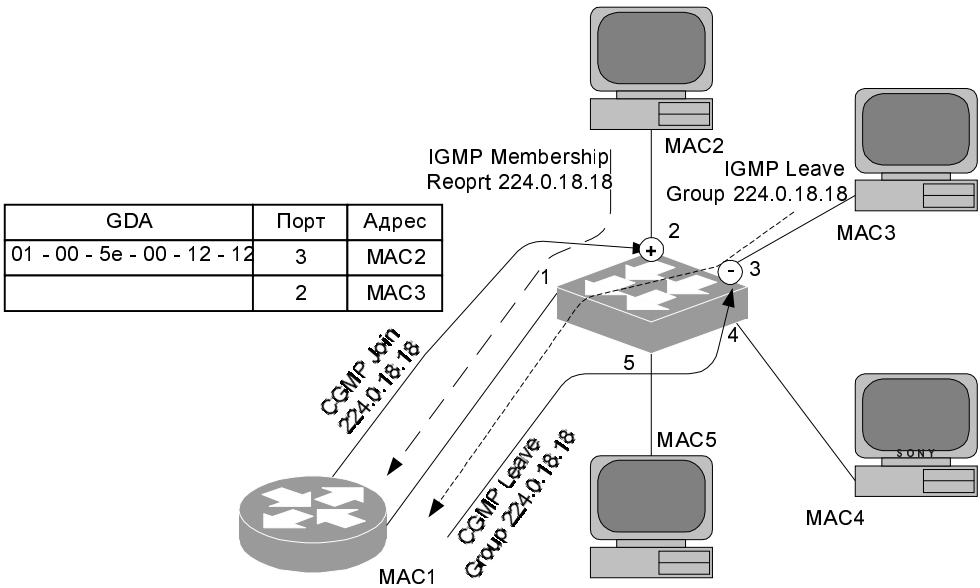
На рис. 6.4 приведена схема организации информационного обмена при использовании протокола CGMP. Следует учесть, что содержимое таблицы коммутации в данном случае представлено весьма условно.

При получении сообщения протокола IGMP Membership Report от узла MAC2, маршрутизатор устанавливает, что в группе 224.0.18.18 появился новый член, вследствие чего возникла потребность в корректировке содержимого таблицы коммутации на коммутаторе. Для внесения этих изменений маршрутизатор формирует кадр протокола CGMP, в котором он указывает MAC-адрес нового члена группы и соответствующий групповой адрес канального уровня. Получив такой кадр, коммутатор вносит изменения в таблицу коммутации, обеспечивая тем самым распространение группового трафика 224.0.18.18 через порт 2. Примерно таким же образом осуществляется корректировка таблицы коммутации при удалении пользователя из группы.

Примечание

Обратите внимание, что взаимодействие по протоколу CGMP производится с использованием MAC-адресов, поскольку коммутатор второго уровня не опе-

рирует сетевыми адресами, а маршрутизатор не имеет информации о номерах портов коммутатора, к которым подключен тот или иной потребитель.



GDA	Порт	Адрес
01 - 00 - 5e - 00 - 12 - 12	3	MAC2
	2	MAC3

Рис. 6.4. Схема организации информационного обмена по протоколу CGMP.

Таким образом, при появлении нового члена выполняется определенная последовательность действий.

1. Узел, заинтересованный в приеме группового трафика, направляет сообщение IGMP Membership Report в адрес группового маршрутизатора.
2. На основании полученного сообщения групповой маршрутизатор формирует кадр протокола CGMP, в который он помещает MAC-адрес группы и MAC-адрес нового потребителя ее трафика.
3. Коммутатор принимает кадр протокола CGMP, корректирует свою таблицу коммутации и определяет номер порта пользователя, используя в качестве ключа полученный от коммутатора MAC-адрес.

Прослушивание сообщений протокола IGMP коммутаторами ЛВС

Формирование таблицы коммутации для обработки группового трафика коммутатор может выполнить самостоятельно в том случае, если будет анализировать сообщения, направляемые групповому маршрутизатору. На первый взгляд, схема информационного обмена в данном случае существенно упрощается. Коммутатор принимает сообщение Membership Report протокола

IGMP, анализирует его и устанавливает канальный адрес группы, к которой хочет подключиться потребитель группового трафика. Поскольку номер порта, к которому подключен потребитель, в данном случае определить очень легко, корректировка таблицы коммутации выполняется также просто. Когда коммутатор услышит сообщение Leave Group, сформированное этим потребителем, он произведет обратные действия и извлечет номер порта потребителя из своей таблицы коммутации.

На самом деле, информационный обмен производится по существенно более сложной схеме, поскольку коммутатор должен учитывать некоторые факторы.

1. При получении сообщения Leave Group обязательно нужно провести проверку его достоверности, поскольку оно вполне может быть послано не последним, а просто безответственным членом группы.
2. Узлы, использующие протокол IGMP v1, вообще не способны формировать сообщения Leave Group.
3. Коммутатор должен уметь определять, к какому из его портов подключен групповой маршрутизатор, для того чтобы не отправлять полученные сообщения протокола IGMP во все порты.
4. Сообщение Membership Report протокола IGMP формируют не только узлы, намеревающиеся стать членами группы, но и те, которые уже являются членами этой группы.
5. Рассмотрим пример сети, представленной на рис. 6.5. В рассматриваемом варианте предполагается, что коммутатор способен выполнять прослушивание сообщений протокола IGMP. В частности, при поступлении в первый порт сообщения Membership Query протокола IGMP коммутатор может с уверенностью констатировать, что к этому порту подключен групповой маршрутизатор. Поскольку это сообщение адресуется всем узлам группы, имеющей групповой адрес 224.0.0.1, коммутатор передает его во все свои порты. В ответ на полученное сообщение Membership Query узлы, заинтересованные в приеме группового трафика, формируют ответные сообщения Membership Report протокола IGMP. При обработке этих сообщений коммутатор должен проанализировать, является ли уже пославший его узел членом данной группы. В противном случае коммутатор выполняет корректировку таблицы коммутации, добавляя новый порт в существующую группу. Принятые сообщения Membership Report коммутатор направляет в порт 1, к которому, по его мнению, должен быть подключен групповой маршрутизатор.
6. Коммутатор, выполняющий прослушивание сообщений протокола IGMP, может взять на себя дополнительные функции, чем существенно облегчит жизнь групповому маршрутизатору. Допустим, что в приведенном примере узел MAC3 сформировал ошибочное сообщение Leave Group, не будучи последним членом группы 224.0.0.1 на данном сегмен-

те сети (репитер на рисунке не показан). Умный коммутатор вместо того, чтобы отправить это сообщение групповому маршрутизатору, может сам выполнить проверку его достоверности. Для этого коммутатор должен направить в подозрительный сегмент самодельное сообщение Membership Query. Поскольку в рассматриваемом примере на это сообщение было получено сообщение Membership Report от узла MAC6, коммутатору не нужно передавать ошибочное сообщение Leave Group дальше по инстанции.

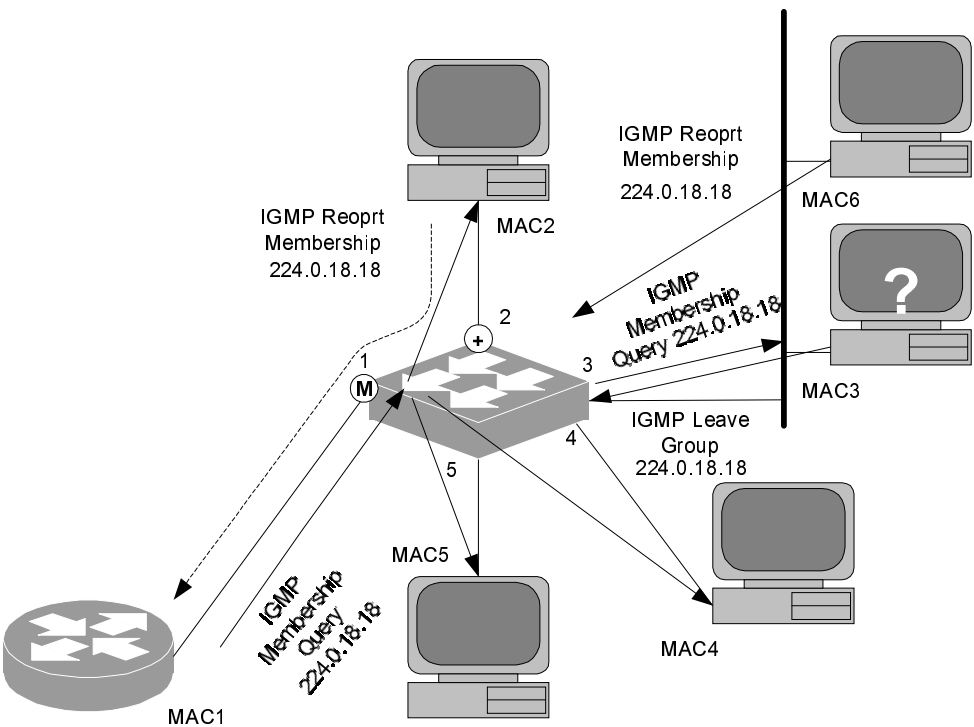


Рис. 6.5. Прослушивание сообщений протокола IGMP коммутатором ЛВС

Режим прослушивания сообщений протоколов третьего уровня, таким образом, позволяет не только обеспечить повышение эффективности использования пропускной способности вычислительной сети, но и снизить нагрузку на маршрутизатор. Следует, однако, учитывать, что применение этого режима, во-первых, резко увеличивает нагрузку на коммутатор, а во-вторых, требует внимательного конфигурирования и правильного администрирования.

Кроме того, при рассмотрении протоколов групповой маршрутизации будет показано, что использование механизма протокола IGMP snooping не

позволяет избежать перегрузки сети ненужным трафиком в ситуации, когда к центральному коммутатору подключены несколько маршрутизаторов. В этом случае коммутатор, в соответствии с требованиями алгоритмов протокола IGMP, направляет ВЕСЬ групповой трафик сегмента сети в КАЖДЫЙ из портов, к которым подключены маршрутизаторы. Ни один из маршрутизаторов, подключенных к коммутатору, не сможет отказаться от получения группового трафика с помощью специальной процедуры Prune (отсечение), даже если захочет, поскольку в этом случае произойдет отключение всего сегмента целиком.

Протоколы групповой маршрутизации

При описании процедуры группового информационного взаимодействия в сети Интернет указывалось, что особая роль в организации этого взаимодействия отводится групповым маршрутизаторам. Действительно, только эти устройства способны оптимальным образом обеспечить соединение источника группового трафика, который может находиться в любом месте сети, с потребителями этого трафика, расположение и количество которых также ничем не ограничено. Для того чтобы обеспечить распространение группового трафика в нужном направлении по сети, групповые маршрутизаторы используют специальные протоколы маршрутизации (Multicast routing protocols). Эти протоколы позволяют строить оптимальные пути распространения Multicast-трафика от одного маршрутизатора к другому. Следовательно, вначале групповой маршрутизатор при помощи протокола IGMP узнает о необходимости приема трафика некоторой группы G, а затем при помощи протоколов групповой маршрутизации подыскивает в сети подходящий источник такого трафика.

Примечание

Поскольку не существует четкой дисциплины группового вещания, следует учитывать, что в сети одновременно может существовать несколько источников с одинаковым значением GDA (Group Destination Address).

Так же, как и обычные протоколы одноадресной маршрутизации, протоколы групповой маршрутизации основаны на концепции "Hop-by-Hop", следовательно, групповому маршрутизатору достаточно определить, от какого из соседних маршрутизаторов он будет получать трафик для группы G. Двигаясь от одного маршрутизатора к другому, запрос проходит в направлении от потребителя — к источнику. Затем по проложенному таким образом маршруту направляется групповой трафик от источника к потребителю.

Режимы доставки группового трафика

Для построения маршрута доставки группового трафика могут быть использованы два основных режима — DENSE (частый) и SPARSE (редкий).

Отличие между этими двумя режимами заключаются в том, что при использовании режима DENSE каждый маршрутизатор по умолчанию считается заинтересованным в приеме группового трафика. Если на самом деле это не так, групповой маршрутизатор должен направить вышестоящему маршрутизатору сообщение об отказе от приема группового трафика. Поэтому в режиме DENSE сеть сначала затапливается трафиком некоторой группы, а затем ненужные ветки отключаются. В результате этих процессов формируется дерево, корнем которого является источник, а листьями — приемники группового трафика. Ветви этого дерева образуют промежуточные маршрутизаторы. Такое дерево называется *деревом распространения группового трафика* (Distributed Tree). Режим DENSE идейно очень хорошо согласуется с концепцией информационного взаимодействия с использованием групповых адресов и достаточно прост для реализации.

Основным недостатком этого метода является то, что маршрутное дерево DENSE принципиально не может иметь стабильную структуру. Легко заметить, что если потенциальный потребитель группового трафика не успел подать соответствующую заявку до того, как "волна" этого трафика "омоет" ближайший к нему групповой маршрутизатор, то у него практически не остается шансов на получение данного трафика. Следовательно, "волны" группового трафика должны формироваться регулярно и с не очень большим периодом, поскольку именно величина этого периода определяет оперативность подключения к источнику группового трафика абонента. Очевидно, что при использовании режима DENSE не может быть и речи об эффективном использовании сетевых ресурсов, поскольку периодическую повышенную нагрузку испытывают и групповые маршрутизаторы, и каналы передачи данных.

Режим DENSE целесообразно использовать в тех случаях, когда большинство абонентов сети заинтересовано в приеме группового трафика. В противном случае должен применяться более эффективный и экономичный режим SPARSE.

В противоположность режиму DENSE, при использовании режима SPARSE групповой трафик направляется маршрутизатором через один из своих портов только после получения запроса на этот порт от потребителя этого трафика. Естественно, что реализация режима SPARSE представляет значительные трудности, поскольку круг задач, решаемых маршрутизатором в этом режиме, значительно более широк. Вспомним, что маршрутизатору DENSE трафик "падает в рот сам собой" — задача маршрутизатора заключается в том, чтобы либо принять этот трафик, либо отторгнуть его. В режиме SPARSE дерево распространения маршрутной информации формируется в направлении, противоположном распространению группового трафика — "от листьев к корню".

Групповые маршрутизаторы, функционирующие в режиме DENSE или SPARSE, должны использовать специальные процедуры, исключающие

возможность перемещения группового трафика по циклическим маршрутам. Поскольку маршрутизатор SPARSE должен сам искать источник группового трафика, то в дополнение к этим процедурам он должен применять специальные алгоритмы, упрощающие процедуру поиска этого источника.

В сетях, использующих режим групповой адресации, применяются специальные протоколы маршрутизации. Основной особенностью этих протоколов является то, что в противоположность протоколам одноадресной маршрутизации, они предназначены для поиска пути к источнику информации, а не к ее приемнику. Поэтому в маршрутных таблицах, формируемых протоколами групповой маршрутизации, размещаются не адреса сетей назначения, а адреса источников группового трафика. В таблице групповой маршрутизации источник группового трафика имеет два адреса, обозначаемых S и G:

- S — Unicast-адрес протокола IP класса (AC) этого источника в сети Интернет;
- G — Multicast-адрес класса D группы, для которой этот источник формирует трафик.

Первый адрес протоколы групповой маршрутизации используют для построения дерева распространения трафика, а второй — в качестве ключа для поиска первого. Как правило, при построении маршрута протоколы групповой маршрутизации достаточно тесно интегрируются с обычными протоколами маршрутизации, поскольку при этом обеспечивается возможность полного или частичного использования имеющихся баз маршрутной информации.

Для построения маршрута распространения группового трафика протоколы групповой маршрутизации могут применять режимы DENSE или SPARSE. Выше уже было отмечено, что протоколы групповой маршрутизации DENSE должны использовать специальные процедуры, для того чтобы предотвратить появление циклических маршрутов. На примере сети, структура которой приведена на рис. 6.6, рассмотрим схему возникновения циклического маршрута при использовании алгоритма маршрутизации DENSE.

В данной сети основной маршрут распространения группового трафика проходит через маршрутизатор ГМ1. В соответствии с правилом затопления маршрутизаторы ГМ2 и ГМ3 направляют полученный от ГМ1 трафик через все свои интерфейсы, за исключением того, по которому они получили этот трафик. Поэтому они оба отправляют групповой трафик NG в сеть N3. В результате этого каждый из маршрутизаторов получает дополнительную копию всех дейтаграмм, отправленных источником группового трафика. Легко видеть, что, если не предпринять дополнительных усилий, процесс циклического тиражирования групповых дейтаграмм будет шириться и развиваться, нарушая нормальный режим функционирования сегмента сети.

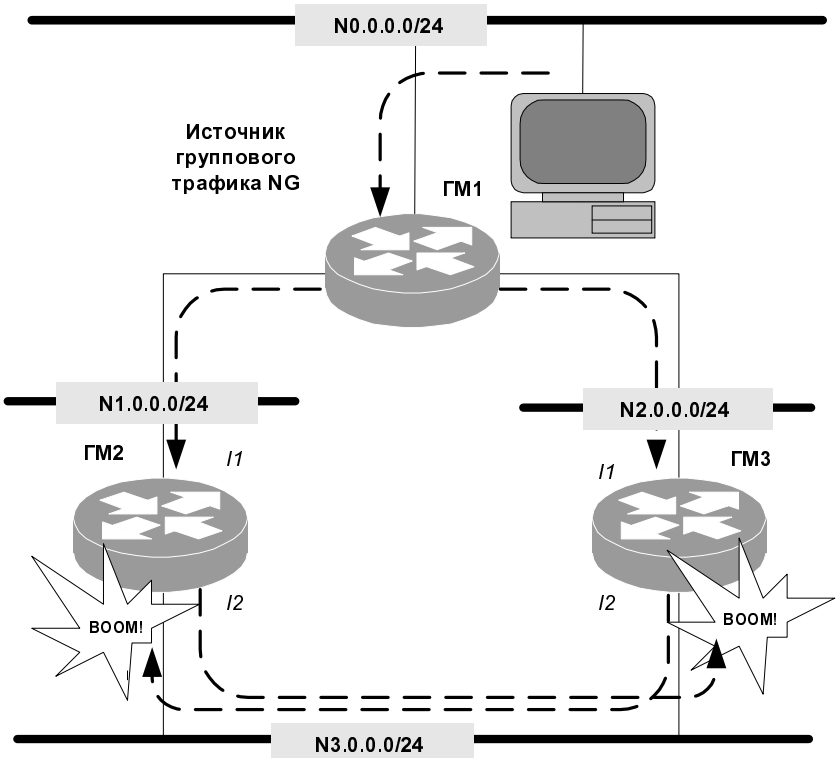


Рис. 6.6. Пример возникновения циклического маршрута

Классической мерой борьбы с этим явлением является ограничение времени жизни дейтаграмм в сети. Эта универсальная мера, однако, не может обеспечить достаточный уровень оперативности реакции на возникновение петли. Поэтому в протоколах групповой маршрутизации для противодействия возникновению циклических маршрутов используется специальный алгоритм, который называется Reverse Path Forwarding (RPF).

При использовании этой процедуры групповой маршрутизатор, используя таблицу протокола одноадресной маршрутизации, определяет номер интерфейса, на который возможен прием группового трафика NG от источника из сети N0.0.0.0/24. Если кратчайший маршрут в сеть N0.0.0.0/24 проходит через этот интерфейс, дейтаграммы принимаются для дальнейшей обработки. В том случае, если дейтаграммы прибывают на несоответствующий интерфейс, маршрутизатор уничтожает их без формирования каких-либо дополнительных сообщений. Поскольку в приведенном на рис. 6.6 примере интерфейсы, связывающие ГМ2 и ГМ3 с сетью N3, не входят в число разрешенных для приема группового трафика NG, оба маршрутизатора уничтожат пакеты, полученные из этих интерфейсов, что предотвратит возникновение порочного круга.

Некоторые из протоколов групповой маршрутизации используют дополнительные алгоритмы и процедуры, обеспечивающие возможность построения дерева распространения группового трафика, имеющего наиболее оптимальную структуру.

Протокол DVMRP

Протокол DVMRP (Distance Vector Multicast Routing Protocol) относится к протоколам групповой маршрутизации класса DENSE. Как следует из его названия, для построения маршрутных таблиц этот протокол использует алгоритм вектора дистанции, применяя число переходов в качестве метрики оценки качества маршрута. Протокол DVMRP был одним из первых протоколов групповой маршрутизации, использованных в сети Интернет. За время своего существования он подвергался многочисленным доработкам. В настоящее время в сети используется третья версия протокола DVMRP.

Так же, как и другие протоколы, применяющие схему DENSE, протокол DVMRP строит дерево распространения группового трафика от источника G, используя периодические затопления всего дерева этим трафиком. В том случае, если в некоторой ветви отсутствуют потребители трафика G, групповой маршрутизатор, образующий эту ветвь, должен передать сообщение об отказе от приема трафика данной группы — Prune — тому маршрутизатору, от которого он получает групповой трафик G. Каждому сообщению Prune ставится в соответствие величина временного интервала, в течение которого это сообщение считается действующим. По истечении этого интервала передача группового трафика в данном направлении автоматически возобновляется. Если до истечения временного интервала сообщения Prune в данной ветви все-таки появятся потребители этого вида трафика, ближайший групповой маршрутизатор должен передать вверх по инстанции сообщение Graft (подключение), назначением которого является отзыв переданного ранее отказа от приема группового трафика.

Принцип построения маршрута протоколом DVMRP

В отличие от некоторых других протоколов групповой маршрутизации, протокол DVMRP при построении дерева распространения группового трафика не "катается на хвосте" протокола одноадресной маршрутизации, а строит это дерево на основании собственной таблицы маршрутизации. Подобное решение не является бесспорным, поскольку дублирование функций протокола одноадресной маршрутизации протоколом групповой маршрутизации неминуемо приводит к существенному увеличению вычислительной нагрузки на маршрутизатор. В данном случае использование независимого протокола одноадресной маршрутизации в DVMRP объясняется следующими причинами:

- необходимостью применения всеми маршрутизаторами согласованной политики определения веса маршрута;

- обеспечением возможности построения независимых маршрутов для передачи одноадресных и групповых дейтаграмм.

Для построения оптимальной структуры дерева распространения группового трафика в дополнение к стандартной для протоколов групповой маршрутизации процедуре RPF (Reverse Path Forwarding) в протоколе DVMRP используется специальная процедура определения подчиненных маршрутизаторов на основе принципа Poisoned Reverse. Оптимальная структура дерева распространения группового трафика должна обеспечивать:

- прием группового трафика только с того интерфейса, который находится на кратчайшем пути между маршрутизатором и источником трафика;
- передачу группового трафика только через те из выходных интерфейсов маршрутизатора, которые по кратчайшему пути связывают потребителей трафика с его источником.

Первая из задач решается использованием процедуры RPF. При решении второй задачи каждый групповой маршрутизатор для каждого из источников группового трафика при помощи расширенной процедуры Poisoned Reverse определяет список подчиненных маршрутизаторов, которым он должен будет направлять трафик этой группы. Подчиненными маршрутизаторами (Dependent Downstream Routers) маршрутизатора ГМ являются непосредственно соседствующие с ним маршрутизаторы, которые выбрали ГМ в качестве промежуточного источника трафика группы G.

Таким образом, задача построения оптимальной структуры дерева передачи группового трафика решается с двух сторон — со стороны источника и со стороны приемника. Как известно, при использовании классической процедуры Poisoned Reverse маршрутизатор направляет первоисточнику маршрутной информации обновления с метрикой, соответствующей недостижимой сети. Использование такой процедуры позволяет одноадресным протоколам маршрутизации строить маршруты, не содержащие петель. Маршрутизатор DVMRP посылает источнику маршрутной информации обновления с метрикой "плюс бесконечность", добавляя установленное в этом протоколе значение метрики недостижимой сети (32) к метрике, представленной источником сети. Таким образом, если маршрутизатор получает от соседнего маршрутизатора обновления с метрикой, значение которой находится в диапазоне от "бесконечность" до 2-х "бесконечность", он вправе считать, что перед ним — подчиненный маршрутизатор, который ожидает приема группового трафика именно от него.

На рис. 6.7 приведена схема определения подчиненных маршрутизаторов при использовании расширенной процедуры Poisoned Reverse протокола DVMRP.

На представленной схеме пунктирными линиями отмечены направления передачи обновлений маршрутной информации для сети N1.0.0.0/24. Цифры над стрелками соответствуют метрикам представляемых маршрутов.

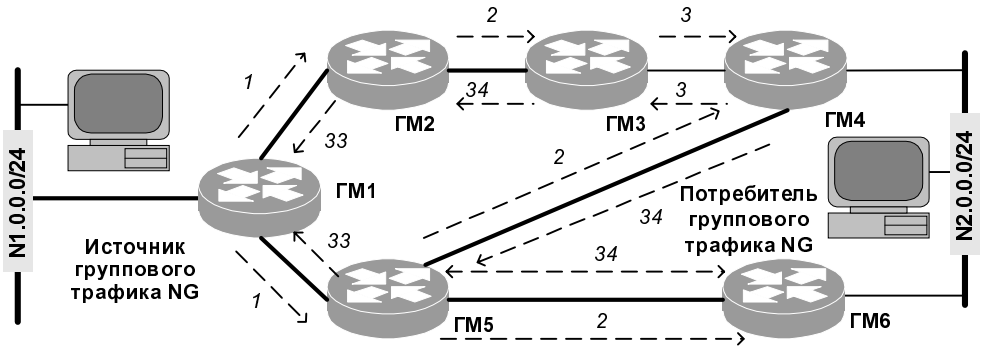


Рис. 6.7. Определение подчиненных маршрутизаторов DVMRP

Примечание

По умолчанию в протоколе DVMRP вклад каждого интерфейса в метрику маршрута устанавливается равным 1.

Легко видеть, что в данном случае у маршрутизатора ГМ4 не оказалось ни одного подчиненного маршрутизатора, поскольку маршрутизатор ГМ5 поставляет более короткий путь в сеть N1.0.0.0/24 для единственного кандидата — маршрутизатора ГМ4. Утолщенными линиями на рисунке показаны каналы, образующие построенное таким образом обрезанное дерево распространения (Truncated Broadcast Tree — TBT) группового трафика NG.

Дополнительные особенности построения дерева распространения группового трафика касаются тех случаев, когда несколько групповых маршрутизаторов DVMRP соревнуются за право поставлять трафик в сегмент локальной сети с множественным доступом. Очевидно, что для обеспечения более эффективного использования сетевых ресурсов вполне достаточно одного маршрутизатора, который поставляет групповой трафик в сегмент сети. В этой ситуации конфликтующие маршрутизаторы должны разобраться и выбрать наиболее достойного. В протоколе DVMRP такой маршрутизатор называется выбранным поставщиком (Designated Forwarder). Естественно, что в качестве основного критерия для выбора Designated Forwarder в данном случае должна быть использована метрика маршрута, связывающая этот маршрутизатор с источником группового трафика. При прочих равных условиях преимущество получает маршрутизатор, имеющий наименьший IP-адрес. На рис. 6.8 представлен заключительный этап построения дерева распространения группового трафика NG — выбор поставщика трафика для сети N2.0.0.0/24.

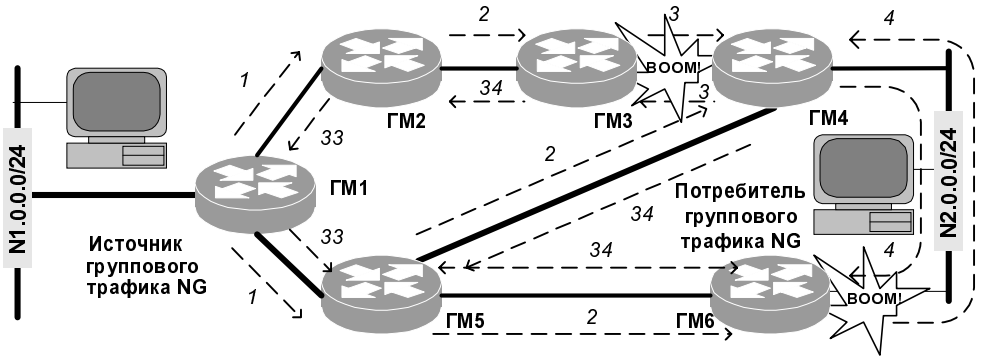


Рис. 6.8. Определение Designated Forwarder для сети N2.0.0.0/24

Сообщения протокола DVMRP

Для организации информационного обмена между маршрутизаторами протокола DVMRP используются специальные сообщения, передаваемые в дейтаграммах с установленным значением кода протокола IP равным 2. Некоторые из сообщений протокола, которые требуют подтверждения их доставки, квитируются на уровне приложения. Перечень сообщений протокола DVMRP приведен в табл. 6.4.

Таблица 6.4. *Сообщения протокола DVMRP*

Код	Обозначение	Назначение
1	Probe	Установление партнерских отношений
2	Report	Передача маршрутной информации
7	Prune	Отказ от приема группового трафика
8	Graft	Отзыв отказа от приема группового трафика
9	Graft Ack	Подтверждение отзыва отказа от приема группового трафика

Сообщение Probe исполняет в протоколе DVMRP функции, аналогичные функциям сообщения HELLO протоколов маршрутизации OSPF, EIGRP. Основной задачей этого сообщения является обеспечение установления партнерских отношений между маршрутизаторами протокола DVMRP. Эта задача включает в себя несколько более частных задач, в число которых входят:

- поиск соседних маршрутизаторов протокола DVMRP;
- согласование параметров протокола DVMRP;
- установление функционального статуса партнера.

Сообщение Probe маршрутизатор периодически передает через все свои интерфейсы. В это сообщение он помещает адреса всех маршрутизаторов, от которых он получил аналогичное сообщение. В этом сообщении также размещаются признаки, указывающие на номер версии реализованного протокола, признаки поддержки дополнительных возможностей протокола и идентификатор, который соседние маршрутизаторы протокола DVMRP используют для определения неисправностей.

Сообщение Report предназначено для передачи маршрутной информации протокола DVMRP. Как и все остальные протоколы маршрутизации Distance-Vector, протокол DVMRP предполагает периодическую передачу обновлений маршрутной информации. Обновление маршрута обязательно содержит метрику, выраженную в количестве промежуточных узлов, и маску представляемой сети. В дополнение к периодическим обновлениям маршрутной информации, подобные обновления могут быть сформированы непосредственно при возникновении изменения текущего состояния системы.

Сообщение Prune представляет собой требование о прекращении передачи группового трафика. Это сообщение направляется только тому ближайшему групповому маршрутизатору, который поставляет невостребованный теперь трафик. Если получение этого сообщения означает для вышестоящего маршрутизатора полную потерю потребителей, он, в свою очередь, также формирует сообщение Prune и направляет его дальше по инстанции.

Сообщение Graft предназначено для устранения последствий от выдачи сообщения Prune, то есть, по сути является запросом на возобновление передачи группового трафика. Важность этого сообщения легко понять, если учесть, что установленные интервалы действия сообщения Prune могут достигать единиц часов. Поэтому для сообщения Graft в протоколе DVMRP применяется специальный режим передачи — с подтверждением доставки. Передача сообщения Prune повторяется, если в течение установленного временного интервала на него не будет получено квитирующее сообщение Graft Ack.

Протокол PIM

В отличие от протокола DVMRP, протокол групповой маршрутизации PIM (Protocol Independent Multicast) использует таблицы маршрутизации, которые построены алгоритмами одноадресной маршрутизации для создания на их основе деревьев доставки многоадресного трафика до его приемников. Поэтому характерной особенностью этого протокола маршрутизации является его независимость от типа протокола одноадресной маршрутизации, который используется в сети.

Протокол групповой маршрутизации PIM может применяться в двух основных режимах — SPARSE и DENSE.

Особенности использования PIM в режиме DENSE

Протокол PIM при его использовании в режиме DENSE (PIM-DM) имеет много общего с протоколом DVRMP. Так же, как и протокол DVRMP, протокол PIM строит дерево распространения группового трафика от источника G, используя периодические затопления всего дерева этим трафиком. В том случае, если в некоторой ветви отсутствуют потребители трафика G, групповой маршрутизатор протокола PIM, образующий эту ветвь, должен передать сообщение об отказе от приема трафика данной группы (Prune) вышестоящему маршрутизатору. Существенное отличие этих протоколов состоит в том, что протокол PIM не усекает дерево распространения группового трафика на стадии его создания. Таким образом, задача построения дерева в данном случае решается только со стороны приемника трафика с использованием процедуры RPF. Вследствие этого, построенное таким образом дерево распространения группового трафика будет заведомо неоптимальным, поскольку маршрутизаторы будут передавать полученный групповой трафик через все свои интерфейсы, за исключением того, с которого этот трафик был получен. Следовательно, построенное дерево наверняка будет содержать дублирующие ветви передачи группового трафика. Корректировка структуры дерева в данном случае выполняется уже на стадии передачи трафика с использованием сообщений Prune и Graft. В качестве примера построения дерева группового трафика с использованием протокола PIM-DM рассмотрим фрагмент сети, представленной на рис. 6.9.

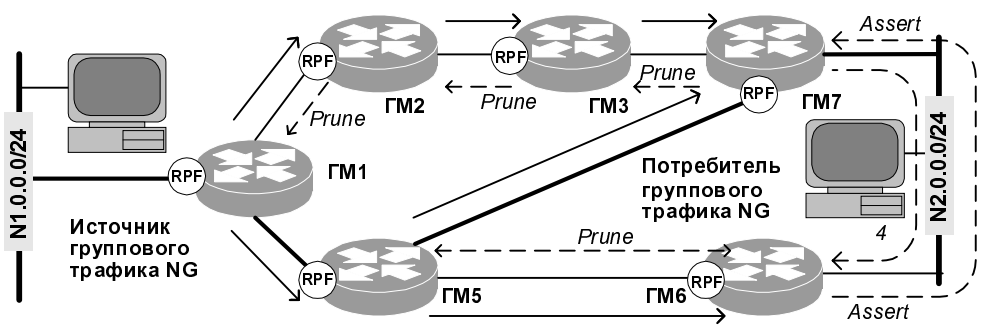


Рис. 6.9. Построение дерева распространения группового трафика с использованием протокола PIM-DM

Как уже было указано, для определения RPF интерфейсов групповых маршрутизаторов применяется маршрутная таблица используемого в сети протокола одноадресной маршрутизации. Стрелками на рисунке отмечены направления распространения группового трафика. Пунктирными стрелками обозначены сообщения протокола PIM, формируемые групповыми маршрутизаторами. Функция сообщения Prune абсолютно идентична функции одноименного сообщения протокола DVMRP. Сообщения Assert в протоколе

PIM-DM предназначены для определения поставщика группового трафика в сегмент сети с множественным доступом. Опять же, в отличие от протокола DVMRP, в протоколе PIM-DM преимущество при прочих равных условиях получает групповой маршрутизатор с наибольшим IP-адресом.

Примечание

Поскольку протокол PIM не использует сообщений для проверки состояния партнера, использование процедуры ASSERT может привести к тому, то в качестве победителя будет выбран неисправный маршрутизатор.

В представленном примере групповой маршрутизатор ГМ7 победил в соревновании за право представлять трафик NG в сети N2.0.0.0/24. Маршрутизатор посылает сообщение Prune в направлении маршрутизатора ГМ3 потому, что последний не подключен к RPF интерфейсу первого. В свою очередь, маршрутизаторы ГМ2, ГМ3 и ГМ6 формируют сообщения Prune из-за того, что они оказались не подключены к потребителям группового трафика. Результирующая структура дерева передачи групповой информации представлена на рисунке утолщенными линиями.

Режим SPARSE протокола PIM

Наиболее широко, однако, используется другой режим протокола PIM — режим SPARSE (PIM-SM). Как уже было выше отмечено, этот режим является существенно более экономичным и требует значительно меньше сетевых ресурсов для реализации. Особенностью протокола PIM-SM является то, что в нем для передачи группового трафика наравне с деревьями кратчайших путей (Shortest Path Tree) могут быть использованы разделяемые деревья (Shared Tree). Разделяемое дерево образует виртуальную магистраль, предназначенную для передачи группового трафика возможно от нескольких источников.

Информационный обмен в данном случае организуется с использованием специальных групповых маршрутизаторов, которые выполняют функцию "Rendezvous Point" — точка встречи и обозначаются как RP. Назначением этих маршрутизаторов является накопление информации об Unicast-адресах зарегистрированных источников группового трафика и передача ее по запросам, которые RP-маршрутизаторы получают от маршрутизаторов, непосредственно подключенных к приемникам группового трафика (Last hop router). После получения Unicast-адреса источника группового трафика Last hop router может построить дерево доставки группового трафика непосредственно от источника этого трафика. На рис. 6.10 представлена схема организации информационного взаимодействия групповых маршрутизаторов PIM-SM.

На приведенном примере функции Rendezvous Point выполняет маршрутизатор ГМ2. Поэтому маршрутизатор ГМ4, получив от потребителя заказ на

прием группового трафика NG, направляет запрос на получение этого трафика по Unicast-адресу маршрутизатора ГМ2. В процессе прохождения этого запроса по сети от ГМ4 до ГМ2 происходит построение дерева корневого пути Root Path Tree — RPT, которому впоследствии групповой трафик будет поступать от RP к потребителям. После получения сообщения Join корневой маршрутизатор констатирует, что построение дерева завершено, и в том случае, если RP принимает групповой трафик NG, начинает процесс доставки группового трафика по дереву RPT. В том случае, если корневой маршрутизатор в текущий момент не принимает групповой трафик, он формирует сообщение Join (участие) в адрес одного из зарегистрированных источников этой группы. Процесс регистрации источника инициируется ближайшим к нему групповым маршрутизатором — в приведенном примере это маршрутизатор ГМ1. Для регистрации источника на корневом маршрутизаторе ГМ1 формирует сообщение Register, которое он передает по известному ему адресу корневого маршрутизатора RP. Назначением этого сообщения является передача Unicast IP-адреса источника группового трафика NG корневому маршрутизатору. Сообщением Register STOP корневой маршрутизатор подтверждает, что регистрация источника завершена успешно. После того как маршрутизатор ГМ4 начинает прием группового трафика NG по дереву RP, он уже знает адрес источника этого трафика и, следовательно, может инициировать процедуру переключения на доставку группового трафика по более короткому пути (Source Path — SP). На схеме, представленной на рис. 6.10, этот путь отмечен утолщенной пунктирной линией.

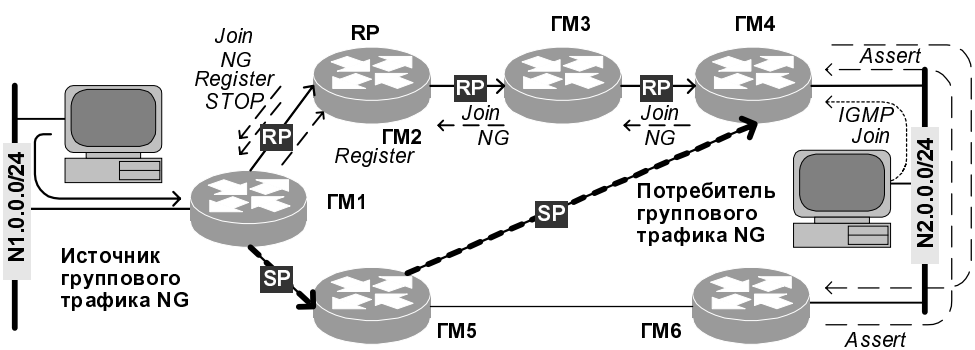


Рис. 6.10. Схема информационного взаимодействия маршрутизаторов PIM-SM

Таким образом, корневой маршрутизатор играет ключевую роль при выполнении всех процедур доставки группового трафика протокола PIM-SM, являясь посредником между всеми источниками и приемниками группового трафика сети.

Следовательно, для надежного функционирования протокола PIM в режиме SPARSE должно обеспечиваться динамическое назначение и надежное

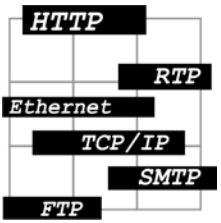
резервирование самих RP-маршрутизаторов. Эти функции могут быть выполнены с использованием двух режимов:

- режима Auto-RP;
- режима Bootstrap.

В режиме Auto-RP один или несколько маршрутизаторов конфигурируются для передачи анонсов о зарегистрированных источниках группового трафика. Для передачи этих анонсов используется специальная группа 224.0.1.39. Эти анонсы обрабатываются на маршрутизаторах-агентах для определения RP-маршрутизатора. Определение RP-маршрутизатора выполняется параллельно всеми агентами по единому алгоритму, в соответствии с которым среди равных выбирается маршрутизатор, имеющий максимальный IP-адрес. Выбор RP осуществляется независимо для каждой передающей группы, этот процесс называется mapping. Информация о сделанном выборе передается всем членам группы 224.0.1.40, в которую входят все остальные маршрутизаторы сети. Поскольку все узлы используют один и тот же алгоритм выбора, информация о выбранных RP-маршрутизаторах, сформированная разными агентами, является идентичной. Вследствие этого обеспечивается надежность функционирования всей системы в целом.

В режиме Bootstrap (начальная загрузка) один или несколько маршрутизаторов конфигурируются в качестве RP-кандидатов. Эти же или другие маршрутизаторы конфигурируются в качестве Bootstrap-кандидатов (BS). Последние начинают периодически (один раз в минуту) передавать всем членам группы 224.0.0.13 содержимое таблицы соответствия группового адреса и адреса RP-маршрутизатора. Кроме этого передается адрес самого Bootstrap-кандидата. Информация распределяется по сети в соответствии с классическими алгоритмами затопления с блокированием дублирующих ветвей путем использования процедуры Reverse Path Forwarding (RPF). Маршрутизаторы, которые являются RP-кандидатами, передают информацию о зарегистрированных источниках группового трафика BS-кандидату в одноадресном режиме. Все остальные маршрутизаторы сети принимают сообщения, сформированные BS-кандидатами, и формируют на их основе свои внутренние таблицы соответствия групповых адресов и адресов RP-маршрутизаторов. Для каждой записи этой таблицы устанавливается период времени, в течение которого эта запись считается актуальной. В том случае, если до истечения этого периода времени информация не будет подтверждена очередным сообщением, полученным от Bootstrap-кандидата, запись аннулируется.

Глава 7



Интеграция разнородного трафика в сетях TCP/IP

Особенность современных информационно-вычислительных сетей состоит в высокой интенсивности и неоднородности передаваемого по ним сетевого трафика. Это объясняется в первую очередь тем, что сегодня вычислительные сети играют роль несущей структуры, на основе которой строят свой бизнес современные предприятия. Широкое проникновение информационных технологий в сферу бизнеса заставляет современные предприятия внедрять и активно использовать разнообразные программные продукты и системы. Многие из широко распространенных современных программных продуктов используют модель взаимодействия "клиент-сервер", что предъявляет повышенные требования к пропускной способности вычислительной сети.

В последнее время вычислительные сети все активнее используются для передачи по ним голосового и видеотрафика. Для передачи этих разновидностей трафика к параметрам сети предъявляются жесткие специфические требования. Дело в том, что аудио- и видеотрафик имеет ярко выраженный потоковый характер и поэтому плохо приспособлен для передачи по сетям с коммутацией блоков данных. Для нормального функционирования потоковых приложений сеть должна обеспечивать стабильную минимальную задержку передаваемых блоков данных. Следовательно, в данном случае задача интеграции должна включать оптимальное преобразование входного аналогового потока в последовательность пакетов и выполнение обратного преобразования. Для обеспечения большей эффективности использования сетевых ресурсов может выполняться также сжатие передаваемых данных.

Таким образом, современные потоковые приложения предъявляют требования к следующим характеристикам производительности сети:

- Bandwidth — пропускная способность, скорость передачи данных;
- Delay — среднее или максимальное значение временной задержки передаваемого блока данных;
- Delay variation — максимальное значение изменения величины временной задержки передаваемого блока данных.

На вычислительные сети, основу которых составляют сети Интернет, возлагается задача обеспечения оптимальной интеграции разнородного трафика в условиях ограниченности сетевого ресурса. Для решения этой задачи могут быть использованы две различные модели обеспечения качества обслуживания:

- Integrated Service Model (IntServ) — модель объединенного обслуживания;
- Differentiated Service Model (DiffServ) — модель раздельного обслуживания.

При использовании модели IntServ весь трафик, передаваемый в системе, считается изначально однородным. Заданный уровень качества обслуживания в данном случае обеспечивается путем применения специальных механизмов динамического резервирования ресурсов для выделенной части передаваемого трафика. Такой механизм может быть, в частности, реализован при использовании протокола RSVP (ReSerVation Protocol).

Использование модели DiffServ предполагает неоднородность трафика, передаваемого в системе, и включает несколько последовательных этапов преобразования данных.

На первом этапе выполняется классификация трафика, проходящего через устройство. При этом в соответствии с реализуемой политикой обеспечения QoS (Quality of Service) весь трафик разбивается на классы обслуживания CoS (Class of Service), для каждого из которых устанавливается уровень качества обслуживания QoS в виде совокупности установленных требований к характеристикам производительности сети. В качестве критериев принадлежности трафика к некоторому классу может быть использовано значение поля **IP Precedence** или любого другого поля дейтаграммы (адрес источника, порт назначения и т. д.). В качестве признака принадлежности блока данных к определенному классу также может быть использована специальная метка, передаваемая по сети вместе с блоком.

В зависимости от типа функций, выполняемых сетевым устройством, на следующем этапе может выполняться регулирование трафика. Применение регулирования обеспечивает реализацию заданных значений QoS для каждого из установленных классов обслуживания. Для регулирования трафика устройство может использовать жесткий метод — Policing ("обрезание"), при котором блоки данных, выходящие за лимит сетевого ресурса своего класса, просто уничтожаются. Также может быть использован более мягкий метод регулирования трафика — Shaping ("формирование"). При использовании этого метода сверхлимитные блоки данных задерживаются в специальных буферах — отстойниках, дожидаясь там разрешения на использование сетевого ресурса. Основу обоих этих методов составляет универсальный механизм упорядочивания трафика (Queuing).

1. Для каждого из классов CoS создается очередь (Queue), в которую помещаются поступившие на обработку блоки данных этого класса. Порядок обслуживания классовых очередей определяется приоритетом соответствующего класса и типом реализуемой политики обеспечения QoS.

2. В классовых очередях блоки данных одного класса обычно обслуживаются по принципу FIFO (First In — First Out, то есть "первый пришел — первый ушел").

Применения описанных выше методов и процедур может оказаться недостаточным для обеспечения гарантированного значения задержки доставки блоков данных. Особенно сильно величина этой задержки влияет на выполнение мультимедийных приложений. В частности, установлено, что для нормального функционирования приложений технологии VoIP (Voice over IP) — "голос поверх IP", о которой будет рассказано ниже, величина задержки не должна превышать значение 0,15 секунды. Для того чтобы обеспечить гарантированное выполнение этого требования в условии, когда мультимедийный трафик конкурирует за право обладания сетевым ресурсом с трафиком локальной вычислительной сети, должен применяться метод Fragmentation (фрагментация, то есть дробление больших дейтаграмм).

Применение этой схемы на сетевых устройствах, однако, вовсе не гарантирует, что установленная система приоритетов будет соблюдена и соответствующие приложения получат в свое распоряжение требуемые ресурсы сети. Рассмотрим схему интеграции трафика нескольких приложений, представленную на рис. 7.1. В данном случае предполагается, что функция преобразования мультимедийного трафика в последовательность пакетов протокола IP выполняется специализированными устройствами. Функция интеграции разнородного трафика возлагается на маршрутизаторы. Эти маршрутизаторы могут обеспечить формирование и реализацию локальной политики обеспечения требуемых уровней QoS для каждого из классов трафика, однако этого недостаточно, чтобы обеспечить реализацию выбранной политики на всем протяжении маршрута. Для того чтобы обеспечить такую возможность, должны быть использованы специальные процедуры и протокол резервирования сетевых ресурсов.

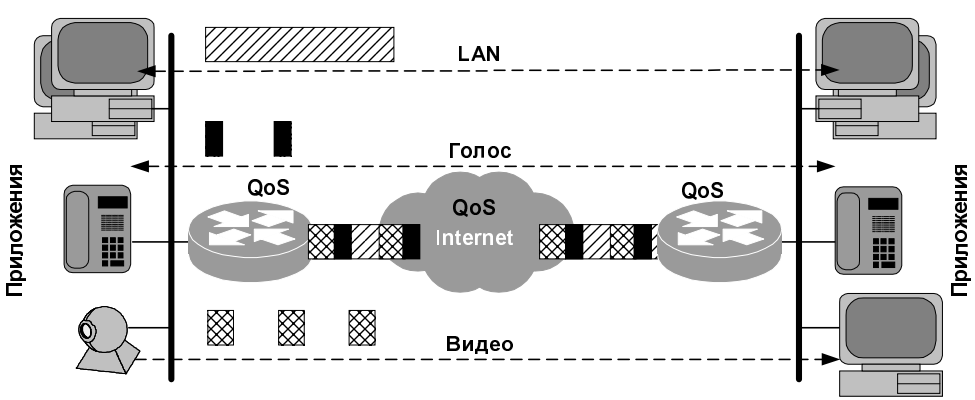


Рис. 7.1. Интеграция трафика нескольких приложений

К сожалению, наиболее популярные на настоящий момент сетевые технологии — протокол IP на сетевом уровне и Ethernet/IEEE 802.3 — на канальном — изначально одинаково плохо приспособлены для реализации описанных выше процедур. В этих условиях резко вырастает потребность в дополнительных процедурах и технологиях, позволяющих расширить возможности современных сетей в части оптимальной интеграции разнородного сетевого трафика.

Принципы построения мультимедийных компонентов сети Интернет

В рекомендации ITU-T H.323 (ITU-T — Telecommunication Standardization Sector of the International Telecommunications Union, отдел стандартизации международного союза телекоммуникаций, H.323 — рекомендация этого отдела) приведены требования к составу и компонентам систем преобразования мультимедийных аналоговых сигналов для передачи их по сетям с коммутацией пакетов, в соответствии с которыми строятся современные системы интеграции трафика.

Выполнение требований рекомендации H.323 обеспечивает:

- возможность управления полосой пропускания сети в интересах мультимедийных приложений;
- возможность взаимодействия компонентов сетей пакетной коммутации с компонентами сетей с коммутацией каналов;
- возможность организации многоточечных конференций;
- поддержку режима групповой адресации и многоадресной передачи;
- использование широкого спектра алгоритмов взаимного преобразования потоковых аналоговых сигналов в пакеты данных.

Основными протоколами доставки речевого- и видеотрафика в реальном масштабе времени служат протоколы RTP (Real Time Protocol, Протокол реального времени) и RTCP (Real Time Control Protocol, Протокол управления реального времени), регламентируемый рекомендацией H.225. В соответствии с положениями этой рекомендации при построении подобных систем могут быть использованы следующие функциональные компоненты:

1. Terminals — терминалы H.323;
2. Gateways — шлюзы H.323;
3. Gatekeepers — привратники H.323;
4. MCU — устройства управления многоточечными конференциями H.323.

Схема организации информационного взаимодействия между компонентами рекомендации ITU-T H.323 представлена на рис. 7.2.

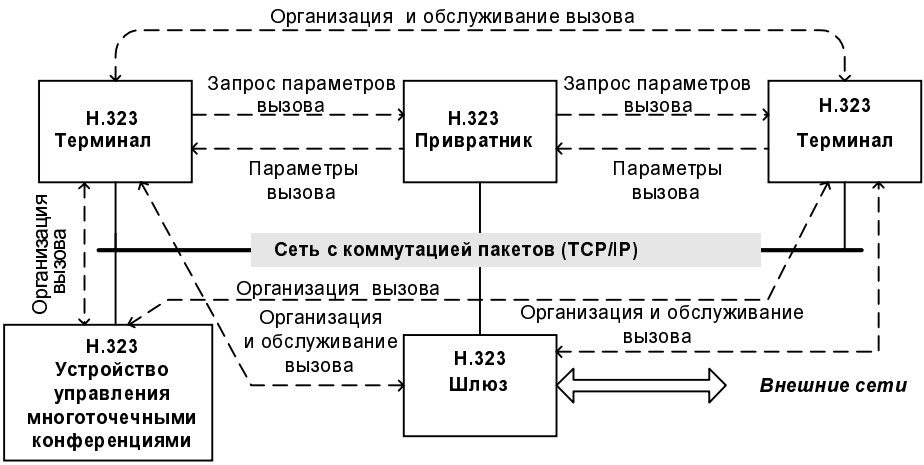


Рис. 7.2. Схема информационного взаимодействия между компонентами рекомендации ITU-T H.323

Терминал H.323 предназначен для обеспечения непосредственного доступа в сеть с коммутацией пакетов и выполняет прямое и обратное преобразование мультимедийных потоков данных в последовательность пакетов.

Шлюз H.323 обеспечивает взаимодействие компонентов этой системы с внешними сетями. По сути, шлюз представляет собой устройство, осуществляющее взаимное преобразование форматов представления данных (например, между форматами рекомендаций H.225.0 и H.221) или взаимное преобразование процедур информационного обмена (например, протокола управления мультимедийными объектами H.245 и процедуры управления терминалом H.242).

В современных сетях IP шлюзы H.323 обеспечивают взаимодействие этой сети с телефонной сетью общего пользования.

Привратник H.323 предназначен для управления шлюзами и терминалами H.323. При этом привратник выполняет следующие задачи:

- организацию и обслуживание процедур, трансляцию адресов, взаимный обмен адресной информацией с другими привратниками;
- регистрацию, аутентификацию и авторизацию пользователей, пользовательского оконечного оборудования и шлюза, сбор и накопление тарификационной информации;
- формирование запросов о выделении ресурсов сети с коммутацией пакетов (в данном случае — сети Интернет) для обеспечения передачи мультимедийной информации.

Устройства управления многоточечными конференциями H.323 предназначены для организации мультимедийного информационного обмена с участием трех и более абонентов. Эти устройства могут представлять собой отдельные объекты

или могут быть реализованы в виде функциональных модулей, входящих в другие устройства H.323. Различают два вида модулей организации конференций:

- Multipoint Controllers — контроллеры конференций;
- Multipoint Processors — процессоры конференций.

Контроллеры конференций обеспечивают выполнение необходимых для организации конференций предварительных процедур по согласованию протоколов и форматов передачи данных. Контроллеры конференций могут в качестве функциональных модулей входить в такие устройства рекомендации H.323, как терминал, шлюз и привратник.

Процессоры конференций предназначены для непосредственного управления потоками мультимедийной информации. Процессоры конференций могут в качестве функциональных модулей входить в такие устройства рекомендации H.323, как шлюз и привратник.

Терминальные устройства H.323

Как уже было выше отмечено, терминальные устройства H.323 выполняют прямое и обратное преобразование мультимедийных информационных потоков в последовательность блоков данных, предназначенных для передачи через сеть с коммутацией пакетов, в нашем случае — через сеть Интернет. На рис. 7.3 представлена функциональная схема организации информационного взаимодействия терминального устройства H.323 с источниками внешней информации.

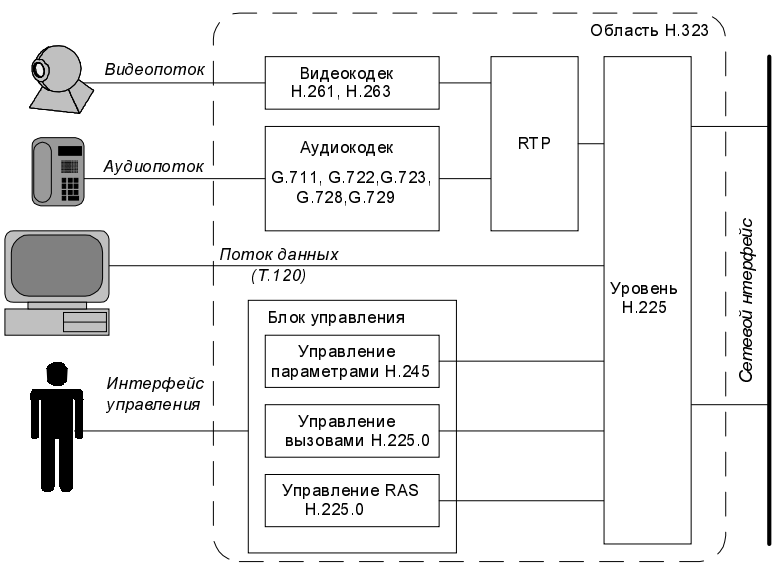


Рис. 7.3. Функциональная схема взаимодействия терминала H.323 с источниками внешней информации.

Компоненты и функции терминальных устройств H.323

В состав терминального устройства H.323 могут входить следующие функциональные блоки:

- Video Codec — видеокодек (кодер-декодер);
- Audio Codec — аудиокодек;
- Data Channel — канал передачи данных;
- System Control Unit — блок управления системой.

Видеокодеки терминалов H.323

Видеокодеки относятся к необязательным компонентам терминального устройства H.323. Они предназначены для выполнения взаимного преобразования видеопотока в последовательность блоков данных сетевого уровня. Для видеотерминала H.323 обязательной является поддержка режима QCIF (Quarter Common Intermediate Format — сокращенный универсальный формат представления изображения) рекомендации H.261. В дополнение к этому видеотерминалы H.323 могут поддерживать любые режимы, определенные рекомендациями H.261 и H.263. Также допускается использование других форматов представления данных видеопотока, согласованных при помощи процедуры H.245. В качестве дополнительной возможности видеотерминала H.323 может быть организован режим одновременной передачи и приема нескольких видеопотоков.

Основные параметры входящего видеотрафика: скорость передачи видеоданных и формат изображения, приемлемые для декодирующего устройства терминала, а также использование дополнительных алгоритмов должны быть согласованы при помощи процедуры H.245. При этом кодирующее устройство удаленного видеотерминала может для представления видеоданных использовать любой из режимов, поддерживаемых декодером. Видеодекодер может инициировать процедуру установления конкретного режима формирования видеотрафика, однако адекватное реагирование на такой запрос не является обязательным для видеокодера удаленного терминала H.323.

Видеотерминалы H.323 должны обеспечивать возможность приема и передачи асимметричных видеопотоков с различными значениями скорости передачи данных и форматом изображения. В том случае, если видеотерминал обеспечивает возможность приема нескольких видеопотоков, в качестве дополнительной возможности может быть использован режим смешивания видеоизображений.

Аудиокодеки терминалов H.323.

Аудиокодеки (голосовые) также являются необязательными компонентами терминального устройства H.323. Они предназначены для выполнения

взаимного преобразования аудиопотока в последовательность блоков данных сетевого уровня. Для голосового терминала обязательной является возможность кодирования и декодирования аудиопотока в соответствии с требованиями рекомендации G.711. При этом кодирующие и декодирующие устройства терминала должны обеспечивать нелинейное кодирование и декодирование речевого сигнала по схемам A-law и μ-law.

Примечание

Схемы нелинейного кодирования A-law и μ-law используются соответственно в европейской и американской телефонных системах для компенсации воздействия шума квантования на передаваемый сигнал.

Также допускается использование других алгоритмов представления данных аудиопотока, согласованных при помощи процедуры H.245.

Терминал H.323. должен обеспечивать использование асимметричных режимов, когда приемник и передатчик применяют различные схемы преобразования аудиопотока. Формат аудиопотока должен соответствовать требованиям, изложенным в рекомендации H.225.0. Пакеты, содержащие закодированные данные аудиопотока, должны передаваться протоколу транспортного уровня с периодом, определяемым типом используемой кодирующей схемы (audio frame interval). Терминалы H.323 могут обеспечивать одновременный прием нескольких голосовых потоков. В этом случае на терминале может быть использован дополнительный режим, обеспечивающий смешивание принимаемых потоков.

Для обеспечения временной взаимной синхронизации аудио- и видеопотоков терминалы H.323 должны обеспечивать возможность формирования и передачи сообщения "h2250MaximumSkewIndication", определяющего максимальное значение временного рассогласования аудио- и видеопотоков одного направления.

Голосовой кодек G.711

В рекомендации ITU-T G.711 приведены принципы реализации и требования к основным характеристикам кодирующего и декодирующего устройств, использующих алгоритм классической импульсно-кодовой модуляции для преобразования аналогового сигнала. Диапазон изменения оцифрованного сигнала составляет 8 бит, преобразование производится с тактовой частотой 8 кГц. Таким образом, скорость потока данных на выходе кодирующего устройства составляет 64 Кбит/с (8 бит × 8 кГц). Для снижения влияния шума квантования на сигналы с небольшой амплитудой при кодировании используется нелинейное преобразование входного аналогового сигнала по схемам A-law или μ-law.

Благодаря простоте реализации кодек G.711 широко распространен в системах традиционной телефонии с коммутацией каналов. Однако кодеки, построен-

ные в соответствии с требованиями рекомендации G.711, достаточно редко применяются в современных системах интегрированной передачи данных из-за высоких требований, предъявляемых к пропускной способности используемого канала. Применение кодирующих схем, отвечающих требованиям рекомендации G.711, можно считать обоснованным лишь в тех случаях, когда требуется обеспечить максимальное качество кодирования речевой информации при небольшом числе одновременных разговоров.

Голосовой кодек G.723.1

В рекомендации ITU-T G.723.1 приведены принципы реализации и требования к основным характеристикам кодирующего и декодирующего устройств, использующих для преобразования голосовой информации технологию многоимпульсного многоуровневого квантования (Multy Pulse-Multy Level Quantization — MP-MLQ). Кодирующие устройства G.723.1 выполняют преобразование аналогового сигнала в поток данных со скоростью 64 Кбит/с по стандартному алгоритму импульсно-кодовой модуляции, а затем при помощи многополосного цифрового фильтра выделяют в сформированном сигнале частотные фонемы и формируют значение результирующего выходного кода, определяемое текущим состоянием фонем во входном аналоговом сигнале. Использование подобного алгоритма кодирования позволяет снизить скорость передачи кодированного голосового сигнала информации до 5,3—6,3 Кбит/с без ощутимого ухудшения качества речи.

Кодирующие схемы ITU-T G.723.1 обеспечивают возможность применения двух вариантов кодирования. Первый вариант предполагает использование кодирующего алгоритма MP-MLQ и обеспечивает формирование цифрового потока с интенсивностью не более 6,3 Кбит/с. Второй вариант предполагает использование кодирующего алгоритма линейной экстраполяции с кодовым инициированием (Code Excited Linear Prediction — CELP) и обеспечивает формирование цифрового потока с интенсивностью не более 5,3 Кбит/с. Алгоритм MP-MLQ предназначен для использования в сетях с пакетной передачей голоса и обеспечивает лучшее качество кодирования по сравнению с вариантом CELP. Однако этот алгоритм относительно меньше приспособлен к использованию в сетях со смешанным типом трафика (голос/данные).

Голосовой кодек G.726

В рекомендации ITU-T G.726 приведены принципы реализации и требования к основным характеристикам кодирующего и декодирующего устройств, использующих для преобразования голосовой информации технологию адаптивной дифференциальной импульсно-кодовой модуляции (АДИКМ). Применение этой технологии обеспечивает формирование цифрового потока с интенсивностью 32 Кбит/с, 24 Кбит/с и 16 Кбит/с. Кодирующие схемы, построенные в соответствии с требованиями рекомендации ITU-T G.726, могут применяться совместно со схемами G.711 для повышения эффективности использования сетевых ресурсов.

Голосовой кодек G.728

В рекомендации ITU-T G.728 приведены принципы реализации и требования к основным характеристикам кодирующего и декодирующего устройств, использующих для преобразования голосовой информации технологию линейной экстраполяции с кодовым инициированием и малыми задержками (Low Delay Code Excited Linear Prediction — LD-CELP).

Метод основан на предсказании аналогового сигнала в сочетании с элементами кодирования формы сигнала. При использовании этого алгоритма процесс аналого-цифрового преобразования выполняется в несколько этапов. После выполнения операции кодирования аналогового сигнала производится его декодирование, восстановление аналогового сигнала и сравнение восстановленного сигнала с входным сигналом. Для обеспечения наибольшего соответствия производятся изменения параметров кодирующей схемы, после чего выполняется повторное аналого-цифровое преобразование. При достижении оптимального результата кодирующее устройство формирует очередное значение выходного кода. Использование кодирующих устройств, построенных в соответствии с требованиями рекомендации G.728, обеспечивает формирование цифрового потока с интенсивностью 16 Кбит/с.

Голосовой кодек G.729

В рекомендациях ITU-T G.729 приведены принципы реализации и требования к основным характеристикам кодирующего и декодирующего устройств, использующих для преобразования голосовой информации технологию алгебраической экстраполяции с кодовым инициированием (Conjugate Structure-Algebraic Code Excited Linear Prediction — CS-ACELP).

Рекомендация G.729 Annex (раздел) B предполагает использование дополнительной функции — определение активности голосового канала (Voice Activity Detector — VAD). Применение этой функции обеспечивает существенное повышение эффективности использования канала передачи данных, поскольку в данном случае аналого-цифровому преобразованию предшествует амплитудное нормирование входного сигнала. В том случае, если амплитуда этого сигнала не превышает порогового значения, аналого-цифровое преобразование этого сигнала не производится. При использовании данной процедуры паузы на приемной стороне заполняются комфортным шумом, который формирует специальный генератор.

Использование кодирующих устройств, построенных в соответствии с требованиями рекомендации G.728, обеспечивает формирование цифрового потока с интенсивностью 8 Кбит/с.

Канал передачи данных терминала H.323

Канал передачи данных не является обязательным компонентом терминалов H.323 и предназначен для обеспечения информационного обмена между абонентами терминалов с использованием следующих режимов и функций:

- File exchange — обмен файлами;

- Electronic whiteboard — электронная доска;
- Database access — доступ к базам данных.

В зависимости от требований, предъявляемых реализуемым приложением, может быть создан однонаправленный или двунаправленный канал передачи данных терминалов H.323.

Рекомендация T.120 определяет установленный по умолчанию базовый набор требований по обеспечению взаимного информационного обмена между терминалом H.323 и терминалами других типов (например, H.324). Формат данных, передаваемых по этому каналу, определен в рекомендации H.225.0.

Соединение T.120 создается при организации терминального вызова и является его частью. Для обеспечения передачи данных по этому каналу любая из взаимодействующих сторон может сформировать управляющее сообщение OpenLogicalChannel. Для обеспечения аутентификации сторона, инициирующая создание логического канала, должна передать в этом сообщении свой транспортный адрес (Transport Address). В том случае, если принимающая сторона согласна на установление логического канала с инициатором, она формирует ответное сообщение OpenLogicalChannelAck.

Созданный таким образом логический канал может быть использован как для обеспечения информационных потребностей пользовательских приложений, так и для формирования команд управления для удаленного устройства.

Функции управления H.245 терминала H.323

Изменение параметров терминала H.323 по протоколу H.245 выполняется путем передачи специальных сообщений по управляющему каналу H.245. Управляющий канал H.245 может быть организован между двумя терминалами H.323, между терминалом и привратником или между терминалом и контроллером конференций. Рекомендация H.245 определяет набор сообщений, обеспечивающих следующие режимы управления:

- Master/Slave Determination — определение ведущего и подчиненного устройства;
- Capability Exchange — обмен информацией о возможностях устройства;
- Logical Channel Signaling — сигнализация логического канала;
- Mode Request — запрос текущего режима;
- Round Trip Delay Determination — определение задержки распространения сигнала.

Для управления параметрами терминала H.323 используются четыре категории управляющих сообщений H.245.

1. Request — запрос параметра.
2. Response — ответное сообщение.

3. Command — команда управления.
4. Indication — информационное сообщение.

Сообщения Request и Response используются для обмена оперативной информацией между взаимодействующими сторонами. Сообщение Response обязательно должно быть сформировано в ответ на полученное сообщение Request. Сообщение Command передается для инициирования указанных процедур удаленной стороной. Передача ответных сообщений на сообщение Command не предполагается. Сообщения Indication носят чисто информативный характер. При получении этого сообщения обычно не требуется выполнения никаких действий или формирования ответного сообщения.

Примечание

Сообщение Indication рекомендации H.245 может быть использовано для организации вспомогательного канала передачи информации пользователя — по аналогии с тоновыми сигналами, которые используются для управления в телефонной сети общего доступа. В этом случае при нажатии клавиши или кнопки терминала передается соответствующее сообщение, которое может быть должным образом интерпретировано ответной стороной.

Функции управления RAS терминала H.323

Для выполнения функций управления доступом (Registration, Admission and Status — RAS) терминалы и привратники H.323 используют сообщения H.225. Для передачи этих сообщений применяется специальный канал передачи (RAS signaling channel), не зависимый от канала, используемого для управления вызовами, или логического канала передачи данных. Сообщения управления, передаваемые по этому каналу, обеспечивают выполнение привратником всего комплекса операций по обслуживанию зоны вызовов — установление подлинности, авторизация, учет трафика и т. д.

Функции управления вызовами терминала H.323

Для выполнения функций управления вызовами терминалы H.323 используют сообщения H.225. Для передачи этих сообщений также применяется отдельный от всех описанных ранее логический канал. Сообщениями управления вызовом могут обмениваться между собой два терминала H.323 или терминал H.323 со шлюзом H.323.

Компоненты и функции шлюза H.323

Как уже отмечалось выше, шлюз H.323 обеспечивает взаимодействие компонентов системы преобразования потоковых данных с внешними сетями общего пользования. Назначение шлюза состоит в выполнении прямого и обратного преобразования между форматами представления данных и коммуникационными процедурами взаимодействующих систем; при выполнении этих функций шлюз H.323 может создавать и разрывать вызовы.

Минимальный набор операций, которые должен выполнять шлюз H.323, включает в себя:

- преобразование формата передаваемых данных;
- преобразование процедур организации вызова;
- преобразование сигналов и процедур управления.

Шлюзы могут быть использованы для обеспечения взаимодействия H.323 терминалов через телефонную сеть общего пользования PSTN(Public Switching Telephone Network), как это показано на рис. 7.4.

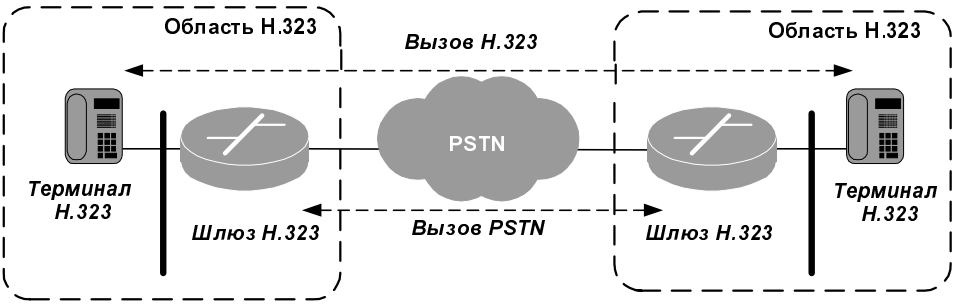


Рис. 7.4. Взаимодействие терминалов H.323 через сеть PSTN с использованием шлюзов H.323

На приведенной схеме представлен вариант организации информационного взаимодействия терминалов H.323 через коммутируемую сеть общего пользования (PSTN). В данном случае терминалы осуществляют взаимное преобразование сигнализации PSTN и сигнализации управления вызовом H.323. При этом каждый из шлюзов выполняет роли двух терминалов — терминала PSTN и терминала H.323.

В том случае, если шлюз не может правильно интерпретировать сигнализацию, используемую некоторым вызовом, он должен пропустить такой вызов в сеть без изменений.

Компоненты и функции привратника H.323

Привратник не является обязательным компонентом комплексов H.323. Задачей привратника является выполнение вспомогательных функций при организации вызовов терминалами H.323. Хотя привратник представляет собой логически обособленное устройство, его функции может выполнять по совместительству любой из имеющихся функциональных компонентов комплексов H.323 (терминал, шлюз и т. д.).

Несмотря на то, что количество устройств, выполняющих функции привратника, не ограничено, только одно из них может считаться активным

привратником данной зоны и выполнять соответствующие действия. Все остальные привратники считаются вспомогательными.

Активный привратник H.323 обеспечивает выполнение следующих основных и дополнительных функций:

- Address Translation — трансляция адресов;
- Admissions Control — управление доступом;
- Bandwidth Control — управление пропускной способностью;
- Zone Management — обслуживание Зоны;
- Call Control Signaling — управление сигнализацией вызовов;
- Call Authorization — авторизация вызовов;
- Bandwidth Management — обслуживание сетевого ресурса;
- Call Management — обслуживание вызовов;
- Alias Address Modification — модификация адресов.

Функция трансляции адресов привратника H.323

Функция трансляции адресов относится к основным функциям привратника H.323. При выполнении этой функции привратник должен осуществить преобразование внутреннего (alias) адреса в транспортный адрес. Для выполнения этого преобразования привратник должен использовать таблицу соответствия, содержимое которой периодически обновляется при помощи специальных процедур.

Функция управления доступом привратника H.323

Функция управления доступом относится к основным функциям, исполняемым привратником H.323. Для управления доступом к сети привратник использует сообщения ARQ — Admission Request (Запрос доступа), ACF — Admission Confirmation (Подтверждение доступа) и ARJ — Admission Reject (Отказ в доступе) рекомендации H.225.0. В качестве критериев аутентичности при организации дисциплины доступа могут применяться разнообразные параметры. Допускается применение нуль-дисциплины, при использовании которой доступ ничем не ограничивается.

Функция управления пропускной способностью привратника H.323

Функция управления пропускной способностью относится к основным функциям, исполняемым привратником H.323. Для управления пропускной способностью привратник использует сообщения BRQ — Bandwidth Change Request (Запрос изменения пропускной способности), BCF — Bandwidth Change Confirmation (Подтверждение изменения пропускной способности) и BRJ — Bandwidth Change Reject (Отказ в изменении пропускной способности) рекомендации H.225.0. Допускается применение нуль-дисциплины,

при использовании которой все запросы на изменение пропускной способности удовлетворяются.

Функция обслуживания зоны привратника H.323

Функция обслуживания зоны относится к основным функциям, исполняемым привратником H.323. Эта функция заключается в выполнении всех описанных выше функций для терминалов H.323 и шлюзов, прошедших специальную процедуру регистрации в зоне.

Функция управления сигнализацией вызовов привратника H.323

Функция управления сигнализацией вызовов относится к вспомогательным функциям привратника H.323. При выполнении данной функции привратник выбирает один из возможных режимов организации вызова абонентом системы и при необходимости обеспечивает выполнение этого режима. В некоторых случаях привратник может полностью взять на себя управление организацией вызова у терминала своей зоны.

Функция авторизации вызовов привратника H.323

Функция авторизации вызовов относится к вспомогательным функциям привратника H.323. Исполнение этой функции заключается в отклонении вызовов, инициированных абонентами, не прошедшими удовлетворительно процедуру авторизации.

Функция обслуживания сетевого ресурса привратника H.323

Функция обслуживания сетевого ресурса является вспомогательной функцией привратника H.323. Исполнение этой функции заключается в отклонении вызовов, сформированных абонентами в условиях перегруженности сети.

Функция обслуживания вызовов привратника H.323

Функция обслуживания вызовов является вспомогательной функцией привратника H.323. При выполнении этой функции привратник отклоняет вызов, если согласно имеющимся у него данным вызываемый абонент уже занят.

Функция модификации адресов привратника H.323

Функция модификации адресов является вспомогательной функцией привратника H.323. Выполнение этой функции состоит в изменении направленного ему внутреннего адреса. После внесения этого изменения вызывающая сторона должна использовать исключительно предложенное значение для организации вызова при поддержке привратника H.323.

Протокол RSVP

Протокол RSVP (ReSerVation Protocol) предназначен для резервирования сетевых ресурсов с целью обеспечения заданного уровня QoS в системах

объединенного обслуживания — IntServ. При помощи протокола RSVP конечный узел может запрашивать специальные уровни QoS для выделенных групп источников поступающего к нему трафика. Маршрутизаторы сети Интернет могут использовать этот протокол для обеспечения распространения требований, сформулированных конечным узлом, на всем протяжении маршрута, связывающего этот узел с источником информации. Благодаря тому, что протокол RSVP построен по принципу "от приемника", он может одинаково успешно обеспечивать резервирование ресурсов, необходимых для передачи одноадресного и группового трафика. По своему назначению протокол RSVP относится, наряду с такими протоколами, как ICMP (Internet Control Message Protocol) и IGMP (Internet Group Management Protocol), к классу управляющих протоколов Интернета. Для обеспечения своей функции протокол RSVP активно взаимодействует с протоколами Unicast и Multicast маршрутизации.

Механизмы управления трафиком протокола RSVP

Протокол RSVP позволяет определить и реализовать политику обеспечения качества обслуживания для групп блоков данных, проходящих по некоторому маршруту и называемых потоками. Для управления потоками протокол RSVP использует следующие функциональные модули и механизмы:

- Packet classifier — классификатор пакетов;
- Packet scheduler — планировщик пакетов;
- Admission controller — контроллер ресурса;
- Policy controller — контроллер политики.

Информационный обмен между узлами сети, использующими протокол RSVP для резервирования сетевых ресурсов, осуществляется при помощи специальных сообщений. Сообщения протокола RSVP могут быть сформированы источником потока данных, приемником (или приемниками) этого потока, а также промежуточными узлами сети.

Сообщение "Запрос резервирования" (Reservation Request) используется для распространения по сети информации о потребности в сетевых ресурсах для некоторого информационного потока. Элементарный запрос резервирования включает в себя две спецификации:

- Flow Specification — спецификация потока;
- Filter Specification — спецификация фильтра.

Спецификация потока содержит значения запрашиваемых параметров сетевых ресурсов и предназначается для настройки модулей планировщиков пакетов. В состав спецификации потока входят два набора цифровых параметров.

1. RSpec (Reserve Specification) — содержит запрашиваемые параметры QoS;

2. TSpec (Traffic Specification) — содержит описание потока данных.

Спецификация фильтра содержит информацию, необходимую для настройки классификатора, — критерии принадлежности трафика к выделенной группе.

Схема организации информационного обмена при использовании протокола RSVP приведена на рис. 7.5.

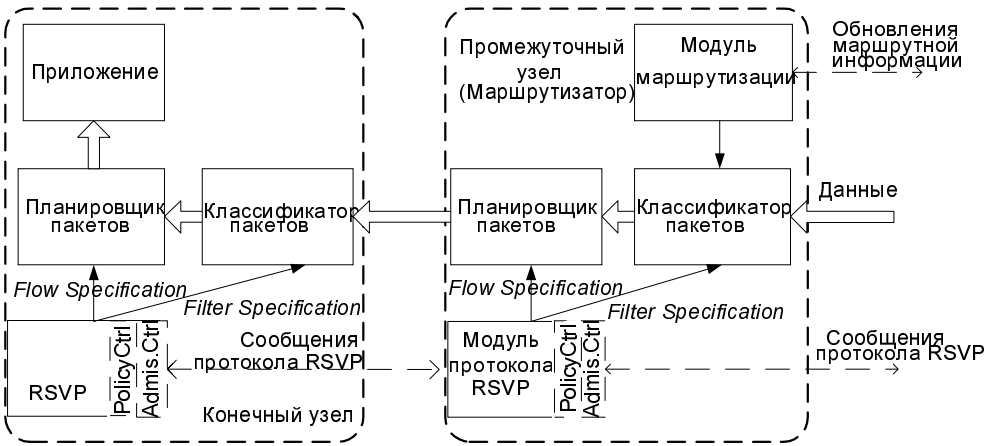


Рис. 7.5. Схема информационного обмена протокола RSVP

Управление информационным потоком, направление которого указано утолщенными стрелками, осуществляется двумя типами модулей — модулями протокола RSVP и модулями маршрутизатора. Инициатором информационного взаимодействия выступает узел, заинтересованный в приеме потока данных. Для определения политики обеспечения качества обслуживания этот узел и маршрутизаторы, через которые пролегает путь, связывающий его с источником потока данных, обмениваются сообщениями протокола RSVP. Этот обмен производится по классическому принципу "Hop-by-Hop". В результате обмена сообщениями на узле и маршрутизаторах осуществляется настройка исполняющих модулей обеспечения качества обслуживания.

В том случае, если сообщение протокола RSVP поступает на промежуточный узел, на этом узле выполняются следующие действия:

- закрепление ресурса канала за выделенным потоком;
- формирование последующего сообщения протокола RSVP.

В свою очередь, процесс закрепления канального ресурса включает в себя две предварительные контрольные стадии. На первой из них устанавливается наличие у источника запроса полномочий на выполнение процедур резервирования на данном узле. В ходе выполнения второй процедуры

устанавливается практическая возможность выделения данным узлом запрашиваемых ресурсов, определяемая, в основном, их наличием на текущий момент. В том случае, если обе описанные выше процедуры были выполнены успешно, на промежуточном узле выполняются действия, обеспечивающие закрепление требуемых ресурсов за выделенным потоком. При этом, в зависимости от типа используемого протокола канального уровня, резервирование может быть выполнено с использованием специфических алгоритмов упорядочивания.

Примечание

Следует учитывать, что собственно резервирование не обеспечивает гарантированный уровень качества обслуживания, а лишь предоставляет возможность для его достижения при использовании соответствующих механизмов.

Назначение классификатора состоит в определении принадлежности к тому или иному классу блоков данных входящего информационного потока. В зависимости от реализуемой политики определение принадлежности к тому или иному классу может производиться на основании анализа значений таких параметров информационного потока, как адрес источника, адрес назначения, тип протокола сетевого и транспортного уровня и т. д. Управление классификатором производится от модуля протокола RSVP и модуля маршрутизации. При этом модуль маршрутизации обеспечивает поступление необходимой маршрутной информации, а RSVP-модуль направляет информацию, необходимую для обеспечения собственно процедуры классификации.

Примечание

Как известно, для обеспечения защиты информации ее источник может использовать процедуры шифрования содержимого заголовка. Очевидно, что в таких случаях для решения задачи классификации должны быть использованы специальные алгоритмы и процедуры.

Назначение планировщика пакетов состоит в определении порядка прохождения на обработку блоков данных сетевого уровня, принадлежащих к различным классам входного потока. Управление планировщиком производится от RSVP-модуля, при этом планировщику передается информация о значениях разрешенных для каждого класса потоков параметров сетевых ресурсов.

Формы и методы резервирования ресурсов

Особенностью протокола RSVP является возможность обеспечения резервирования сетевых ресурсов для передачи группового трафика. Как известно, для передачи трафика этого типа в сети при помощи протоколов групповой маршрутизации создаются виртуальные структуры — деревья распространения

группового трафика. Для повышения эффективности использования сетевого ресурса процедура резервирования, реализуемая протоколом RSVP, располагает возможностью объединения (Merging) резервов сетевых ресурсов, обеспечивающих передачу одного и того же группового трафика в точках ветвления дерева. Собственно резервирование состоит в определении значений пары связанных величин — адреса источника трафика (S) и пропускной способности, требуемой для передачи этого трафика (Q).

Протокол RSVP обеспечивает возможность использования нескольких стилей для формулирования политики резервирования и формирования спецификации фильтра:

- Wildcard-Filter (WF) — стиль резервирования с неопределенным фильтром;
- Fixed-Filter (FF) — стиль резервирования с фиксированным фильтром;
- Shared Explicit (SE) — разделяемый стиль с явным указанием источников.

Стиль Wildcard-Filter (WF)

При использовании стиля резервирования с неопределенным фильтром для формулирования политики управления качеством обслуживания приложение или промежуточный маршрутизатор определяет только требование к значению пропускной способности сети (Q) — WF (*{Q}).

Примечание

Символ "*" в данном случае означает, что резервирование параметров канала для передачи трафика выполняется независимо от того, какой источник (Source — S) сформировал этот трафик.

Создаваемый таким образом резервированный канал обеспечивает передачу всего группового трафика. В том случае, если промежуточный маршрутизатор получает несколько последовательных запросов резервирования WF (*{Q}), он формирует итоговый запрос, передаваемый в направлении источника трафика, используя принцип поглощения меньшего запроса большим запросом:

$$\{WF\ (*\{Q_i\})\}\ i = (1, ..., n) \Rightarrow WF\ (*\{\max(Q_i)\}).$$

На рис. 7.6 представлена схема формирования запросов при использовании стиля Wildcard-Filter.

В данном случае параметр Q запросов, формируемых маршрутизаторами ГМ1 и ГМ4, соответствует максимальному значению Q среди всех запросов на резервирование сетевого ресурса, принятых данными маршрутизаторами, и никак не зависит от типа запрашиваемого трафика.

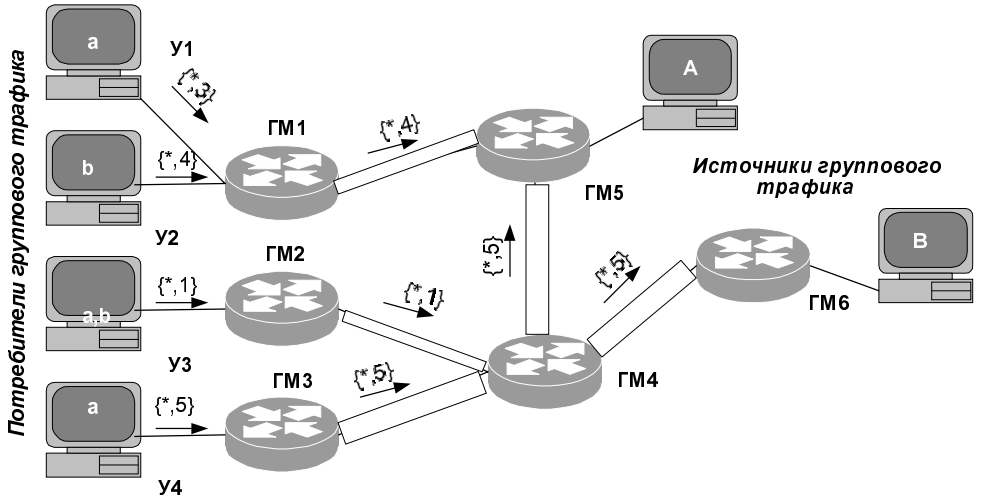


Рис. 7.6. Схема формирования запросов протокола RSVP при использовании стиля Wildcard-Filter

Стиль Fixed-Filter (FF)

При использовании стиля резервирования с фиксированным фильтром для формулирования политики управления качеством обслуживания приложение или промежуточный маршрутизатор явно определяют требования к значению пропускной способности сети для передачи конкретного вида группового трафика FF (S{Q}). Создаваемый таким образом резервированный канал обеспечивает передачу всего группового трафика. В том случае, если промежуточный маршрутизатор получает несколько последовательных запросов резервирования FF (S{Q}), он формирует итоговые запросы, передаваемые в направлении источника трафика, используя принцип поглощения меньшего запроса большим запросом отдельно для каждого источника группового трафика:

$$\{\mathrm{FF}\ (\mathrm{S}_j\{Q_i\})\}\ i = (1, \ldots, n) \Rightarrow \mathrm{FF}_j\ (\mathrm{S}\{\max(\mathrm{Q}_i)\}).$$

На рис. 7.7 представлена схема формирования запросов при использовании стиля резервирования Fixed-Filter.

При использовании этого стиля резервирования промежуточный маршрутизатор ГМ3 формирует суммарный запрос, в котором для источников A и B резервируются максимальные значения соответствующих ресурсов сети (5 и 4 соответственно).

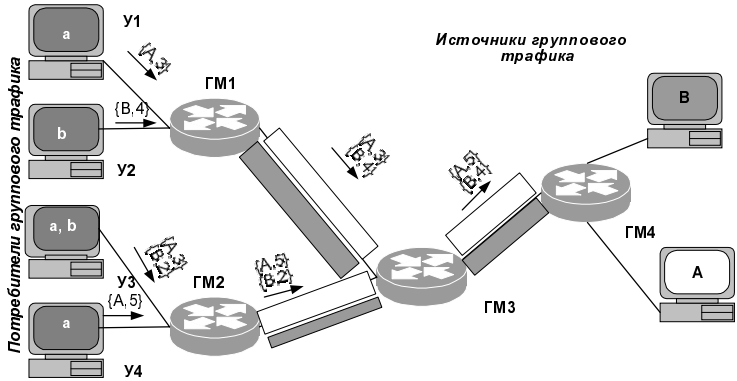


Рис. 7.7. Схема формирования запросов при использовании стиля резервирования Fixed-Filter

Стиль Shared Explicit (SE)

При использовании разделяемого стиля с явным указанием источников приложение или промежуточный маршрутизатор определяют единое требование к значению пропускной способности сети для передачи группового трафика нескольких явно определенных источников SE ($S_1\{Q\}S_j\{Q\}$).

Этот стиль резервирования незначительно отличается от стиля резервирования WF. Отличие состоит в том, что при использовании стиля SE информация об источниках группового трафика накапливается в последующих запросах резервирования по принципу снежного кома. При этом требования к сетевому ресурсу определяются стандартным образом — требование формируемого запроса соответствует максимальному требованию из числа принятых запросов.

На рис. 7.8 представлена схема формирования запросов при использовании стиля резервирования Shared Explicit.

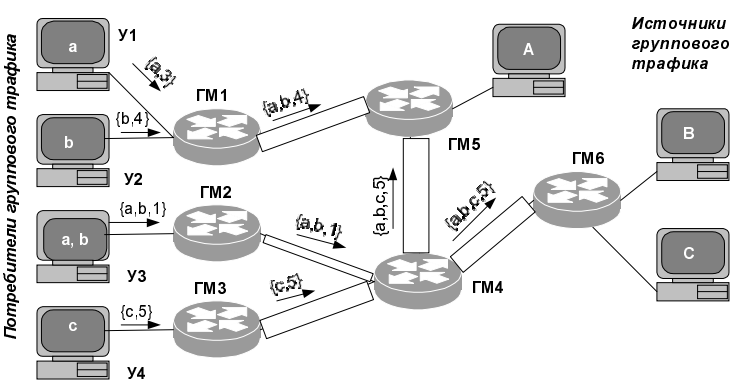


Рис. 7.8. Схема формирования запросов при использовании стиля резервирования Shared Explicit

Сообщения протокола RSVP

Для обеспечения информационного обмена между компонентами в протоколе RSVP используются следующие типы сообщений:

- Reservation-Request — сообщения запросов резервирования;
- Path — сообщения о формировании пути;
- Error — сообщения об ошибках;
- Confirmation — сообщения подтверждения;
- Teardown — сообщения о разрывании пути.

Сообщения Reservation-Request

Каждый конечный или промежуточный узел протокола RSVP, принимающий групповой или одноадресный трафик, должен принимать, обрабатывать и формировать сообщения Reservation-Request, а также передавать сформированные сообщения в направлении источников этого трафика. Сообщения Reservation-Request передаются по пути, проложенному сообщениями Path. Для передачи сообщений Reservation-Request должен быть использован Unicast-адрес ближайшего узла или маршрутизатора. При получении сообщения Reservation-Request промежуточный маршрутизатор переходит в состояние резервирования соответствующего сетевого ресурса. Это состояние является динамическим — в том случае, если маршрутизатор в течение установленного интервала времени не получит сообщений Reservation-Request и Path, подтверждающих необходимость резервирования данного ресурса, он автоматически перейдет в исходное состояние.

Сообщения Path

Сообщения Path формирует каждый источник трафика, а также любой промежуточный узел протокола RSVP, передающий этот трафик по инстанции. Для передачи сообщений Path должен быть использован тот же адрес, который используется для передачи потока данных. Передача сообщений Path предшествует передаче сообщений Reservation-Request. При получении сообщения Path маршрутизатор переходит в состояние хранения пути передачи данных соответствующего источника. Впоследствии этот путь будет использован для передачи по нему потока данных и сообщений Reservation-Request.

Сообщения Error

В протоколе RSVP используются два сообщения об ошибках:

- ResvErr — сообщение об ошибке резервирования;
- PathErr — сообщение об ошибке маршрута.

К сожалению, существует достаточно много самых разнообразных причин, вызывающих формирование сообщения ResvErr. Такое сообщение может быть сформировано в том случае, если полученный запрос резервирования не может быть удовлетворен на данном узле из-за существующих ограничений сетевого ресурса. К аналогичному результату может привести отсутствие у автора запроса надлежащих прав на выполнение операций резервирования на данном маршрутизаторе. К появлению подобного сообщения могут также привести конфликты между несколькими последовательными запросами, поступившими на маршрутизатор.

Сообщения об ошибке маршрута PathErr, в отличие от исходного сообщения, передаются в направлении источника потока данных по принципу "Hop-by-Hop".

Сообщения Confirmation

Требование подтверждения о применении резервирования может быть включено в любой передаваемый приемником запрос резервирования. При наличии такого требования в запросе резервирования источник потока данных должен получить подтверждение о применении резервирования от того узла, на котором этот запрос закончил свой путь по сети. Обычно такое сообщение передает узел, выполнивший операцию слияния запросов.

Сообщения Teardown

Сообщение Teardown формируется для обеспечения немедленного изменения существующего состояния маршрутизатора, хранящего информацию о пути передачи потока данных или резервировании сетевых ресурсов для передачи этого потока.

Для выполнения таких изменений используются сообщения PathTear (Изменение маршрута) и ResvTear (Изменение резервирования), соответственно. Эти сообщения могут быть сформированы любым узлом, находящимся на пути распространения потока данных и распространяются по принципу "Hop-by-Hop".

Формат сообщений протокола RSVP

Формат сообщений протокола RSVP представлен на рис. 7.9.

Поле **Version** предназначено для размещения номера использованной версии протокола RSVP. В настоящее время используется первая версия протокола RSVP.

Поле **Flags** зарезервировано для размещения управляющих признаков протокола RSVP.

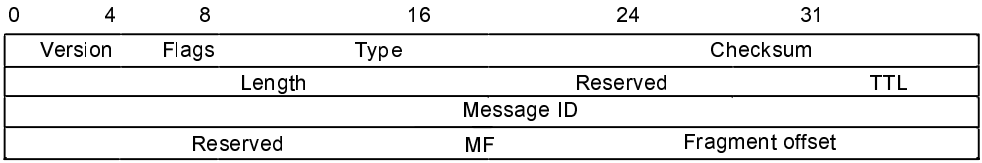


Рис. 7.9. Формат сообщений протокола RSVP

В поле **Type** размещается признак типа используемого сообщения протокола RSVP. Перечень сообщений протокола RSVP и их назначение приведены в табл. 7.1.

Таблица 7.1. *Типы сообщений протокола RSVP*

Тип	Тип сообщения	Назначение
1	Path	Формирование пути передачи трафика
2	Resv	Запрос резервирования сетевого ресурса
3	PathErr	Ошибка пути передачи трафика
4	ResvErr	Неверный запрос резервирования сетевого ресурса
5	PathTear	Разрывание пути передачи трафика
6	ResvTear	Снятие запроса резервирования сетевого ресурса
7	ResvConf	Подтверждение запроса резервирования сетевого ресурса

В поле **Checksum** размещается 16-разрядная контрольная сумма сообщения протокола RSVP.

В поле **Length** размещается выраженная в байтах длина данного сообщения.

Значение, размещенное в поле **TTL,** определяет максимальную величину временного интервала, в течение которого данное сообщение может существовать в сети.

Значение, размещенное в поле **Message ID**, предназначено для обеспечения восстановления исходного сообщения, прошедшего процедуру фрагментации. В этом случае все фрагменты исходного сообщения должны иметь одинаковое значение поля **Message ID**.

В поле **MF** размещается признак, указывающий на положение текущего фрагмента в исходном сообщении. У последнего фрагмента сообщения этот признак устанавливается равным 0.

Значение, размещенное в поле **Fragment offset**, определяет положение текущего фрагмента в исходном сообщении.

Принципы построения и компоненты сетей VoIP

Комплекс технологий VoIP (Voice over IP) представляет реализацию для телефонных сетей общего пользования принципов, сформулированных в рекомендации ITU-T H.323. Применение технологий этого комплекса позволяет обеспечить:

- повышение эффективности использования сетевых ресурсов;
- устранение ненужного дублирования коммуникаций;
- внедрение новых информационных сервисов.

На рис. 7.10 представлен один из возможных вариантов применения технологий VoIP.

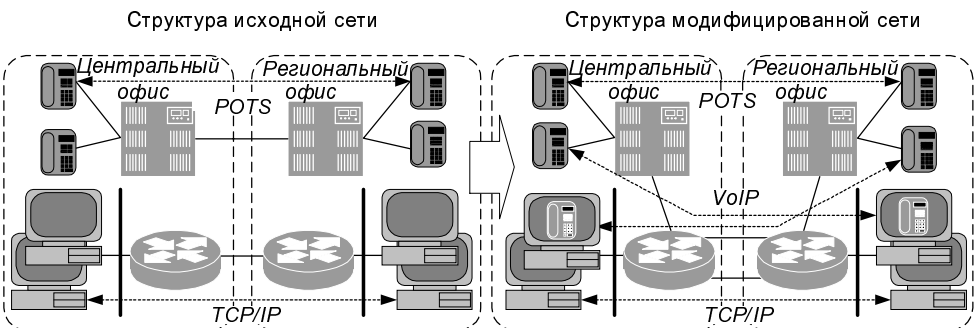


Рис. 7.10. Вариант применения технологий VoIP для модификации структуры сети

В исходном варианте построения сети, функциональная схема которой представлена в левой части рисунка, для связи между центральным и региональным офисом использовались независимые каналы. Один из них связывал офисные АТС, а другой использовался для обеспечения передачи данных между локальными сетями центрального и регионального офиса. При этом ресурсы этих каналов почти наверняка использовались недостаточно эффективно, поскольку телефонный трафик имеет ярко выраженную суточную цикличность. В правой части рисунка представлен вариант построения той же сети с использованием технологий VoIP. При этом офисные АТС подключаются непосредственно к маршрутизаторам, а каналы, связывающие офисы используются только для передачи трафика IP. При

этом голосовой трафик передается в сжатом виде, что обеспечивает повышение эффективности использования каналов передачи данных. Динамическое распределение пропускной способности единого канала позволяет рационально учитывать суточную неравномерность телефонного трафика. Таким образом, в ночное и вечернее время вся пропускная способность канала может быть выделена для передачи данных в фоновом режиме.

Интегрирование в одном канале трафика разных приложений позволяет снизить расходы на их аренду, приобретение каналообразующего оборудования, его сопровождение и обслуживание.

Дополнительным преимуществом новой технологии является возможность внедрения новых информационных услуг. В модифицированной сети, схема которой представлена на правой части рис. 7.10, дополнительно к имеющимся сервисам передачи данных с использованием протоколов TCP/IP и телефонии Plain Old Telephone Service (POTS) могут быть реализованы специфические услуги VoIP, например, вызов с IP-телефона абонента POTS и наоборот.

Компоненты и алгоритмы систем VoIP

В соответствии с требованиями рекомендации ITU-T H.323, при построении систем VoIP могут быть использованы следующие функциональные блоки:

- пользовательские терминалы;
- шлюзы для телефонных сетей общего пользования;
- привратники VoIP;
- программные системы управления компонентами VoIP.

Терминальные устройства VoIP

Назначением терминала VoIP является обеспечение непосредственного доступа абонента в сеть IP. Поэтому терминальное устройство VoIP выполняет прямое и обратное преобразование голосового сигнала в последовательность пакетов протокола IP.

Терминал VoIP состоит из двух основных функциональных блоков:

- блок кодирования-декодирования речевого сигнала (кодек);
- блок сопряжения с локальной сетью.

В начале данной главы дано краткое описание основных алгоритмов, которые могут быть использованы для оцифровки голосового аналогового сигнала. В табл. 7.2 приведены характеристики кодеков, наиболее часто используемых при построении терминальных устройств в системах VoIP.

Таблица 7.2. *Характеристики кодеков, применяемых в системах VoIP*

Обозначение	Размер пакета		Информационная скорость (Кбит/с)
	Диапазон	Значение по умолчанию	
G.711	40—240	160	64
G.723.1	24—240	24	6.3
G.723.1 (53)	20—240	20	5.3
G.726-32.	20—240	40	32
G.726-24.	15—240	30	24
G.726-16.	10—240	20	16
G.728.	10—240	10	16
G.729.	10—240	20	8

В настоящее время используется несколько вариантов реализации терминалов VoIP:

- ❒ отдельные устройства (IP-телефоны);
- ❒ программные реализации (программы-приложения);
- ❒ гибридные схемы (адаптеры для аналоговых телефонов).

Особенности применения аппаратных терминалов VoIP

В том случае, когда терминал VoIP выполнен в виде отдельного устройства, он обычно называется IP-телефоном. Такие телефоны, помимо основных атрибутов, присущих обычным телефонам (трубка, наборное поле), имеют также интерфейс для непосредственного подключения к локальной сети. Такие IP-телефоны, как правило, обладают всеми возможностями и функциями, присущими оконечному устройству сети Интернет, и в дополнение к ним обеспечивают многочисленные возможности по созданию и обслуживанию адресных баз абонентов. Поскольку процедуры кодирования-декодирования аналогового сигнала в IP-телефонах выполняются аппаратно-программным способом, выбор требуемого кодека может быть выполнен достаточно оперативно.

Современные IP-телефоны обязательно поддерживают протокол динамического конфигурирования DHCP, что существенно облегчает процедуру инсталляции и обеспечивает требуемую мобильность. Для удаленного управления IP-телефоном могут быть использованы протоколы Telnet или HTTP. В последнем случае настройка и конфигурирование IP-телефона могут быть выполнены при помощи стандартной программы Web-browser (например, Internet Explorer Microsoft Corporation).

IP-телефоны могут обладать следующими дополнительными возможностями, улучшающими их эксплуатационные характеристики:

- внутренний повторитель сети IEEE 802.3;
- электропитание IP-телефона по IEEE 802.3.

Внутренний сетевой повторитель IP-телефона позволяет размещать аппараты практически в любом месте сети, причем для этого не требуется никаких дополнительных приобретений. Аппарат IP-телефона, например, может быть подключен к сетевому порту, занимаемому прежде персональным компьютером. При этом персональный компьютер подключается к внутреннему повторителю IP-телефона. Кроме того, если IP-телефон допускает электропитание по сетевому кабелю от коммутатора, для установки такого аппарата не потребуется даже наличия свободной розетки электропитания.

Программные реализации терминалов VoIP

При программной реализации терминала VoIP его функции обычно выполняет персональный компьютер стандартной комплектации. При этом для акустического взаимодействия с абонентом используются звуковая карта и микрофон персонального компьютера. Подобный способ реализации терминала VoIP является наиболее дешевым, поскольку не требует приобретения дополнительного оборудования. Функцию кодека в программных терминалах VoIP выполняет специальное программное обеспечение, оперирующее на прикладном уровне OSI ISO. Пользовательский интерфейс программных терминалов VoIP во многом напоминает интерфейс обычного телефонного аппарата. Совместное использование программных терминалов VoIP и развитых систем IP-телефонии позволяет осуществлять их гибкое интегрирование с системами голосовой почты и обеспечивает проведение аудиоконференций. Организация и прием телефонных вызовов в данном случае выполняется средствами персонального компьютера, что может в некоторых случаях быть недостаточно удобным для оператора.

Гибридные терминалы VoIP

Для реализации терминала VoIP также могут быть использованы разнообразные гибридные схемы, сочетающие в себе достоинства аппаратной и программной реализации тех или иных функциональных компонентов. В частности, терминал VoIP может включать в себя стандартный телефонный аппарат и блок сопряжения — адаптер, выполняющий функцию сопряжения этого аппарата с локальной сетью. Один адаптер также может быть использован для подключения нескольких аналоговых телефонных аппаратов.

Подобная схема построения терминала VoIP обеспечивает максимальное удобство для пользователя, поскольку в этом случае пользовательский интерфейс полностью сохраняется, однако при этом для обеспечения возмож-

ности применения дополнительных сервисов VoIP требуется установка вспомогательного программного обеспечения на персональный компьютер абонента.

Принципы построения и компоненты шлюзов VoIP

Шлюзы VoIP предназначены для обеспечения взаимодействия компонентов систем Интернета — телефонии с внешними телефонными сетями общего пользования POTS (Plain Old Telephone Service). Основные функции, выполняемые шлюзом VoIP, могут включать:

- обеспечение физических интерфейсов с внешними сетями;
- распознавание и формирование соответствующих сигналов абонентской сигнализации;
- прямое и обратное преобразование формы представления сигналов VoIP и PSTN;
- формирование и разрывание межсетевых вызовов.

Для организации интернациональных вызовов шлюз должен обеспечивать формирование и распознавание сигналов специфических систем сигнализации, используемых в различных странах.

В зависимости от состава выполняемых функций, шлюз VoIP может быть либо выполнен в виде отдельного устройства, либо быть программно реализованным.

Поскольку большая часть функций шлюза выполняется процессами прикладного уровня, его реализация может потребовать значительных вычислительных ресурсов центрального процессора. Поэтому целесообразным следует признать такой вариант реализации шлюза VoIP, когда некоторые из его функций выполняются аппаратно.

К таким функциям следует, в первую очередь, отнести описанную ранее функцию преобразования аналогового телефонного сигнала в цифровую форму. Особенность выполнения этой функции шлюзом состоит в том, что это устройство должно обеспечивать одновременное преобразование десятков поступающих и передаваемых информационных потоков. Для обеспечения эффективного и качественного сопряжения с существующими системами POTS шлюз должен решить задачи подавления эхо-сигналов и передачи тональных сигналов набора номера. Эхо-сигнал возникает из-за паразитного взаимного просачивания звуковых сигналов, поступающих на дифференциальную систему, осуществляющую прием-передачу данных по двухпроводному абонентскому каналу. Сформированные таким образом паразитные эхо-сигналы проходят обработку в системе преобразования и поступают обратно к говорящему абоненту, что может привести к существенному ухудшению разборчивости принимаемой речи. Особенно сильное влияние на качество принимаемого сигнала оказывается в том случае, если задержка

распространения звукового сигнала от источника к приемнику и обратно превышает 50 мс.

Телефонный сигнал с двухпроводной линии поступает на дифференциальную систему, которая разделяет приемную и передающую часть канала. Далее принимаемый сигнал вместе с "просочившейся" частью передаваемого сигнала поступает на вход аналого-цифрового преобразователя. Сформированный этим преобразователем цифровой код поступает на блок компенсации эхо-сигнала. В устройстве эхо-компенсации из принимаемого сигнала передачи вычитается ослабленный и задержанный передаваемый сигнал. Эхо-компенсатор представляет собой адаптивный фильтр, объем памяти (количество разрядов памяти) которого и механизм адаптации определяются требованиями рекомендации G.165 (ITU-T).

Для обнаружения и определения Multi-Frequency (MF) — многочастотных сигналов телефонной сигнализации, двухтональных многочастотных сигналов Dual-Tone Multi-Frequency (DTMF) либо сигналов импульсного набора используются детекторы соответствующих типов.

Блок определения голосовой активности (VAD — Voice Activity Detector) определяет, является ли текущий анализируемый фрагмент сигнала речью или паузой. При обнаружении паузы сформированный информационный кадр может не передаваться для дальнейшей обработки. В зависимости от установленного режима на сеансовый уровень может передаваться лишь каждый пятый кадр оцифрованной паузы. Приемная сторона при этом получает признак кадра паузы. На паузных кадрах вместо речевого синтезатора включается генератор комфортного шума, который восстанавливает спектральный состав паузного сигнала. Параметры генератора обновляются при получении очередного информационного кадра паузы.

В табл. 7.3 представлены наиболее распространенные варианты возможного подключения оборудования шлюза VoIP к телефонной сети общего пользования POTS.

Таблица 7.3. *Варианты подключения шлюза VoIP к POTS.*

Соединение	Сигнализация	Назначение
Аналоговая соединительная линия (СЛ)	FXO (Foreign exchange Office)	Двухпроводная соединительная линия для подключения офисной АТС к районной АТС
Аналоговая абонентская линия (АЛ)	FXS (Foreign Exchange Station)	Двухпроводная абонентская линия с сигналом вызова и тональным или импульсным набором номера
E&M	E&M	Двух- или четырехпроводный интерфейс, по функциональности аналогичный FXO

Таблица 7.3 *(окончание)*

Соединение	Сигнализация	Назначение
G.703 E1	R1/5	Цифровой несимметричный поток
E1 PRI G.703	PRI ISDN	Цифровой 30-канальный симметричный поток
E1 R2 G.703	R2	Цифровой 30-канальный симметричный поток

Принимая во внимание совокупность требований, предъявляемых к шлюзу VoIP, следует признать целесообразным его построение на базе модульного маршрутизатора. При этом будет обеспечена возможность использования всего перечня необходимых внешних интерфейсов и достаточная для обработки многоканальных вызовов производительность центрального процессора.

Функции привратника VoIP

Назначением привратника VoIP является преобразование адресов и распространение адресной информации по сети, а также выполнение других вспомогательных процедур, обеспечивающих организацию вызовов.

В сети Интернет привратник VoIP обеспечивает обработку запросов, поступающих от терминалов, выполняющих внешние вызовы. При получении такого запроса привратник либо формирует ответное сообщение, содержащее необходимую для организации внешнего вызова информацию (IP-адрес шлюза, например), либо формирует вызов самостоятельно, выполняя функцию Proxy.

Для выполнения своих функций привратник создает и обслуживает информационную базу локальной номерной зоны. Кроме того, привратник VoIP накапливает информацию о внешних зонах, информацию о которых он получает от других привратников.

Пример практического применения технологии VoIP

В качестве примера практического применения технологии VoIP рассмотрим схему сети, представленную на рис. 7.11.

В сети представленной конфигурации обеспечивается возможность организации телефонного вызова абонента VoIP с любого аппарата учрежденческой АТС (PBX) по внутреннему номеру 1990. В данном случае в качестве абонента VoIP используется персональный компьютер с установленным программным обеспечением Net Meeting Microsoft Corp (кодек G.711/A-law). Ниже приведен фрагмент конфигурационного файла сервера доступа AS5300 Cisco Systems Corp.

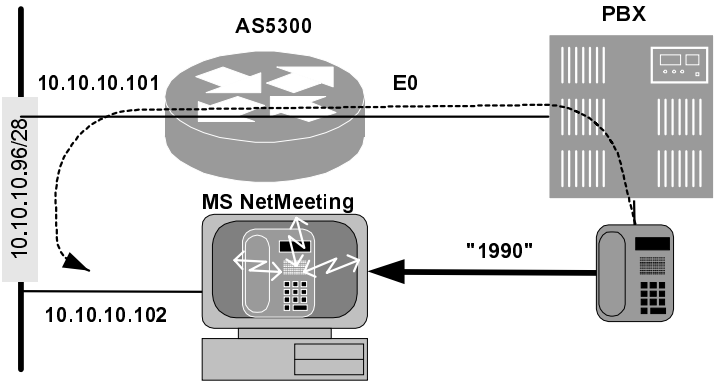
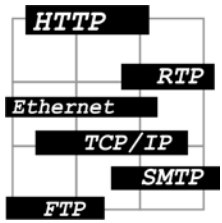


Рис 7.11. Схема организации вызова POTS — VoIP

Фрагмент конфигурации AS5300

```
!
 controller E1 0
 clock source line secondary 1
 pri-group timeslots 1-31
!
 interface Serial0:15
 no ip address
 isdn switch-type primary-net5
 isdn incoming-voice modem
 isdn send-alerting
 fair-queue 64 256 0
 no cdp enable
!
 dial-peer voice 1790 voip
 destination-pattern 1990
 session target ipv4:10.10.10.102
 codec g711alaw
!
```

Глава 8



Обеспечение информационной безопасности в сетях TCP/IP

Сегодня любая современная компания в той или иной степени использует информационные сервисы сети Интернет. Это может быть электронная почта, корпоративный WWW-сервер, службы доступа удаленных клиентов и так далее. Однако наряду с большими преимуществами использование глобальной сети может породить серьезные опасности для этой компании. Причиной является открытость сети Интернет, которая изначально не содержала никаких дополнительных средств, обеспечивающих безопасность информационного обмена. Вследствие этого злоумышленники могут относительно легко и незаметно вторгаться во внутренние сети корпораций, перехватывать и фальсифицировать отправляемые сообщения, нарушать функционирование WWW-серверов. Благодаря разветвленности и глобальному характеру сети Интернет, перехват и вторжение могут быть осуществлены в любой точке земного шара. Резкое повышение производительности процессоров и бурное развитие системного и прикладного программного обеспечения персональных компьютеров позволяют сегодня даже не очень квалифицированному взломщику успешно преодолевать барьеры систем обеспечения информационной безопасности, реализованные на прикладном уровне. Бурное развитие электронного бизнеса делает незаконные операции в сети особенно привлекательным и для преступников, поскольку такие махинации могут обеспечить практически мгновенный результат и анонимность.

Таким образом, в условиях быстрого развития сети Интернет особое значение приобретают комплексы средств, обеспечивающих безопасность информационного обмена. Особенно важно, чтобы комплексы этих средств были максимально прозрачны для пользователя и позволяли применять привычные сервисы и приложения на новом уровне информационной безопасности без потери скорости и удобства их использования.

Комплекс протоколов, носящий условное название IP Security (IPsec), предназначен для решения задач обеспечения безопасности информационного обмена в сети Интернет. Поскольку протоколы IPsec функционируют на сетевом уровне OSI ISO, это обеспечивает их полную совместимость с многочисленными существующими приложениями. Эти протоколы, в частности, обеспечивают шифрование содержимого пакетов, передаваемых

по сети, процедуры обмена ключей и аутентификации абонента. При необходимости, для повышения быстродействия преобразования информации часть алгоритмов IPsec может быть реализована аппаратно.

Технология MPLS (Multi Protocol Label Switching) предназначена для построения закрытых частных сетей внутри сети общего пользования Интернет. Благодаря тому, что в сетях MPLS перемещение пакета выполняется в соответствии с его меткой, а не адресом, информация частных сетей, передаваемая через Интернет, может быть полностью зашифрована.

Комплекс протоколов IPsec

Комплекс процедур и протоколов IPsec, повышающий уровень информационной безопасности сети Интернет, был построен благодаря усилиям одноименной рабочей группы IETF (Internet Engineering Task Force). Важной особенностью этого комплекса является то, что системы обеспечения информационной безопасности, построенные на его основе, могут быть масштабируемыми и наращиваться по мере роста потребности в их использовании.

Схемы защиты информации

Следует отметить, что весь комплекс средств IPsec предназначен для обработки исключительно блоков данных сетевого уровня — пакетов протокола IP. Только такое решение способно одновременно обеспечить открытость и наращиваемость создаваемых информационных систем. В тех случаях, когда комплексы защиты информации реализуются на канальном или прикладном уровне, создаваемые системы являются либо негибкими, либо недостаточно прозрачными и эффективными. Схемы возможных вариантов реализации систем информационной безопасности приведены на рис. 8.1—8.3.

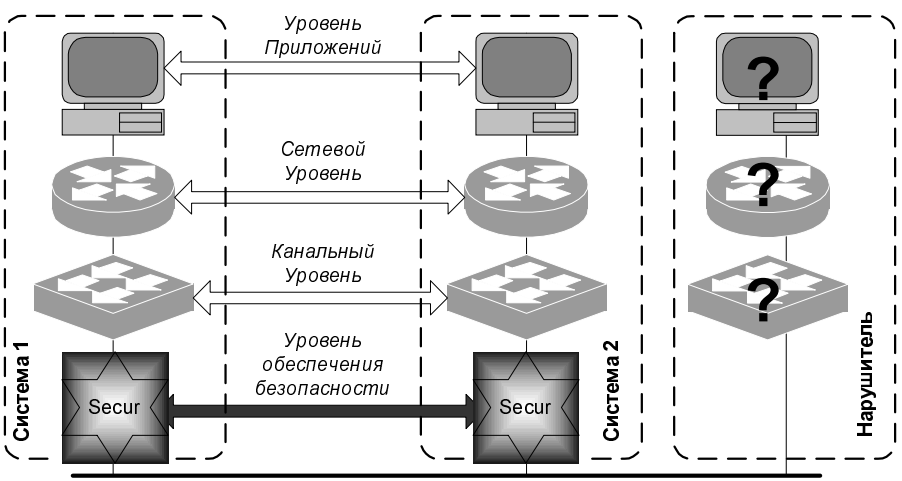


Рис. 8.1. Обеспечение информационной безопасности на физическом уровне OSI

На рис. 8.1 представлена схема реализации системы информационной безопасности на физическом уровне OSI. В данном случае обеспечивается максимальный уровень защиты информации, поскольку потенциальный нарушитель без преодоления этой системы не сможет получить абсолютно никаких данных о взаимодействующих системах 1 и 2. Кодирование информации при использовании этой схемы обычно производится аппаратно, что обеспечивает экономию ресурсов информационных систем. Приведенная схема является прозрачной для всех уровней информационного взаимодействия, начиная с канального уровня и выше. Существенным недостатком такой схемы, однако, является ее недостаточная гибкость. Действительно, в том случае, если потребуется наращивать комплекс путем внедрения новых систем, в первую очередь необходимо будет развивать уровень обеспечения безопасности. Недостатками такой схемы является невозможность использования арендуемых канальных и сетевых сервисов передачи данных и неудобство администрирования.

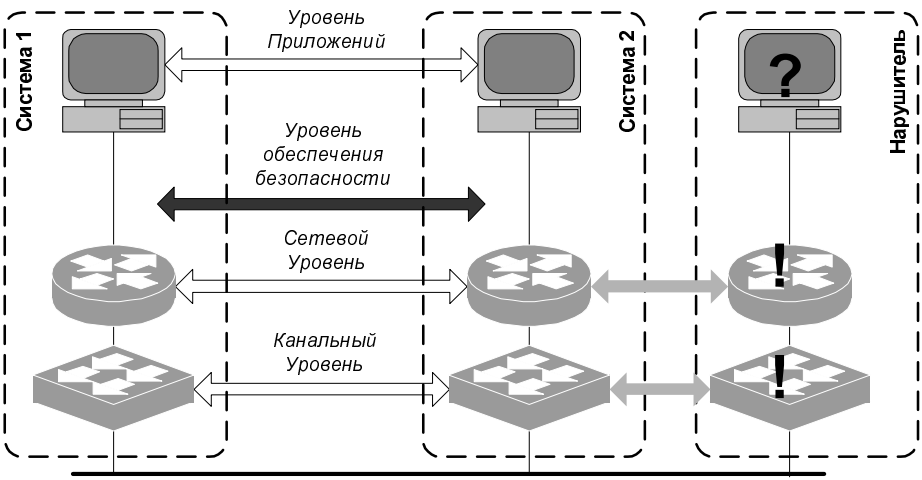


Рис. 8.2. Схема реализации системы информационной безопасности на уровне приложений

На рис. 8.2 представлена схема реализации системы информационной безопасности на уровне приложений. Такая схема применяется для защиты информационного взаимодействия одного или нескольких приложений. Достоинством подобной схемы следует признать простоту ее реализации и возможность наращивания реализуемых функций. Однако поскольку данная схема не обеспечивает полной защиты информационного потока между взаимодействующими системами, возможность ее применения представляется весьма ограниченной. Нарушитель в данном случае может контролировать параметры информационного обмена на канальном и сетевом уровне, что позволяет ему тем или иным образом вмешаться в этот обмен. К тому

же рост числа приложений, обладающий индивидуальными средствами информационной защиты, без сомнения, приведет к существенному усложнению администрирования всей системы в целом.

Наиболее целесообразной следует признать реализацию системы обеспечения информационной безопасности на сетевом уровне OSI. При этом пользователь сохраняет все преимущества информационного обмена с использованием сервисов сети Интернет, так же как и возможность применения каналов арендованных общего доступа для организации сетевой инфраструктуры. При использовании этого метода в сети общего доступа создаются защищенные области, компоненты которых могут взаимодействовать как между собой, так и с остальной сетью. Доступ к сетевым ресурсам защищенных областей ограничен пределами этих областей благодаря применению специальных алгоритмов шифрования и процедур доступа. Системы обеспечения информационной безопасности, реализованные на сетевом уровне OSI, наиболее просты для администрирования. Также достаточно легко в этом случае производится и масштабирование подобных систем. На рис. 8.3 представлена схема реализации системы информационной безопасности на сетевом уровне.

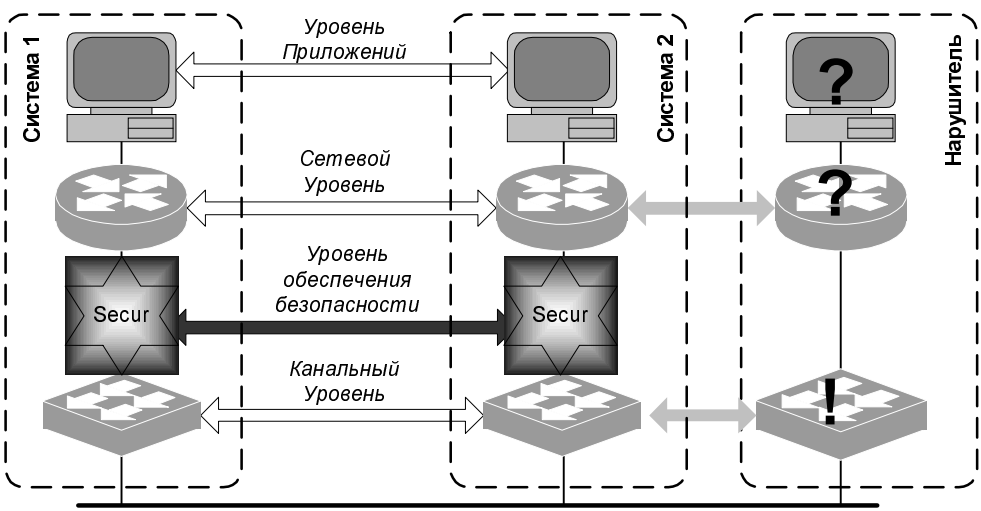


Рис. 8.3. Схема реализации системы информационной безопасности на сетевом уровне.

Несмотря на то, что при реализации системы безопасности на сетевом уровне нарушитель может перехватывать и анализировать блоки данных сетевого и канального уровней, его возможности по их декодированию и фальсификации могут быть существенно ограничены благодаря использованию мощных и эффективных алгоритмов шифрования. Для обеспечения еще большего уровня информационной безопасности могут быть использо-

ваны туннели, соединяющие взаимодействующие системы. В этом случае целиком зашифрованный блок данных сетевого уровня защищаемой системы помещается в пакет, направляемый маршрутизатору дальнего конца туннеля. Таким образом, информация сетевого уровня становится недоступной для перехвата. При анализе пакета, перемещающегося по сети, нарушитель может установить только IP-адреса ближнего и дальнего концов туннелей.

Несмотря на очевидные преимущества, приведенная выше схема все-таки не свободна от некоторых недостатков. В частности, передача информации сетевого и транспортного уровней по сети Интернет в закодированном виде делает невозможным использование многих полезных и эффективных механизмов, направленных на управление информационными потоками и обеспечение гарантированного качества обслуживания.

Соединения безопасности IPsec

Комплекс IPsec обеспечивает возможность одновременного использования нескольких протоколов для обеспечения безопасности информационного взаимодействия между узлами сети. Для выполнения этой задачи каждый из протоколов обеспечения безопасности, как правило, должен быть индивидуально настроен. Таким образом, возникает необходимость в создании и обслуживании базы данных протоколов обеспечения безопасности, в которой хранятся ключи и остальные параметры, согласованные при организации защищенного соединения между узлами.

Соединением безопасности (Security Association — SA) называют одностороннее соединение между узлами, которому соответствует согласованный набор параметров протокола обеспечения безопасности. Каждому из соединений безопасности присваивается уникальный идентификатор, который узел использует в качестве ключа для поиска параметров протокола обеспечения безопасности при получении очередного пакета IPsec от соседнего узла.

Соединения безопасности IPsec между двумя сторонами информационного взаимодействия могут быть организованы с использованием одного из двух возможных режимов:

- транспортного режима (Transport Mode);
- туннельного режима (Tunnel Mode).

При использовании транспортного режима шифрованию подвергается только содержимое поля полезной нагрузки IP пакета, следовательно, он может быть доставлен по сети Интернет так же, как и любой незакодированный пакет. Следовательно, транспортное соединение безопасности могут установить между собой любые два узла сети.

При использовании туннельного режима шифрованию подвергается все содержимое IP-пакета вместе с его заголовком. В данном случае доставка за-

кодированного пакета обеспечивается шлюзом, в качестве которого обычно используется маршрутизатор.

Состав комплекса протоколов IPsec

В состав комплекса протоколов IPsec (IP Security) обеспечивающего безопасность информационного взаимодействия в сети Интернет, входят следующие протоколы:

- протокол заголовка аутентификации (Authentication Header — AH);
- протокол защиты полезной нагрузки пакета протокола IP (Encapsulating Security Payload — ESP).

Хотя формально протоколы управления ключами шифрования (Key Management) не являются частью комплекса IPsec, некоторые из них будут упомянуты в этом разделе, поскольку протоколы IPsec взаимодействуют с этими протоколами достаточно тесно.

При реализации задач комплекса протоколов IP Security могут быть также использованы стандартные алгоритмы аутентификации и шифрования данных.

Протокол заголовка аутентификации AH

Протокол аутентификации AH (Authentication Header) предназначен для обеспечения защиты информации, размещенной в заголовке IP-пакета, от фальсификации нарушителем. Защита информации в данном случае обеспечивается благодаря размещению в IP-пакете дополнительных полей, содержащих контрольную информацию для заголовка переносящего пакета. При формировании этой информации используются ключи, значения которых известны только отправителю и получателю пакетов. Если нарушитель попытается исказить содержимое заголовка и направить фальсифицированный пакет в адрес получателя, такой пакет будет легко выявлен, вследствие несовпадения контрольной информации заголовка аутентификации протокола AH и содержимого заголовка фальсифицированного пакета.

Структура заголовка аутентификации протокола AH

На рис. 8.4 представлена структура заголовка аутентификации протокола AH.

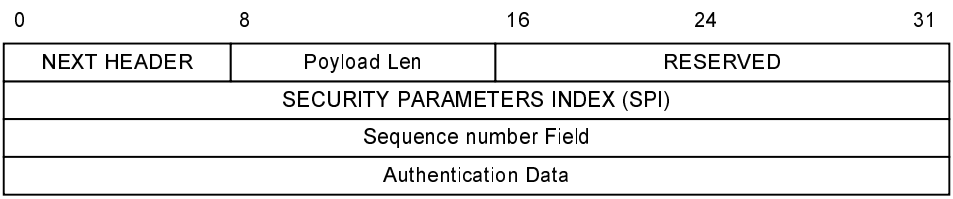


Рис. 8.4. Структура заголовка аутентификации протокола AH

В поле **NEXT HEADER** (следующий заголовок), занимающем один байт в заголовке аутентификации протокола AH, помещается информация о типе следующего протокола полезной нагрузки пакета. Этим протоколом может быть, в частности, один из протоколов транспортного уровня сети Интернет или один из протоколов, обеспечивающих совместно с протоколом AH информационную безопасность в сети.

В поле **Payload Len** (длина поля нагрузки), также занимающем один байт в заголовке аутентификации протокола AH, помещается информация о размере заголовка аутентификации, выраженная в 32-разрядных словах. В режиме отсутствия защиты, который может быть использован только в целях проверки функционирования системы, это поле содержит значение 1.

Последующее за описанными выше полями двухбайтовое поле **RESERVED** зарезервировано для дальнейшего использования. Это поле при формировании заголовка аутентификации устанавливается равным 0 и не должно анализироваться приемником.

В поле **SECURITY PARAMETERS INDEX — (SPI)** (индекс параметров безопасности) помещается 32-битовое число, которое в совокупности и адресом назначения, и идентификатором протокола (в данном случае — протокола AH) однозначно определяет номер соединения безопасности для данной дейтаграммы. Значения поля **SPI** от 1 до 255 зарезервированы организацией IANA, значение 0 может быть использовано некоторыми протоколами для указания на отсутствие требований по обеспечению безопасности.

32-разрядное поле **Sequence Number Field** (последовательный номер) предназначено для размещения в нем текущего порядкового номера пакета в пределах данного соединения безопасности. При создании и рестарте соединения безопасности порядковый номер устанавливается равным 0, первый пакет, передаваемый с использованием соединения безопасности, должен иметь последовательный номер 1, следующий — 2, и так далее. Порядковый номер предназначен для обеспечения возможности определения потери пакетов, передаваемых с использованием сервисов этого соединения. При необходимости потерянный пакет может быть передан повторно, при этом ему должен быть присвоен его первоначальный номер.

Поле **Authentication Data** (Данные аутентификации) в заголовке аутентификации протокола AH имеет переменную длину и предназначено для размещения контрольной информации для заголовка исходного пакета протокола IP Integrity Check Value — ICV (Значение контроля целостности).

Размещение заголовка аутентификации

При использовании соединения безопасности в транспортном режиме IP-заголовок аутентификации должен быть размещен непосредственно после заголовка пакета перед заголовком протокола транспортного уровня.

При использовании соединения безопасности в туннельном режиме IP-заголовок аутентификации размещается непосредственно после нового заголовка пакета перед заголовком исходного пакета, поскольку в данном случае он применяется для обеспечения контроля информационной целостности внешнего пакета. На рис. 8.5 приведены схемы размещения заголовков аутентификации при использовании различных режимов установления соединения безопасности.

Рис. 8.5. Схемы размещения заголовков аутентификации в пакете IP

Вычисление контрольного значения заголовка — ICV

Как было указано выше, контрольная информация целостности заголовка — Integrity Check Value — ICV предназначена для обеспечения защиты информации, размещенной в исходном пакете протокола IP, от несанкционированной модификации. Следует, однако, отметить, что некоторые поля заголовка пакета могут изменяться по мере продвижения пакета по сети без какого-либо злого умысла, в соответствии с алгоритмами обработки пакетов протокола IP. Поэтому в протоколе AH все поля заголовка IP-пакета разделены на две большие группы — модифицируемые и неизменные поля.

К модифицируемым полям заголовка пакета протокола IP (см. рис. 1.5), в частности, относятся:

- **TYPE OF SERVICE (TOS)** — поле "Тип сервиса";
- **FLAGS** — поле "Флаги";
- **FRAGMENT OFFSET** — поле "Смещение фрагмента";
- **TIME TO LIVE (TTL)** — поле "Время жизни";
- **HEADER CHECKSUM** — поле "Контрольная сумма заголовка".

Очевидно, что значение перечисленных выше полей у заголовка дейтаграммы, принятой получателем, может сильно отличаться от значений первоначально сформированной дейтаграммы. Приведенное выше утверждение особенно справедливо для поля **TTL**, которое будет изменено в любом случае. Остальные поля могут быть изменены в том случае, если при передаче

по сети дейтаграмма подвергнется фрагментации (**FRAGMENT OFFSET**, **FLAGS**) или изменению первоначально присвоенного уровня качества обслуживания (**TYPE OF SERVICE**).

К неизменным полям заголовка пакета протокола IP относятся:

- **VERS** — (Version) поле "Версия" (в данном случае — 4);
- **HLEN** — (Internet Header Length) поле "Длина заголовка";
- **TOTAL LENGTH** — поле "Общая длина";
- **IDENTIFICATION** — поле "Идентификатор";
- **PROTOCOL** — поле "Протокол" (в данном случае — AH);
- **SOURCE ADDRESS** — поле "Адрес источника";
- **DESTINATION IP ADDRESS** — поле "Адрес назначения".

При расчете значения ICV (Integrity Check Value) протокола AH (Authentication Header) используются только содержимое неизменных полей заголовка дейтаграммы протокола IP. Приемник, получивший дейтаграмму, содержащую заголовок аутентификации, при помощи имеющихся у него ключей определяет контрольное значение для принятого заголовка дейтаграммы и в том случае, если это значение совпадает со значением, размещенном в поле **Authentication Data**, дейтаграмма принимается для дальнейшей обработки. В противном случае дейтаграмма уничтожается без уведомления об этом предполагаемого источника.

Протокол защиты полезной нагрузки ESP

Протокол защиты полезной нагрузки ESP (Encapsulating Security Payload) предназначен для обеспечения защиты от перехвата, несанкционированного копирования и изменения содержимого поля полезной нагрузки дейтаграмм протокола IP, передаваемых через защищаемые соединения протокола IPsec. Протокол ESP может быть использован как отдельно, так и совместно с протоколом AH. В последнем случае обеспечивается наибольший уровень информационной защиты. В зависимости от конфигурации соединения безопасности, протокол ESP может обеспечивать следующие режимы защиты информации:

- Confidentiality — конфиденциальность;
- Data origin authentication — установление подлинности источника данных;
- Connectionless integrity — подтверждение целостности данных;
- Anti-replay service — защита от повторной передачи.

Хотя режим обеспечения конфиденциальности может быть использован независимо от остальных режимов, применение такого решения не реко-

мендуется. В том случае, если не применяются режимы Data origin authentication и Connectionless integrity, у нарушителя сохраняется возможность нанесения ущерба сети атаками DoS (Denial of Service).

Формат блока данных протокола ESP

На рис. 8.6 представлена структура блока данных протокола ESP.

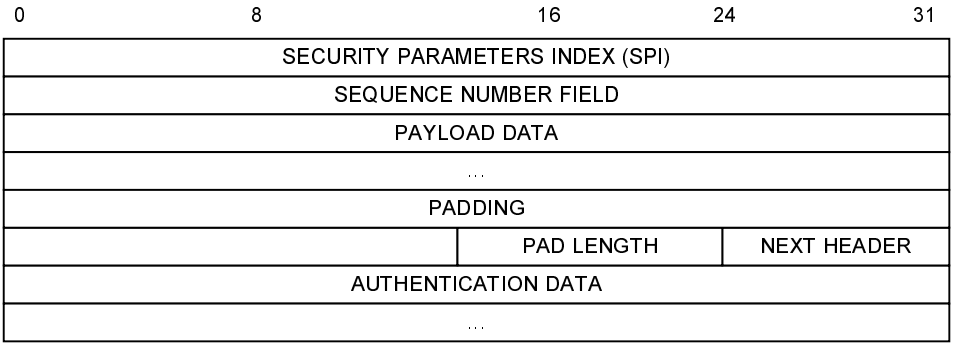


Рис. 8.6. Структура заголовка протокола ESP

В поле **SECURITY PARAMETERS INDEX — (SPI)** (Индекс параметров безопасности) помещается 32-битовое число, которое в совокупности и адресом назначения, и идентификатором протокола (в данном случае — протокола ESP) однозначно определяет номер соединения безопасности для данной дейтаграммы. Значения поля **SPI** от 1 до 255 зарезервированы организацией IANA. Поле **SPI** относится к числу обязательных полей протокола ESP.

Поле **SEQUENCE NUMBER** (Последовательный номер) также относится к числу обязательных полей и предназначено для размещения в нем текущего порядкового номера пакета. При создании и рестарте соединения безопасности порядковый номер устанавливается равным 0, первый пакет, передаваемый с использованием соединения безопасности, должен иметь последовательный номер 1, второй — 2 и так далее. Порядковый номер предназначен для обеспечения возможности определения потери пакетов, передаваемых с использованием сервисов этого соединения.

В поле переменной длины **PAYLOAD DATA** (Данные полезной нагрузки) блока данных протокола ESP размещается закодированная информация поля полезной нагрузки исходной дейтаграммы. В этом поле может также размещаться информация, обеспечивающая восстановление первоначального вида дейтаграммы при ее декодировании. Поле **PAYLOAD DATA** относится к числу обязательных полей протокола ESP.

Поле **PADDING** (Дополнение) не относится к числу обязательных полей протокола ESP. Содержимое этого поля используется для дополнения до 4 байтовых границ размера предыдущего поля блока данных протокола ESP.

В поле **PAD LENGTH** (Длина дополнения) помещается выраженная в байтах длина поля **PADDING**. Поле **PAD LENGTH** также относится к числу обязательных полей протокола ESP.

В поле **NEXT HEADER** (Следующий заголовок), занимающее один байт в заголовке протокола ESP, помещается информация о типе следующего протокола полезной нагрузки пакета. Этим протоколом может быть, в частности, один из протоколов транспортного уровня сети Интернет или один из протоколов, обеспечивающих совместно с протоколом ESP информационную безопасность в сети.

Поле **AUTHENTICATION DATA** (Данные аутентификации) имеет переменную длину и предназначено для размещения контрольной информации содержимого пакета протокола IP — ICV (Integrity Check Value). Хотя процедура установления подлинности источника информации не является обязательной, ее использование настоятельно рекомендуется разработчиками протокола ESP. Дело в том, что при совместном использовании шифрования и контроля целостности передаваемых дейтаграмм в первую очередь должна выполняться именно аутентификация, гарантирующая контроль целостности принятой информации. Несмотря на кажущееся дублирование функций, выполнение отбраковки фальсифицированных дейтаграмм именно на этом этапе является экономически оправданным, поскольку требует для своей реализации существенно меньше вычислительных ресурсов, чем прямое декодирование "сырого" входного потока.

Размещение блока данных протокола ESP в IP-пакете

При использовании соединения безопасности в транспортном режиме блок данных протокола ESP должен быть размещен непосредственно после заголовка пакета перед заголовком протокола транспортного уровня.

При использовании соединения безопасности в туннельном режиме блок данных протокола ESP должен быть размещен непосредственно после нового заголовка пакета перед заголовком исходного пакета, поскольку в данном случае он используется для обеспечения контроля информационной целостности внешнего пакета. На рис. 8.7 приведены схемы размещения заголовков протокола ESP при использовании различных режимов установления соединения безопасности.

На рис. 8.7 одинарной штриховкой отмечены поля, которые подвергаются шифрованию при использовании протокола ESP. Двойной штриховкой на этом рисунке отмечены поля, которые аутентифицируются этим протоколом.

Примечание

Блок Хвост ESP (Trailer ESP) в совокупности образуют поля **PADDING** и **PAD LENGTH**.

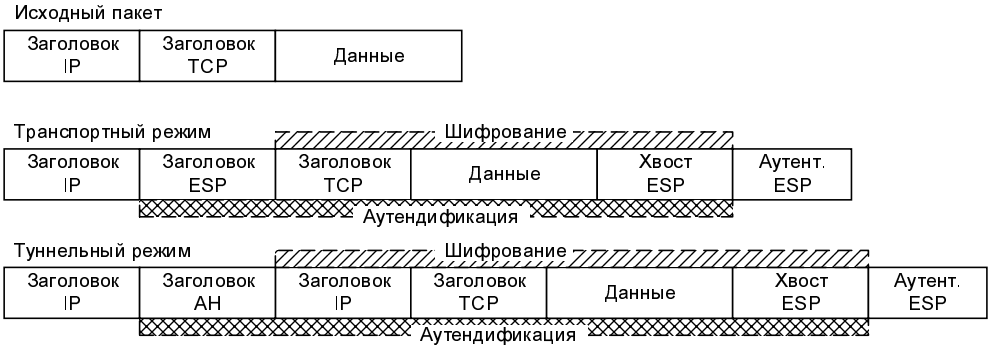


Рис. 8.7. Схемы размещения заголовков протокола ESP в IP-пакете

Процедура обработки порядковых номеров пакетов

Процедура проверки порядковых номеров пакетов предназначена для предотвращения перегрузки узлов защищенной сети копиями адекватных пакетов. Такая перегрузка, в частности, может возникнуть при проведении атаки, в ходе которой нарушитель направляет многочисленные копии перехваченных пакетов на тот узел защищенной сети, куда были направлены их оригиналы. Поскольку каждая из таких копий адекватно отвечает требованиям системы обеспечения безопасности, все они должны быть переданы для обработки на следующий уровень информационного взаимодействия, где и будет обнаружена подделка. Очевидно, что достаточно интенсивный проток копий может легко преодолеть систему безопасности и нарушить нормальное функционирование системы вследствие ее перегрузки ненужным трафиком. Использование входного контроля порядкового номера пакета позволяет существенно снизить возможность успешной реализации подобной атаки и повысить тем самым уровень информационной безопасности системы в целом.

Для обеспечения контроля порядкового номера в протоколе ESP используется принцип "скользящего окна". Рекомендуемый размер окна w находится в диапазоне от 32 до 64 пакетов. Правую границу окна образует максимальный номер принятого адекватного пакета *N*. В том случае, если номер вновь принятого пакета *M* превышает w, этот пакет передается системе безопасности для проверки адекватности ее требованиям. Если пакет был адекватен требованиям системы безопасности, окно сдвигается — правая граница окна w(*r*) устанавливается равной M, левая вычисляется по формуле

$$w(l) = \max[1, (M - 32)].$$

Пакеты, которые не отвечают требованиям системы обеспечения безопасности, уничтожаются без уведомления об этом их предполагаемого источника.

В том случае, если номер вновь принятого пакета находится левее левой границы окна, он тоже должен быть уничтожен, и на этот раз без проведения какого-либо анализа, по принципу: "кто не успел — тот опоздал".

Только пакеты, порядковые номера которых N удовлетворяют соотношению

$$w(l) \le N \le w(r),$$

допускаются к процедуре проверки порядковых номеров. При выполнении этой процедуры устанавливается, не приходил ли ранее пакет с порядковым номером N. Если принятый пакет не является первым среди равных, то он опять-таки уничтожается без уведомления об этом предполагаемого источника. Все остальные пакеты после успешного прохождения остальных проверок системы обеспечения информационной безопасности поступают, наконец, на следующий уровень обработки.

Структура систем обеспечения безопасности IPsec

Выше уже было отмечено, что протоколы IPsec, которые используются для обеспечения защиты информации, в качестве фундамента используют соединения безопасности (Security Association). Эти соединения обеспечивают реализацию политики информационной безопасности, на их основе предоставляются данные необходимые для функционирования протоколов непосредственной защиты информации AH и ESP.

Формирование соединения безопасности производится в два этапа. На первом этапе выполняется процедура установления подлинности абонентов. При условии успешного выполнения первого этапа, на втором между абонентами осуществляется обмен ключами защиты.

Процедуры аутентификации

Основная трудность реализации первой из отмеченных процедур заключается в том, что взаимодействующие стороны находятся, как правило, достаточно далеко друг от друга и изначально между ними не существует защищенных каналов для передачи конфиденциальной информации ключей защиты данных. В таких условиях для аутентификации абонентов могут быть использованы три типа процедур:

- процедура аутентификации с использованием предустановленных ключей;
- процедура аутентификации с использованием открытых ключей;
- процедура аутентификации с использованием цифровой подписи.

Особенность первой из перечисленных процедур заключается в том, что взаимодействующие стороны обладают одинаковым секретным ключом. В этом случае при выполнении процедуры аутентификации стороны передают друг другу закодированное с помощью этого ключа сообщение. В том случае, если обе стороны успешно декодируют переданное сообщение,

процедура аутентификации считается выполненной успешно. Основной недостаток данной процедуры заключается в принципиальном отсутствии в ней механизма оперативного формирования закрытых ключей у потенциальных абонентов.

При использовании процедуры аутентификации с открытыми ключами взаимодействующие стороны используют пару ключей: открытый, который передается абоненту, и закрытый, который формируется случайным образом. Для обмена открытыми ключами в этом случае может быть использован любой из доступных каналов передачи информации. Получив открытый ключ от соседа, каждый из абонентов шифрует им контрольное сообщение, которое может быть восстановлено только с помощью закрытого ключа соседа. В том случае, если оба сообщения успешно восстановятся, процедура аутентификации будет считаться выполненной успешно. Благодаря своей гибкости и эффективности процедура аутентификации с использованием открытых ключей находит большое применение в современных системах обеспечения информационной безопасности, в том числе и в системах на основе протоколов IPsec.

При использовании третьей из перечисленных процедур установление подлинности абонентов осуществляется благодаря использованию цифровой подписи на данных, которыми обмениваются взаимодействующие стороны.

Базы данных систем безопасности

Системы обеспечения информационной безопасности используют два типа баз данных:

- Security Policy Database (SPD) — база политик безопасности;
- Security Association Database (SAD) — база соединений безопасности.

Первая база содержит описание правил, в соответствии с которыми должна производится обработка входящего и исходящего трафика для некоторого объекта сети.

Вторая база содержит данные, которые обеспечивают функционирование активных соединений безопасности.

База политик безопасности

Трафик, передаваемый по защищенным сетям, как правило, неоднороден. Политики безопасности необходимы для того, чтобы обеспечить максимальный уровень информационной защиты именно тем компонентам трафика, которые в этом наиболее сильно нуждаются. Реализация правильно сформулированной политики безопасности наряду с повышением уровня информационной защиты обеспечивает экономию вычислительных ресурсов сети.

При реализации политики безопасности трафик, формируемый и принимаемый узлом, делится на три группы:

- данные, к которым применяются процедуры защиты информации IPsec;
- данные, к которым не применяются процедуры защиты информации IPsec;
- данные, которые подлежат уничтожению.

Поскольку различные компоненты информационного трафика требуют разных степеней и способов защиты информации, политика безопасности представляет собой упорядоченный набор записей, устанавливающих соответствие между значениями разделяющих параметров (Selectors) входящего или исходящего трафика и одним из описанных выше типов процедур обработки трафика. В том случае, если для обработки трафика предполагается использование IPsec-процедур, политика безопасности определяет спецификацию соединения безопасности. В такой спецификации определяются следующие параметры:

- типы протоколов IPsec;
- режимы использования;
- алгоритмы шифрования.

Поскольку при формулировании политики безопасности могут быть использованы метасимволы, определяющие разрешенные диапазоны значений разделяющих параметров, может возникнуть ситуация, когда диапазоны нескольких записей политики будут взаимно перекрываться. Для однозначной интерпретации подобных ситуаций используется следующий алгоритм: в первую очередь реализуется та запись политики безопасности, которая имеет меньший порядковый номер.

При формулировании политики безопасности в качестве разделяющих параметров могут быть использованы следующие параметры трафика.

1. Destination IP Address — адрес назначения. Этот адрес может иметь любой из разрешенных типов сети Интернет (Unicast, Multicast, Broadcast), он может быть задан в виде разрешенного диапазона или стандартного описания внеклассовой сети (маска + сеть).
2. Source IP Address — адрес источника. Этот адрес может также иметь любой из разрешенных типов сети Интернет (Unicast, Multicast, Broadcast), он может быть задан в виде разрешенного диапазона или стандартного описания внеклассовой сети (маска + сеть).
3. Name — имя объекта. При формулировании политики в качестве имени может быть использовано как имя конкретного узла, так и идентификатор системы в целом.
4. Transport Layer Protocol — тип протокола транспортного уровня.

5. Source and Destination Ports — номера портов источника и назначения протокола транспортного уровня.

База соединений безопасности

База данных соединений безопасности Security Association Database (SAD) предназначена для определения параметров активных соединений безопасности. Каждому из таких соединений должна соответствовать одна запись в базе данных SAD. Ссылки на записи базы данных соединений безопасности делаются из базы данных политики безопасности, чем и обеспечивается реализация этой политики. Для однозначной идентификации соединения безопасности используется три его параметра:

- Destination IP address — адрес назначения;
- IPsec Protocol — тип используемого протокола (AH или ESP);
- SPI — индекс параметров безопасности — применяется для того, чтобы различать несколько соединений безопасности, имеющих один адрес назначения и использующих одинаковые протоколы IPsec.

В базе данных политики безопасности могут быть сделаны ссылки на соединения безопасности, не являющиеся активными. В этом случае такое соединение создается сразу же после того, как эта политика будет применена к первому кадру трафика.

Каждому из соединений безопасности соответствует набор параметров, который обеспечивает его корректное функционирование. К ним можно отнести следующие:

- Sequence Number Counter — 32-битное значение, определяющее номер очередного блока данных протокола IPsec (только для исходящего трафика);
- Sequence Counter Overflow — признак обработки переполнения счетчика;
- Anti-Replay Window — 32-разрядный счетчик и 32-битное число, позиционно указывающее на номера уже принятых аутентичных пакетов;
- AH Authentication algorithm, keys — параметры алгоритма аутентификации протокола AH;
- ESP Encryption algorithm, keys — параметры алгоритма кодирования протокола ESP;
- ESP authentication algorithm, keys — параметры алгоритма аутентификации;
- Lifetime of this Security Association — временной интервал существования этого соединения безопасности.

Применение технологии MPLS в сети Интернет

Основным недостатком традиционных систем обеспечения информационной безопасности следует признать отсутствие возможности или неудобство их интеграции с системами обеспечения гарантированного качества обслуживания (QoS). Действительно, во многих случаях задачи этих систем противоречат друг другу или не могут быть решены одновременно. Задачей систем информационной безопасности является максимальное сокрытие информации пользователя, размещенной в информационной части пакета IP. Для обеспечения полноценного функционирования системы обеспечения гарантированного качества обслуживания должна быть доступна практически вся информация о создателе и предназначении каждого блока данных. Одним из возможных путей разрешения этого противоречия является применение специальных технологий, которые способны обеспечить одновременное выполнение этих задач. К таким технологиям, в частности, относится универсальная технология коммутации меток — MPLS (Multiprotocol Label Switching). Использование такой технологии помимо реализации построения системы, сочетающей функции обеспечения информационной безопасности и гарантированного качества обслуживания, предоставляет дополнительные возможности по объединению разнородных инфраструктур передачи данных и позволяет повысить эффективность использования сетевых ресурсов.

Вследствие бурного роста сети Интернет и появления в ней новых информационно насыщенных сервисов современным маршрутизаторам приходится обрабатывать мощные потоки разнородных данных. В этих условиях становится практически невозможным выполнение ими процедур маршрутизации по классической схеме, когда определение следующего узла (Route Decision) осуществляется независимо для каждого пакета. Понятно, что реализовать анализ таблицы маршрутизации, занимающей десятки килобайт для каждого из пакетов, следующих со скоростью десятки Гбит/с, представляется, как минимум, весьма затруднительным.

Для повышения производительности современных маршрутизаторов давно и успешно применяются специальные технологии кэширования. Особенность подобных технологий заключается в том, что Route Decision делается маршрутизатором однократно для всего потока пакетов, связывающих источник и приемник данных. Таким образом, при использовании этих технологий маршрутизатор выполняет функции коммутатора потоков, каждому из которых соответствует одно значение Route Decision. Повышение эффективности маршрутизации обеспечивается за счет того, что после формирования и идентификации потока его коммутация может быть выполнена непосредственно на канальном уровне. Для этого каждому идентификатору потока должен быть поставлен в соответствие адрес канального уровня

маршрутизатора, определенного Route Decision в качестве Next Hop. Очевидно, что в данном случае выполнение процедуры маршрутизации будет осуществляться как минимум на порядок быстрее, поскольку таблица MAC-адресов значительно меньше таблицы маршрутизации по размерам и поиск в ней выполняется простым перебором.

Основные понятия и компоненты MPLS

Благодаря своей гибкости, универсальности и эффективности технология MPLS может быть успешно реализована в сетях, которые используют различные протоколы сетевого и канального уровня. Поэтому в настоящее время институтом IETF (Internet Engineering Task Force) подготовлен ряд документов, определяющих концепцию технологии MPLS и требования к основным ее компонентам.

Одним из основных понятий технологии MPLS является понятие класса однотипной коммутации, а именно Forwarding Equivalence Class (FEC). Этот класс составляют пакеты протокола третьего уровня (в нашем случае — протокола IP), обработка которых на данном маршрутизаторе выполняется однотипно. Для иллюстрации принципов формирования классов однотипной коммутации рассмотрим схему, приведенную на рис. 8.8.

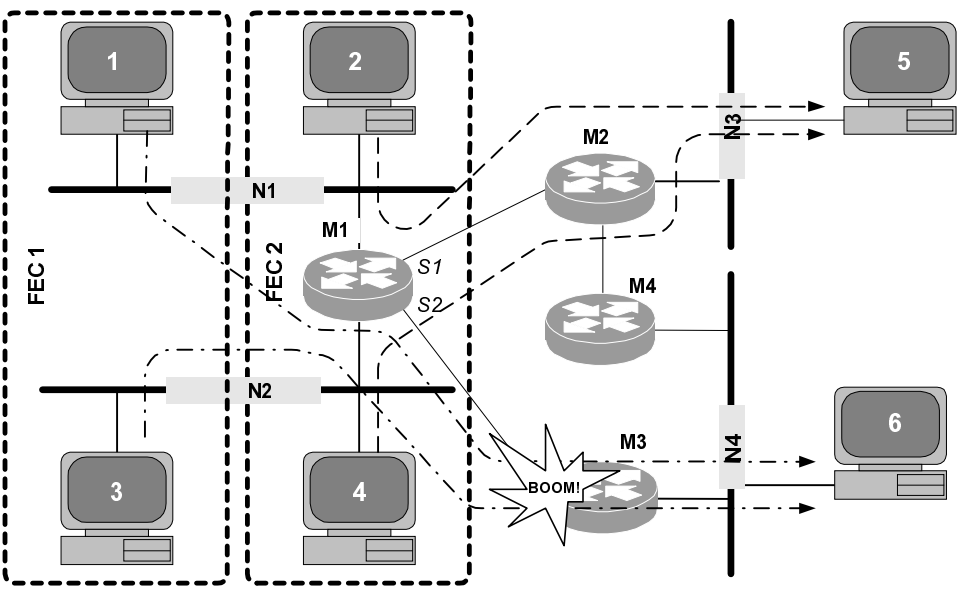


Рис 8.8. Схема сети с двумя определенными классами FEC

На представленной схеме нечетные узлы сетей N1 и N2 взаимодействуют со станцией 6, а четные — со станцией 5 сетей N4 и N3 соответственно.

При этом в маршрутной таблице маршрутизатора M1 должны быть записи, в соответствии с которыми пакеты, адресованные в сеть N3, должны передаваться через интерфейс S1, а пакеты, адресованные в сеть N4, — через интерфейс S2. Критерием, который используется для определения принадлежности к тому или иному классу однотипной коммутации, в данном случае является сетевой префикс — адрес и маска сети. Таким образом, первый класс FEC на маршрутизаторе образуют пакеты, адресованные в сеть N4, а второй — пакеты, адресованные в сеть N3. Маршрутизатору в данном случае нужно проанализировать адрес назначения пакета, определить, к какому классу он относится, и направить его через соответствующий выходной интерфейс. Изменение описанной схемы может произойти лишь в том случае, если по каким-либо причинам произойдут изменения в таблице маршрутизации M1. Предположим, что вышел из строя канал, который связывает, маршрутизаторы M1 и M3. В результате таблица маршрутизации изменится таким образом, что в качестве следующего узла при передаче пакетов как в сеть N3, так и в сеть N4 будет использоваться маршрутизатор M2. Следовательно, в результате этих изменений класс FEC2 исчез как класс, его бывшее население перекочевало в FEC1, зато может появиться новый класс на маршрутизаторе M2.

Описанная выше процедура обеспечивает эффективную маршрутизацию пакетов и, по сути, уже не является собственно процедурой маршрутизации, поскольку выполняется, в основном, на канальном уровне. Однако при использовании описанной схемы процедура классификации — определение принадлежности к тому или иному классу FEC — должна выполняться каждый раз, когда пакет переходит из одного маршрутизатора в другой. Вследствие этого совокупное время выполнения операции существенно увеличивается. Для того чтобы обеспечить выполнение сквозной процедуры коммутации на канальном уровне, необходимо "закодировать" таблицу маршрутизации каждого маршрутизатора. При этом должны выполняться следующие условия:

1. В передаваемом пакете должен быть расположен признак его принадлежности к некоторому классу FEC.
2. Соседние маршрутизаторы должны каким-либо образом установить соответствие между префиксами маршрутов и их закодированными значениями, передаваемыми в пакетах.

Признаки принадлежности пакета к классу FEC, размещаемые в теле пакета, в технологии MPLS принято называть метками (Label). Поэтому маршрутизаторы, которые выполняют процедуры маршрутизации/коммутации с использованием этих меток, называют маршрутизатор-коммутатор меток LSR (Label Switching Router). Замкнутая группа маршрутизаторов LSR образует домен MPLS (Multi Protocol Label Switching). В зависимости от того, какое место в домене занимает маршрутизатор LSR, выполняемые им функции существенно различаются. В функции пограничного маршрутиза-

тора, находящегося на границе домена ELSR (Edge Label Switching Router), входит первоначальная классификация всех входных пакетов. Как уже было отмечено, в результате классификации в тело пакета помещается метка. Если пакет покидает домен MPLS, задачей пограничного маршрутизатора является извлечение метки из этого пакета. Функция внутреннего маршрутизатора домена MPLS заключается только в выполнении процедур коммутации пакетов и замены меток.

Примечание

При выполнении процедуры замены меток маршрутизатор заменяет метку, определившую класс FEC данного пакета в его таблице коммутации, на метку, определяющую этот же класс FEC в таблице коммутации соседнего маршрутизатора, которому будет направлен пакет.

Основные принципы построения MPLS-систем

Основным и главным компонентом систем MPLS является, естественно, маршрутизатор, коммутирующий пакеты. Все процедуры, которые выполняются маршрутизатором MPLS можно разбить на две большие группы:

- задачи, решаемые на уровне управления;
- задачи, решаемые на уровне данных.

Задачи маршрутизатора MPLS, решаемые на уровне управления, обеспечивают взаимодействие между соседними маршрутизаторами MPLS на сетевом уровне.

К задачам, решаемым на уровне управления, относятся:

- формирование и обслуживание таблицы коммутации;
- согласование номеров меток и класса FEC, используемых при обмене с соседними маршрутизаторами.

При формировании и обслуживании таблицы коммутации маршрутизатор MPLS активно использует записи таблицы маршрутизации, построенной с использованием любого протокола маршрутизации или каким-либо другим способом. Маршрутизатор MPLS строит таблицу коммутации динамически, по мере необходимости. При первичном прохождении через маршрутизатор пакетов некоторого класса создается запись в таблице коммутации, которая ставит в соответствие префиксу сети назначения пакета MAC-адрес маршрутизатора Next-Hop. Последующие за первым пакеты, принадлежащие этому же классу, обрабатываются уже только на уровне данных.

На уровне управления решаются также задачи согласования номеров меток и класса FEC с соседними маршрутизаторами. Эта задача решается с использованием специального протокола распределения меток MPLS, а именно протокола LDP (Label Distribution Protocol). Задачами этого протокола, в частности, являются установление и поддерживание партнерских отношений

между маршрутизаторами MPLS, обеспечение аутентифицированного обмена информацией о назначаемых метках, оперативная передача сообщений об изменениях, происходящих в процессе функционирования системы. Протокол LDP сконструирован таким образом, чтобы обеспечивать обмен информацией о назначаемых метках в сетях различной природы и архитектуры.

На уровне данных решаются задачи непосредственной коммутации пакета. При этом в процессе перемещения пакета через домен маршрутизаторами MPLS выполняются следующие действия:

1. Пограничный MPLS маршрутизатор (ELSR) принимает очередной входящий в домен MPLS кадр, классифицирует его, помечает его меткой, определяющей класс FEC этого пакета, и передает его для дальнейшего перемещения.
2. Очередной MPLS маршрутизатор (LSR) принимает пакет, выполняет операцию замены меток и коммутирует этот пакет на один из своих выходных интерфейсов.
3. Пограничный MPLS маршрутизатор (ELSR) принимает очередной выходящий из домена MPLS кадр, убирает из него метку и направляет этот пакет на один из своих выходных интерфейсов.

Структура метки MPLS

Метка, используемая в технологии MPLS, представляет собой 32-разрядное число, размещаемое в теле пакета, для обеспечения передачи его по сетям, использующим технологию MPLS. Метка в кадре размещается непосредственно после заголовка канального уровня, являясь, таким образом, первым полем блока данных сетевого уровня — пакета. Формат метки MPLS представлен на рис. 8.9.

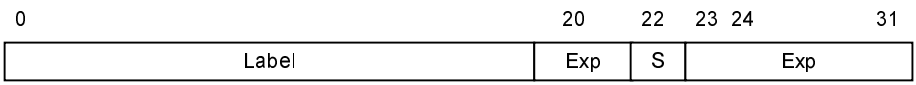


Рис. 8.9. Формат метки MPLS

В поле **Label**, которое имеет размер 20 бит, помещается непосредственно сама метка, значение которой может быть получено с использованием протокола LDP.

Поле **Exp** (Experimental) — экспериментальное и зарезервировано для дальнейшего использования. Некоторые из существующих реализаций предполагают размещение в этом поле признака IP precedence, с помощью которого можно будет связывать с выбранным классом FEC заранее определенный уровень качества обслуживания в сети Интернет.

Поле **S** предназначено для размещения признака того, что данная метка является последней в стеке, и, следовательно, непосредственно после нее размещается заголовок блока данных протокола сетевого уровня (в нашем случае — протокола IP).

Содержимое поля **TTL** определяет максимальное время существования помеченного пакета в сети. При перемещении помеченного пакета по сети, использующей технологию MPLS, значение, первоначально размещенное в этом поле, уменьшается на 1 при прохождении очередного маршрутизатора LSR. Ограничение этого времени обеспечивает уничтожение помеченных пакетов, перемещающихся по циклическим маршрутам.

Стеки меток MPLS

В технологии MPLS предусмотрена возможность многократного маркирования одного и того же кадра различными метками. Эта возможность может быть использована для обеспечения таких дополнительных функций, как организация защищенных частных сетей (Virtual Private Network) или управление трафиком (Traffic Engineering). В качестве иллюстрации рассмотрим пример сети, приведенный на рис. 8.10.

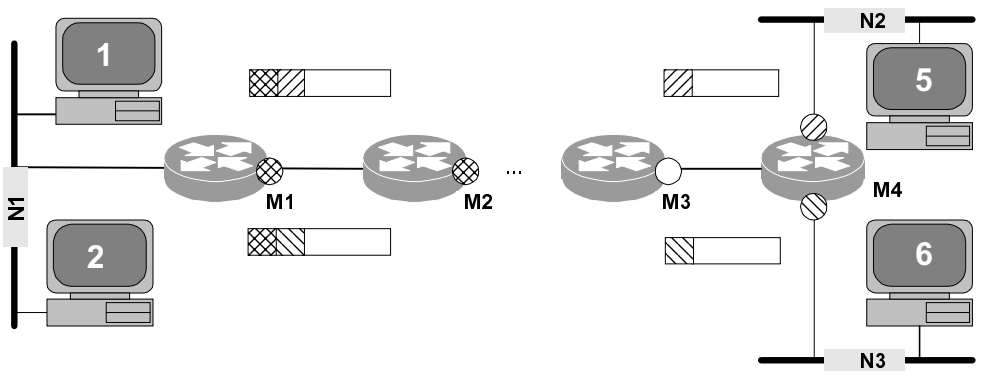


Рис. 8.10. Использование стека меток для управления трафиком в сети MPLS

На схеме представлен вариант построения сети с использованием технологии MPLS. Узлы 1 и 2 сети N1 информационно взаимодействуют с узлами 5 и 6 соответственно. Согласно с принятой схемой формирования меток, пакеты этих узлов получают по две метки, что позволяет им длительное время перемещаться по общему маршруту. Заменяется при этом перемещении только внешняя метка стека. На последнем этапе (маршрутизатор M3) эта метка отбрасывается и маршруты пакетов расходятся.

Принципы построения виртуальных частных сетей

По мере развития своего бизнеса любая достаточно крупная современная компания неизбежно сталкивается с проблемами обеспечения доступа к своим информационным ресурсам и защиты этих ресурсов. Актуальность первой проблемы определяется тем, что современные крупные компании, как правило, имеют много филиалов и офисов. Для обеспечения управляемости и слаженности работы таких компаний в первую очередь необходимо обеспечить оперативное и надежное информационное взаимодействие всех подразделений. При решении этой проблемы, однако, не следует забывать о необходимости обеспечения защиты информации, передаваемой между подразделениями компании. Таким образом, рост конкуренции с одной стороны и развитие средств электронного шпионажа и бандитизма неизбежно подводят современную компанию к дилемме — "развитие или защита". Естественно, большинство современных компаний предпочитает развитие, при этом наиболее крупные из них могут позволить себе использование выделенных линий и каналов передачи данных для объединения филиалов и модемных пулов в целях обеспечения доступа удаленных пользователей во внутреннюю сеть. Однако высокая стоимость и недостаточная эффективность подобных решений заставляет искать альтернативные решения обозначенных проблем. Одним из таких решений является применение технологий виртуальных частных сетей.

Технологии виртуальных частных сетей VPN (Virtual Private Network) предназначены для создания на основе сетей и информационных сервисов общего пользования инфраструктуры, обеспечивающей надежное и безопасное объединение территориально распределенных фрагментов участков сетей и доступа к ним мобильных пользователей. Термин "виртуальная" в данном случае означает, что для создания частной сети используются виртуальные (а не выделенные) каналы, порожденные сетью Интернет. При построении виртуальных частных сетей могут быть использованы самые разнообразные технологии и протоколы. Общим для всех разновидностей современных VPN является применение глобальной сети Интернет в качестве связующего звена при объединении фрагментов сетей. Только в этом случае могут быть удовлетворены все основные требования, которые предъявляются к таким сетям:

- высокий уровень защищенности (Privacy);
- обеспечение возможности развития и наращивания функций (Scalability);
- гибкость, возможность оперативного реконфигурирования (Flexibility);
- эффективность и экономичность (Predictable Performance).

Действительно, развитость и повсеместное распространение сети Интернет делает весьма привлекательным ее использование в качестве инфраструктуры, объединяющей объекты, расположенные в различных уголках земного шара. Существующий институт Интернет-провайдеров подталкивает развитие проводных и беспроводных коммуникаций, что, в свою очередь, приводит к снижению цен на предоставление доступа в сеть. Единственное требование к виртуальной частной сети на основе Интернет, которое в этом случае достаточно трудно реализовать, это требование защиты информации. Несмотря на то, что, на первый взгляд, публичность сети Интернет и приватность сетей VPN кажутся несовместимыми, существуют решения, позволяющие обеспечивать высокий уровень информационной защищенности частных сетей на основе сети Интернет.

Типы виртуальных частных сетей

Прежде чем обсуждать различные способы и концепции построения виртуальных частных сетей, необходимо определить основные типы этих сетей.

Наиболее дешевой и простой для реализации является виртуальная частная сеть на основе коммутируемых каналов доступа в Интернет. Упрощенная функциональная схема подобной сети приведена на рис. 8.11.

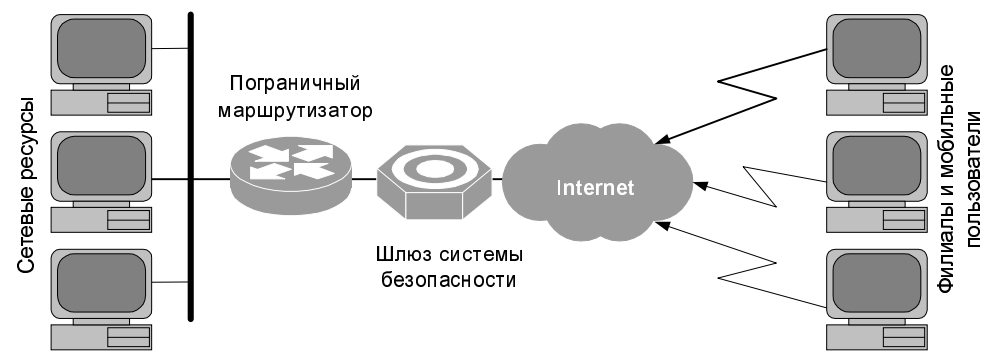


Рис. 8.11. Функциональная схема виртуальной частной сети на основе коммутируемых каналов доступа

При построении подобной сети обеспечивается возможность доступа удаленных филиалов и мобильных пользователей к сетевым ресурсам центрального офиса посредством коммутируемого доступа в сеть Интернет. Как уже было отмечено выше, для построения такой сети VPN требуются минимальные затраты, поэтому подобная схема обеспечивает практически предельные экономичность эксплуатации и гибкость. Существенные недостатки такой схемы определяются типом использованных коммутируемых каналов, которым изначально свойственны:

- невысокая скорость передачи данных;

- низкая надежность;
- непредсказуемый уровень готовности.

Совершенно очевидно, что представленная структура совершенно не в состоянии обеспечить оперативную передачу больших объемов информации, которая может потребоваться, например, для синхронизации баз данных.

Тем не менее, вследствие своей экономичности виртуальные частные сети на основе коммутируемых каналов доступа в Интернет могут вполне удовлетворить потребности небольших развивающихся компаний.

Для относительно крупных компаний, которые могут себе позволить аренду достаточно скоростных выделенных каналов доступа в Интернет, более целесообразным будет признано построение виртуальной частной сети на основе выделенных каналов доступа в Интернет. Упрощенная функциональная схема подобной сети приведена на рис. 8.12.

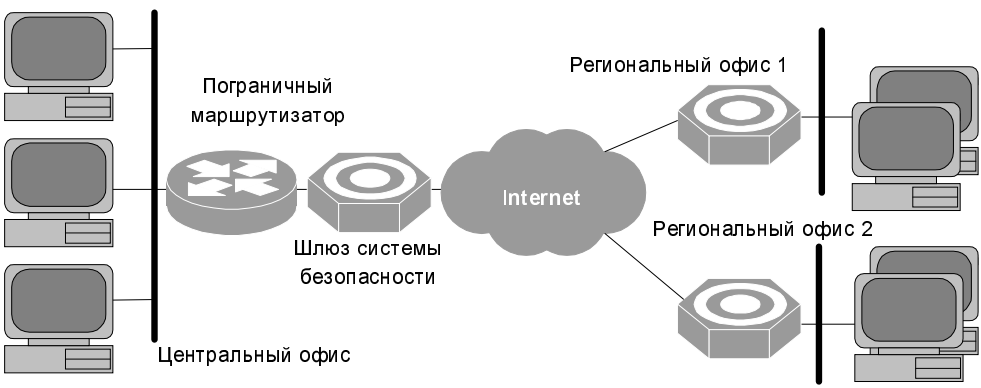


Рис. 8.12. Функциональная схема виртуальной частной сети на основе выделенных каналов доступа

Применение высокоскоростных выделенных каналов при построении виртуальной частной сети не только обеспечивает возможность полноценного функционирования многочисленных приложений "клиент-сервер", но также создания и обслуживания распределенных баз данных. При достаточно высокой пропускной способности арендованных каналов и наличии систем обеспечения гарантированного качества обслуживания, подобная схема позволяет организовывать проведение корпоративных видеоконференций и развивать ведомственную VoIP-телефонию.

Описанные выше схемы не должны рассматриваться в качестве шаблона. В зависимости от индивидуальных особенностей различные компании могут реализовывать гибридные схемы, сочетающие в себе признаки сетей VPN на основе выделенных каналов доступа и сетей VPN на основе коммутируемых каналов доступа в Интернет.

Технологии виртуальных частных сетей

В настоящее время существует два достаточно распространенных подхода к созданию виртуальных частных сетей:

- виртуальные сети на основе технологий IPsec;
- виртуальные сети на основе технологии MPLS.

При использовании первого подхода защита информации реализуется благодаря применению разнообразных алгоритмов шифрования содержимого передаваемых по сети пакетов. Комплекс специальных протоколов обеспечивает также установление подлинности абонента и контроль целостности передаваемых данных. Возможность использования дополнительных процедур управления ключами придает гибкость, управляемость и масштабируемость сетей VPN, построенных на комплексе протоколов IPsec.

Технология MPLS также может быть использована для построения виртуальных частных сетей. Эта технология сама по себе обладает достаточно привлекательными характеристиками для того, чтобы быть выбранной в качестве основы для построения виртуальной частной сети. Как мы помним, перемещение пакета по сети, использующую технологию MPLS, выполняется на основании информации, содержащейся в добавленной к пакету метке. Можно было бы сказать, что в данном случае мы имеем дело с MPLS-тоннелем. Однако в отличие от тоннелей протокола IPsec, перемещение информации в сетях VPN, применяющих технологию MPLS, производится на канальном уровне, что обеспечивает существенно большее их быстродействие.

Виртуальные сети на основе технологий IPsec

Технология IPsec достаточно часто используется для построения виртуальных частных сетей. Применение этой технологии позволяет строить надежные и эффективные частные сети на основе выделенных и коммутируемых каналов доступа. На рис. 8.13 представлены основные компоненты виртуальных частных сетей на основе технологий IPsec.

Основным компонентом виртуальной частной сети на основе технологий IPsec является программный или аппаратный шлюз IPsec — сети VPN. Назначением шлюза является преобразование пакетов, поступающих из внешней открытой сети во внутреннюю сеть и обратно. При этом шлюз сети VPN выполняет комплекс операций кодирования и декодирования данных и необходимые процедуры обмена технологическими данными IPsec. Достаточно высокая сложность используемых при этом алгоритмов предъявляет повышенные требования к производительности шлюза сети VPN, что делает целесообразной его аппаратную реализацию. В современных сетях могут

быть использованы три варианта реализации шлюзов виртуальных частных сетей на основе технологий IPsec:

- специальное устройство;
- функциональный модуль маршрутизатора;
- аппаратно-программный комплекс.

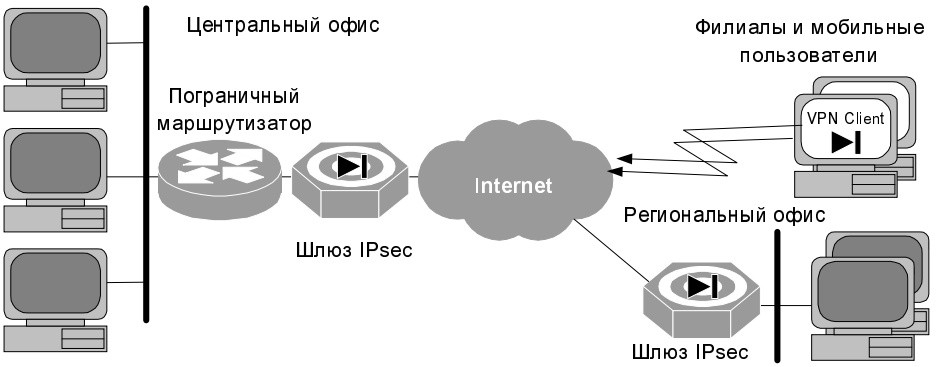


Рис. 8.13. Компоненты виртуальных частных сетей на основе технологий IPsec

Использование специального устройства для выполнения функций шлюза сети VPN позволяет наиболее полно реализовать требования к его эксплуатационным характеристикам и обеспечить практически предельный уровень защиты информации. Однако область возможного применения подобных устройств ограничивается их относительно высокой стоимостью и сложностью администрирования.

Достаточно популярными в настоящее время являются программные реализации алгоритмов IPsec-шлюза сети VPN. Создаваемые при этом аппаратно-программные комплексы позволяют строить относительно дешевые и гибкие виртуальные частные сети, отвечающие самым современным требованиям. Для обеспечения доступа мобильных клиентов и подключения удаленных филиалов по коммутируемым линиям в данном случае применяется специальное программное обеспечение VPN Client (Клиент частной сети), устанавливаемое на компьютеры пользователей. К недостаткам подобного рода сетей можно отнести их относительно небольшую пропускную способность и недостаточную надежность, определяемые, впрочем, соответствующими характеристиками используемого системного программного обеспечения.

Примечание

Для повышения пропускной способности часть наиболее сложных алгоритмов (обычно это алгоритмы шифрования и декодирования) может быть аппаратно

реализована специальными модулями расширения, устанавливаемыми в персональный компьютер или сервер.

Наиболее целесообразной представляется реализация виртуальных частных сетей IPsec с использованием модульных маршрутизаторов. При этом часть функций по созданию и обслуживанию виртуальной сети выполняется программным обеспечением маршрутизатора, а наиболее сложные из алгоритмов аппаратно реализуются в модулях расширения. Обычно такие решения обеспечивают высокий уровень производительности и надежности. При этом также осуществляется необходимая гибкость и масштабируемость и может быть достигнуто оптимальное значение соотношения цена/качество.

Виртуальные сети на основе технологии MPLS

Следует сразу же отметить, что технология MPLS не должна рассматриваться в качестве альтернативы технологии IPsec для построения виртуальных частных сетей. Дело в том, что перед разработчиками этих технологий в свое время были поставлены не совсем одинаковые задачи. При разработке концепции виртуальных сетей IPsec основное внимание уделялось обеспечению защиты информации при передаче ее по открытым сетям. Применение технологии MPLS позволяет связать в единые виртуальные сети фрагменты сетей многочисленных пользователей. Задачи именно такого типа стоят сегодня перед крупными компаниями — провайдерами информационных услуг, имеющими точки своего присутствия в различных удаленных регионах. Помимо задачи обеспечения информационной безопасности, в данном случае должны быть решены еще как минимум две важные задачи:

- обеспечение заданного уровня качества обслуживания;
- обеспечение передачи маршрутной информации.

При построении своих внутренних сетей большинство современных компаний используют корпоративные адреса. Таким образом, необходимая уникальность сетевого адреса обеспечивается только в пределах одной корпоративной сети. Понятно, что в данном случае обычные способы определения маршрута просто не могут быть использованы.

Современные информационные технологии обеспечивают возможность формирования разнородного пользовательского трафика и интеграции его в едином информационном потоке. Совершенно очевидно, что при передаче этого интегрированного трафика через разветвленную провайдерскую сеть должна соблюдаться установленная во внутренней сети пользователя дисциплина обеспечения качества обслуживания.

Все перечисленные выше сложные и взаимно противоречивые задачи могут быть решены при построении виртуальных частных сетей на основе технологии MPLS. Функциональная схема виртуальной частной сети, построенной на основе этой технологии, приведена на рис. 8.14.

Основным компонентом создаваемой в данном случае системы является разветвленная сеть провайдера информационных услуг.

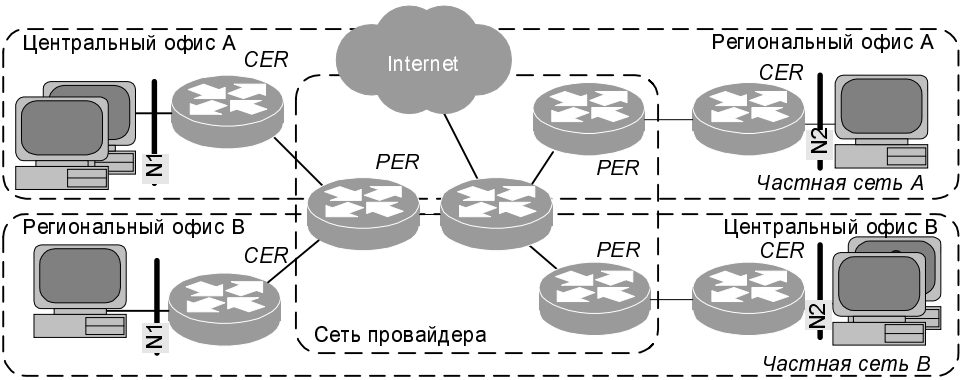
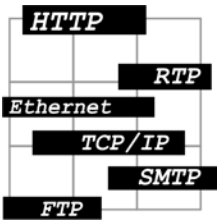


Рис. 8.14. Функциональная схема виртуальной частной сети, построенной на основе технологии MPLS

Сеть провайдера образуют маршрутизаторы, выполняющие обработку пакетов с использованием технологии MPLS. Пограничные маршрутизаторы сети провайдера (Provider Edge) PE помещают в пакеты, полученные от пограничных маршрутизаторов пользователя (Customer Edge) CE, метки, которые позволяют этим пакетам перемещаться по сети провайдера в нужном направлении. Другие пограничные маршрутизаторы извлекают эту метку из пакета и передают его пограничному маршрутизатору пользователя, находящемуся уже в другом регионе или городе. Поскольку перемещение пакета в сети, построенной по технологии MPLS, производится на основании значения его метки, а не содержимого заголовка IP-дейтаграммы, маршрутизация различных пользовательских сетей выполняется независимо, а их адресные пространства могут перекрываться. Информация, размещенная в метке, позволяет маршрутизаторам сети провайдера обрабатывать эти пакеты в соответствии с установленной пользователем дисциплиной обеспечения качества обслуживания. При этом, естественно, обеспечивается надежная информационная связь сегментов сетей пользователя. Информационная безопасность виртуальной частной сети, построенной на основе технологии MPLS, осуществляется по тому же принципу, что и безопасность сетей, функционирующих на канальном уровне. Для обеспечения большего уровня информационной безопасности при необходимости могут быть использованы дополнительные алгоритмы шифрования.

Глава 9



Система доменов и распределенная база данных DNS

Как известно, для обращения к хостам в сети Интернет используются 32-разрядные IP-адреса, однозначно идентифицирующие любой сетевой компьютер. Однако для пользователей применение IP-адресов при обращении к хостам не очень удобно.

Примечание

Здесь и далее термин "хост" используется для обозначения оконечных узлов сети.

Поэтому в сети Интернет принято присваивать имена всем компьютерам. Использование имен дает пользователю возможность лучше ориентироваться в киберпространстве Интернета — куда проще запомнить, например, имя **www.toyota.com**, чем четырехзвенную цепочку IP-адреса. Применение в Интернете понятных для пользователей имен и породило проблему их преобразования в IP-адреса.

Такое преобразование необходимо, так как на сетевом уровне адресация пакетов осуществляется не по именам, а по IP-адресам, следовательно, для непосредственной адресации сообщений в Интернет имена не годятся. Поэтому была создана новая система преобразования имен, позволяющая пользователю в случае отсутствия у него информации о соответствии имен и IP-адресов получить необходимые сведения.

Эта система получила название системы имен доменов — DNS (Domain Name System), для реализации которой был создан специальный сетевой протокол DNS. Кроме того, в сети создавались специальные выделенные информационно-поисковые серверы DNS-серверы.

Система DNS представляет собой множество функций и типов данных, образующих механизм работы с распределенной базой данных имен и адресов хостов Интернет.

Пусть, например, хост компании Horns and Hoofs Ltd. имеет адрес 199.159.206.10. Этот адрес ничего не говорит ни о месте расположения компании, ни о сфере ее деятельности. Кроме того, IP-адрес достаточно сложно

запомнить, он может поменяться, например, при перемещении хоста на другую ветку сети внутри организации.

Допустим, что этому серверу присвоено имя **www.obender.com**. Так же, как и IP-адрес, имя сервера разделено точками для удобства построения иерархии в домене на основе имен. По правилам построения имени иерархия задается справа налево.

В адресе **www.obender.com** домен верхнего уровня **com**. Домен **obender** — это средний уровень, и он является частью домена **com**. В рассмотренном примере **www** — это одно из имен компьютеров, расположенных в домене **obender**. Ясно, что запомнить и использовать имя хоста вам будет уже гораздо легче.

По имени домена верхнего уровня вы можете предположить местоположение или профиль организации. Следующие шесть доменов были определены для использования в США:

- **gov** — правительственные организации;
- **mil** — военные организации;
- **edu** — образовательные организации;
- **com** — коммерческие организации;
- **org** — общественные (свободные) организации;
- **net** — организации, предоставляющие сетевые услуги, как правило, региональные сетевые организации.

Кроме того, все страны мира имеют двухсимвольное имя, обозначающее домен верхнего уровня этой страны. Например, **uk** — для Великобритании; **ca** — для Канады; **au** — для Австралии; **fi** — для Финляндии и т. п. Процедура преобразования имен DNS позволяет использовать от двух до четырех уровней вложенных доменов. Окончательный выбор структуры имени должен быть сделан исходя из соображений удобства использования и простоты запоминания.

Примечание

Соединенным Штатам был также присвоен двухсимвольный код **us** для идентификации компьютерных доменов в США. Вышеупомянутые шесть доменов верхнего уровня, однако, до сих пор используются во всем мире для идентификации сетевых узлов США.

Итак, на основании имени интересующего вас хоста (**www.obender.com**), мы можем заключить, что хост находится в ведении коммерческой организации, зарегистрированной в США. Безусловно, наши догадки могут и не оправдаться, но, как правило, если какая-либо компания заинтересована в своей популярности на определенном рынке услуг, она приложит максимум усилий, чтобы даже имя ее Интернет-хоста работало как реклама.

Структура распределения компьютеров и сетей по доменам Интернет одновременно определяет и иерархические уровни администрирования этих систем.

Ответственность за уникальность машинных имен в каждом домене возлагается на администратора этого домена.

Например, у вас есть два компьютера с именем **kisa**. Вы не можете поместить эти два компьютера в один домен, поскольку тогда у них совпадут имена (оба будут **kisa.obender.com**). Однако можно определить субдомен **chairs** домена **obender.com** и поместить второй компьютер с названием **kisa** внутрь этого субдомена. Таким образом, можно увеличивать длину имени компьютера до тех пор, пока оно не станет уникальным.

История создания DNS

На заре создания сети Интернет таблицы соответствия между именами хостов и их IP-адресами были размещены в единственном файле (`HOSTS.TXT`) на компьютере в информационном центре NIC (Netword Information Center). Этот файл передавался по протоколу FTP всем хостам тогда еще совсем крохотной сети. Однако стремительный рост сети Интернет заставил выработать новую концепцию механизма разрешения адресов и имен.

- Использование старой схемы распространения таблицы соответствий потребовало бы увеличения пропускной способности канала пропорционально квадрату количества хостов сети. При современном росте количества участников сети и размеров таблицы это было бы нереальным.
- Небольшие организации, входящие в состав сети Интернет, хотели бы самостоятельно администрировать свои имена и адреса.
- Приложения, работающие с сетью Интернет, становятся более гибкими и сложными и требуют для работы создания своего сервиса разрешения имен.

Для решения проблемы построения новой системы управления соответствия имен и адресов была предложена концепция иерархического пространства имен. Положение имени в этой системе определяется характеристиками организации — владельца имени в общей схеме сети Интернет.

Решения DNS

В первую очередь проектировщики заботились о создании согласованного пространства имен, которое будет использоваться при работе с сетевыми ресурсами. Поэтому имена не должны содержать каких-либо частей от сетевых идентификаторов, IP-адресов, маршрутизаторов или другую подобную "служебную" информацию.

Администрирование распределенных баз данных было возложено на организации, которые подключаются к сети Интернет (этим занимается, как правило, системный администратор организации). Каждая из организаций

должна создать собственный сервер разрешения имен и фрагмент распределенной базы данных, содержащей информацию о домене хостов данной организации. Сервер обязан обслуживать хосты внутри организации и предоставлять доступ к базе данных этой организации извне.

В данном случае организация доступа является более важной задачей, чем обеспечение целостности и согласованности информации, т. е. изменение какой-либо записи базы данных может быть сделано до того, как это изменение будет перенесено в другие части базы данных. Если изменения других частей базы данных не могут быть сделаны, например из-за сбоя сети, используется старая информация. В этом случае система через определенные промежутки времени будет пытаться обновить данные. Временные интервалы обновления информации устанавливаются администратором системы.

Вся информация хостов домена собирается в главном архиве (master files), который представляет собой набор текстовых файлов определенного формата. Эти файлы корректируются системным администратором и через локальный "сервер имен" (name server) предоставляются пользователям. Пользователи получают доступ к базе данных через стандартные программы, называемые "программами разрешения" (resolves).

Стандартный формат файлов главного архива позволяет хостам обмениваться ими другими средствами, например, по протоколу FTP или по электронной почте. Это удобно для тех организаций, которые не используют "серверы имен". Такие организации могут создавать и модифицировать такие файлы, используя обычный текстовый редактор, передавать их на соседние хосты (где установлен "сервер имен") и таким образом пользоваться сервисом системы DNS.

Состав и основные компоненты DNS

Система DNS имеет три основных компонента.

- *Пространство имен домена (domain name space) и записи базы данных DNS (resource records).* Они определяют структуру "дерева" имен и данных, ассоциированных с этими именами. Каждая запись (или иначе, "лист дерева") пространства имен содержит информацию, ассоциированную с данным именем.
- *Серверы имен (name servers).* Сервер имен управляет всей информацией подчиненной ему области имен и данных. При обращении за информацией домена, который данный сервер не обслуживает, он должен переправить запрос серверу, обслуживающему этот домен, или серверу, стоящему на следующей ступени иерархии имен. Как правило, сервер имен имеет в подчинении (Authority) часть "дерева" данных и имен домена. Такая единица организации пространства имен называется *зоной* (zone). Зоны строятся на основе принадлежности какой-либо части данных к определенной сетевой структуре, например, домену или целой

организации. Зоны автоматически распределяются серверам имен, которые обеспечивают полный сервис содержащейся в них информации.

- *Программы разрешения имен (resolves)*. Эти программы возвращают информацию, хранящуюся в базе данных имен домена, по запросу пользователя. Программы разрешения должны иметь доступ по крайней мере к одному серверу имен и либо использовать полученную от него информацию для формирования ответа на запрос, либо переправить запрос на другой сервер имен. Как правило, программы разрешения реализуются в виде системного модуля, напрямую связанного с пользовательской программой, поэтому для их работы не требуется никакого дополнительного протокола обмена.

С точки зрения пользователя, к системе имен и данных домена можно получить доступ, вызвав простейшую процедуру или функцию операционной системы для работы с сервисом разрешения имен. При этом пространство имен представляется в виде единого дерева, к любой части которого пользователь имеет возможность обратиться.

С точки зрения программы разрешения имен, система имен и данных домена состоит из множества серверов имен. Каждый их этих серверов может обращаться за получением необходимой информации к одной или нескольким базам данных. Программа разрешения "видит" каждую из этих баз данных в статическом виде.

С точки зрения сервера разрешения имен, система имен и данных домена состоит из множества отдельных компонент, называемых зонами. Сервер имен имеет локальные копии некоторых из этих зон. Сервер имен должен периодически обновлять содержание зон из собственных файлов архива или других серверов имен. Кроме того, сервер имен должен немедленно обработать любой запрос от программы разрешения имен.

Примечание

В целях увеличения производительности, сервер имен и программы разрешения имен на одной машине могут частично выполнять функции друг друга и разделять доступ к базе данных.

Основным предназначением системы имен домена является обеспечение механизма именования ресурсов. Описанная выше функциональная структура системы DNS позволяет изолировать проблемы отдельных модулей системы, и, тем самым, создает универсальную модульную архитектуру.

Хост может участвовать в работе системы доменов различными способами, в зависимости от типа программного обеспечения и требуемой информации.

Пользователь взаимодействует с пространством имен через программы разрешения. Запросы пользователя представляют собой вызовы функций программы разрешения имен (или системных функций, если программа разрешения составляет часть операционной системы) с соответствующими параметрами. Программа разрешения отвечает на запросы пользователей на

основании имеющейся у нее в кэше информации или переадресовывает запросы на другие серверы имен. Программа разрешения имен может отправить несколько запросов на несколько серверов имен одновременно.

Простейшая схема взаимодействия с сервером имен представлена на рис. 9.1.

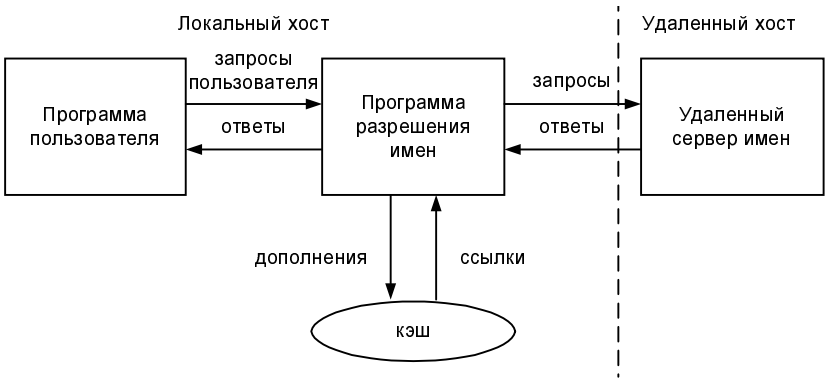


Рис. 9.1. Простейшая схема взаимодействия с сервером имен

Представленный на рис. 9.2 сервер имен обрабатывает запросы на основании информации из файлов главного архива локальной системы.

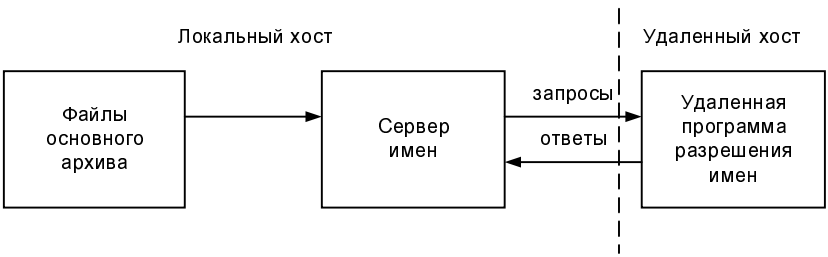


Рис. 9.2. Схема работы сервера имен с локальной базой имен

Система DNS требует, чтобы доступ к информации определенной зоны мог быть осуществлен с нескольких серверов имен. Серверы, имеющие доступ к данной зоне, должны синхронизировать информацию, используя механизм перемещения зон системы DNS.

На схеме, приведенной на рис. 9.3, сервер имен периодически устанавливает виртуальное соединение с другим сервером и синхронизирует информацию общих зон. Протокол синхронизации почти такой же, что используется при запросах пользователей.

Показанная на схеме (рис. 9.4) разделяемая база данных хранит данные пространства имен, которые состоят из записей, поступающих с операциями обновления и кэшированных данных запросов.

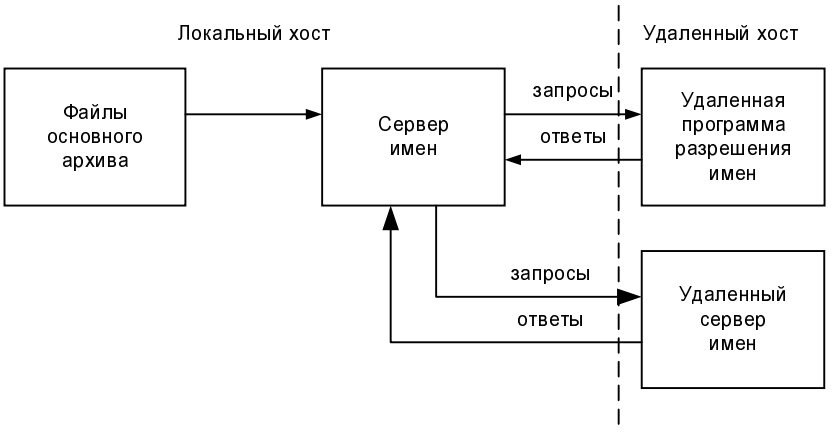


Рис. 9.3. Схема взаимодействия двух серверов имен

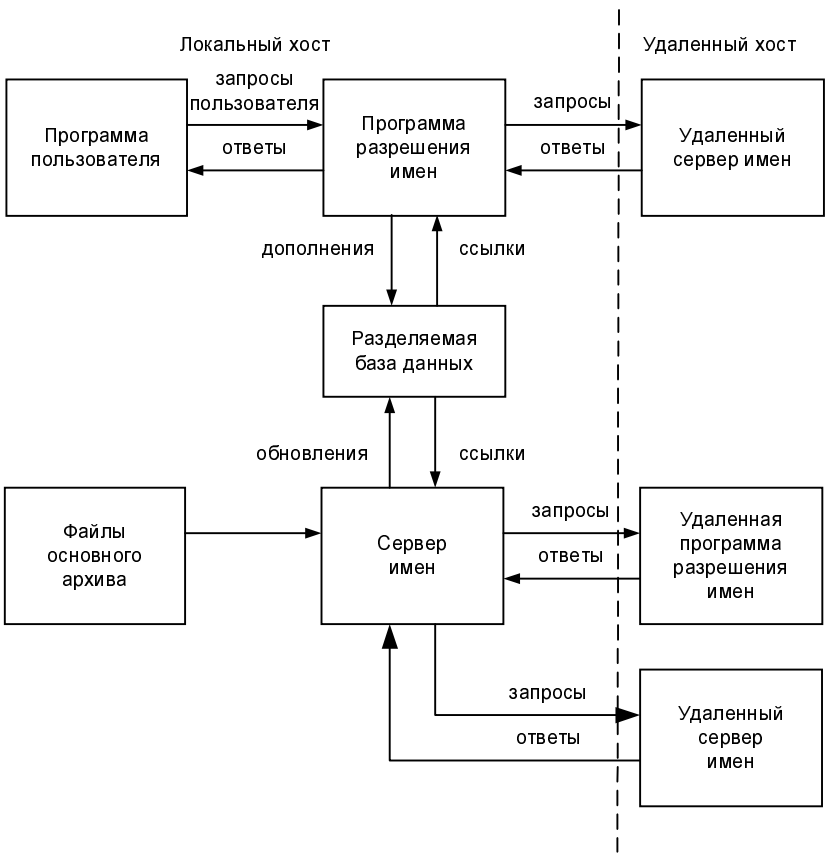


Рис. 9.4. Схема полной функциональности системы DNS

Пространство имен домена и записи базы данных

Спецификация

Структура имен домена представляет собой "дерево". Каждый узел сети (или лист этого "дерева") обозначает ресурс системы. Узел имеет метку, длина которой может достигать 63 байтов. Метка с нулевой длиной используется для обозначения корня дерева, т. е. данного домена. Пример древовидного пространства имен домена приведен на рис. 9.5.

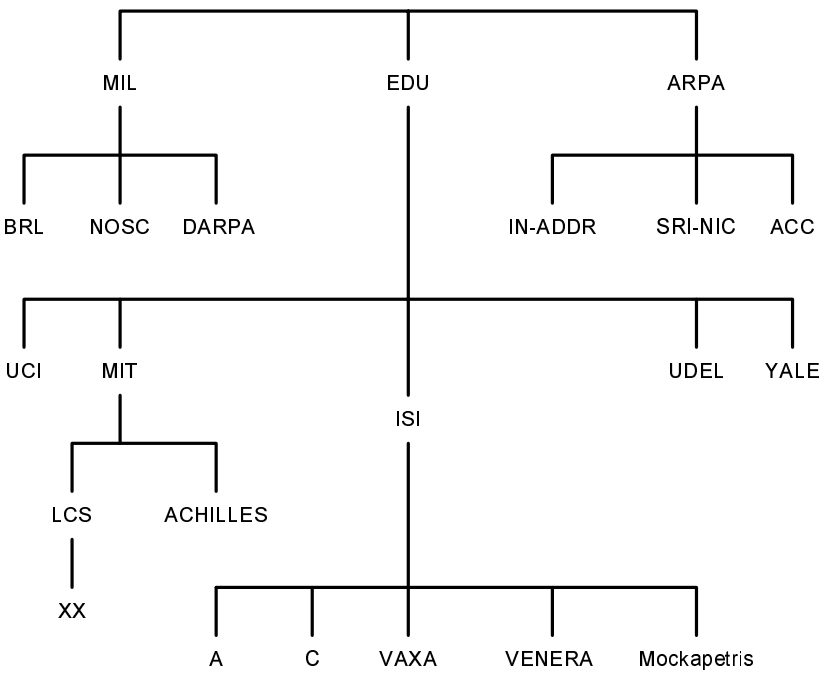


Рис. 9.5. Пример древовидного пространства имен домена

Полное имя домена представляет собой путь от данного узла к корню "дерева", составленный из меток доменов старших уровней, т. е. доменов, расположенных ближе к корню "дерева" в системе иерархии доменов. Имя домена представляет собой строку из ASCII-символов верхнего или нижнего регистра первой половины таблицы символов (до 128). Полное имя состоит из меток доменов (данного домена и всех родительских), отделенных друг от друга символом "точка".

Домен может представлять собой субдомен "родительских" доменов. Например, A.B.C.D — субдомен домена B.C.D, субдомен доменов C.D, D

и корневого домена. К данному домену можно обращаться через полное (абсолютное) имя, например, **silly.tamu.edu**, или через имя внутри домена-родителя, например, **silly**, когда мы находимся внутри домена **tamu.edu**. Длина полного имени домена, т. е. сумма длин всех меток пути ограничена 255 байтами.

Древовидная структура имен доменов подразумевает распараллеливание областей управления частями этой структуры между организациями. Потенциально каждый узел "дерева" может являться родителем неограниченного числа новых субдоменов. Однако на практике число субдоменов ограничивается администраторами серверов имен.

Техническая спецификация системы DNS не указывает, каким образом должно быть построено пространство имен для данной системы доменов — структура определяется сетевым администратором системы. Спецификация определяет только наиболее общие принципы построения DNS , которые используются для взаимодействия с программными приложениями, поскольку разные части дерева могут иметь различную семантику и использоваться приложениями для различных целей.

В примере, приведенном на рис. 9.5, корневой домен содержит три субдомена: **MIL**, **EDU** и **ARPA**. Домен **LCS.MIT.EDU** содержит один субдомен **XX.LCS.MIT.EDU**. Все "листья" "дерева" также являются доменами.

Записи базы данных

Имя домена определяет ресурс системы. Информация о ресурсе хранится в базе данных системы DNS виде отдельной записи — RR (Resource Record), компоненты которой представлены на рис. 9.6.

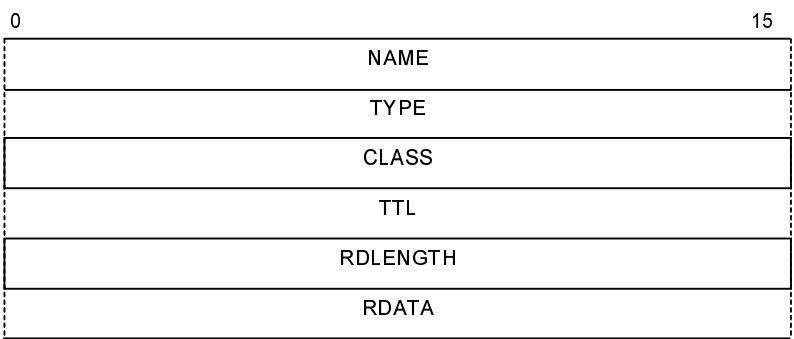


Рис. 9.6. Компоненты записи RR

Примечание

На этом и следующих рисунках: если поле ограничено штриховыми линиями, то это означает, что поле имеет переменную длину.

NAME (255 байтов). Имя, которое идентифицирует ресурс. Имя ресурса содержит имя домена (или хоста пользователя), в котором расположен этот ресурс, т. е. "владельца" информации. Например, запись RR, которая содержит IP-адрес хоста, содержит и имя домена, в котором расположен этот адрес (серверы имен хранят информацию о ресурсе в древовидной структуре параллельно со структурой пространства имен в базе данных).

TYPE (2 байта). Тип ресурса. Тип обозначает группу принадлежности ресурса, например, адрес хоста или идентификатор почтового сервера.

CLASS (2 байта). Поле класса ресурса. Поле класса идентифицирует формат данных ресурса. Например, указывает, принадлежит ли ресурс IP-адреса хоста к ARPA Internet (класс "IN") или к Computer Science Network format (класс "CSNET"). Необходимо отметить, что для разных типов ресурса класс ресурса может иметь различное значение. Например, класс IN использует только 32-битные IP-адреса, а класс CSNET использует как 32-битные IP-адреса, так и адреса X.25 и номера телефонов.

TTL (32 бита). Параметр, используемый для управления записью RR. **TTL** задает временной интервал продолжительности нахождения данной записи в кэше системы (в секундах). При просмотре этой записи в кэше данный интервал уменьшается, и если он становится меньше или равным нулю — данная запись уничтожается. Если данное поле пустое, то в качестве **TTL** берется значение поля **MINIMUM**, задаваемое в описываемой в данном разделе записи RR типа SOA (Start of Authority).

RLENGTH (32 бита). Длина поля данных. Это поле позволяет серверу имен или программе разрешения имен домена определить границы данных, даже если они не в состоянии интерпретировать содержимое поля данных.

RDATA Поле данных ресурса. Максимальная длина поля — 65535 байтов. Формат представления данных определяется полями типа и класса.

В табл. 9.1 представлены наиболее распространенные типы записей системы DNS.

Таблица 9.1. *Типы записей системы DNS*

Обозначение типа	Значение поля TYPE	Описание
A	1	Адрес хоста или ресурса (вид адреса определяется полем класса)
NS	2	Сервер имен домена определяет имя хоста, который управляет пространством имен и адресов домена и, тем самым, устанавливает нижнюю границу зоны, открытой записью типа SOA (точнее, определяет первую запись вне данной зоны)

Таблица 9.1 *(окончание)*

Обозначение типа	Значение поля TYPE	Описание
MX	15	Идентификатор почтового сервера (определяет хост, который служит в данном домене в качестве обработчика всей поступающей почты)
CNAME	5	Псевдоним ресурса
SOA	6	Имя домена, определяющего начало зоны пространства имен данного домена (владельца записи). Кроме тех случаев, когда сервер имен передает полномочия самому себе, тип SOA определяет верхнюю границу области полномочий (в текстовом файле — верхнюю границу зоны). Кроме того, поле данных может содержать дополнительную информацию о зоне, используемую сервером имен (временные интервалы обновления, и т. д.). Записи типа SOA никогда не кэшируются и создаются при инсталляции сервера имен
PTR	12	Указатель другой части пространства имен домена
WKS	11	Описание сервисов хоста (содержит спецификацию работающих на данном компьютере сервисов)
HINFO	13	Идентификатор процессора (CPU) и операционной системы (OS) хоста (используется, например, для проверки соответствия систем, производительности и др.)

Кроме перечисленных выше, тип может принимать и другие значения. Часть из них устарела и заменилась на другие (например, типы MD и MF были заменены на тип MX), а часть используется очень редко и их описание выходит за рамки данной книги.

Класс означает принадлежность записи к определенной системе, тогда как тип определяет функциональность записи в этой системе.

Например, сеть Интернет определена значением поля класса = 1, и обозначением "IN" — Internet, а сеть CHAOS значением класса = 3, и обозначением "CH". Для типа "A" класса "IN" поле данных будет содержать 32-битный IP-адрес, а для того же типа класса "CH" (система CHAOS) — имя домена CHAOS, следующее за 16-битным адресом CHAOS.

Как уже было сказано, формат поля данных во многом зависит от типа записи. Ниже представлены форматы полей данных ресурса для класса "IN" некоторых типов RR.

- *Тип "CNAME".* Запись с ресурсом типа CNAME (рис. 9.7) применяется для указания псевдонима хоста, тип — строка.

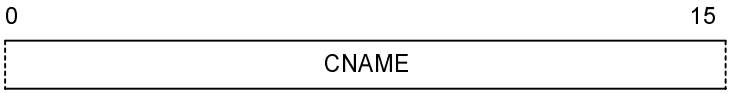


Рис. 9.7. Запись с ресурсом типа CNAME

- *Тип "HINFO"*. Запись с ресурсом типа HINFO (рис. 9.8) служит для хранения информации о хосте, например, об аппаратной платформе и операционной системе компьютера. CPU тип — строка, OS тип — строка.

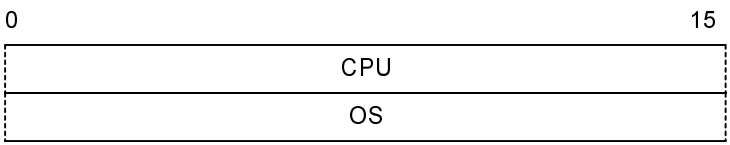


Рис. 9.8. Запись с ресурсом типа HINFO

- *Тип "MX"*. Для отдельных хостов или всего домена запись с ресурсом типа MX (рис. 9.9) позволяет определить почтовый шлюз, то есть компьютер, куда будет направляться электронная почта, предназначенная для этих хостов. Поле данных состоит из 16-битного значения приоритетности почтового сервера, задает приоритет доставки. При этом ноль означает самый высокий приоритет, и строки — имени хоста, выполняющего функции почтового сервера.

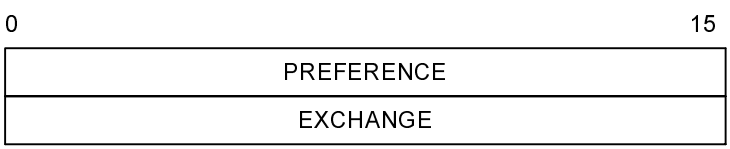


Рис. 9.9. Запись с ресурсом типа MX

- *Тип "NS"*. Запись с ресурсом типа NS (рис. 9.10) обозначает имя хоста, являющегося первичным сервером имен для домена. Поле данных содержит имя хоста.

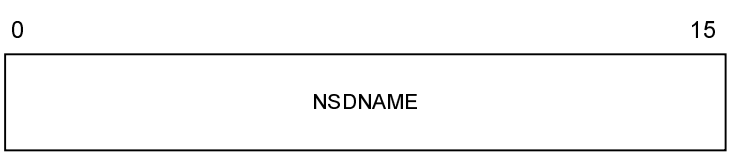


Рис. 9.10. Запись с ресурсом типа NS

- *Тип "SOA"*. Параметры зоны. Запись с ресурсом типа SOA (рис. 9.11) обозначает начало зоны управления сервера имен. Зона управления действует до следующей записи типа SOA. Пакет данных содержит: **MNAME** — имя родительского домена данной зоны; **RNAME** — имя

домена, обозначающее почтовый ящик администратора зоны; **SERIAL** — 32-битный идентификатор зоны; **REFRESH** — 32-битное значение интервала времени (в секундах), через который содержание зоны должно обновляться; **RETRY** — 32-битный интервал (в секундах) повторения запросов обновления содержания зоны в случае неудачи операции обновления; **EXPIRE** — 32-битный интервал (в секундах) продолжительности полномочий домена в зоне; **MINIMUM** — 32-битное значение (в секундах) минимального значения TTL, которое устанавливается в сообщениях ответа в данной зоне на запросы, если у запрашиваемой записи параметр TTL не установлен в большее значение.

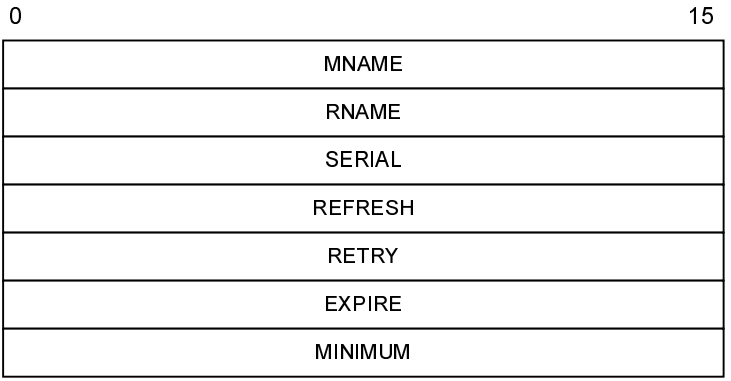


Рис. 9.11. Запись с ресурсом типа SOA

- *Тип "WKS".* Запись с ресурсом типа WKS (рис. 9.12) используется для описания широко известных сервисов, поддерживаемых определенным протоколом, хостом, расположенным по определенному адресу (поле **ADDRESS** — 32 бита). Поле **PROTOCOL** (8 бит) специфицирует тип протокола, а битовое поле переменной длины **BIT MAP** определяет порт сервиса, работающий по данному протоколу: каждый бит поля, равный 1, определяет активный порт (если бит, соответствующий номеру порта отсутствует, считается, что он равен 0). Например, если **PROTOCOL**=6 (TCP), а 26-й бит поля установлен в 1, это означает, что данный хост слушает 25-й (SMTP) порт протокола TCP. Запись типа WKS создается на каждый протокол, с которым работает хост.

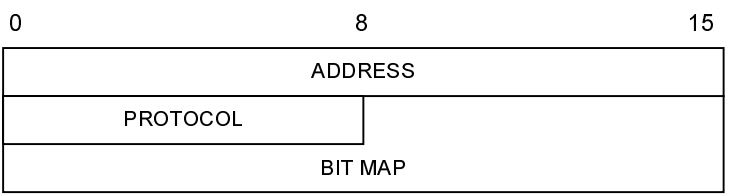


Рис. 9.12. Запись с ресурсом типа WKS

Текстовое представление данных

Записи RR (Resourse Record) хранятся в базе данных системы DNS и передаются в пакетах DNS-протокола в двоичном виде. Однако, как известно, записи RR модифицируются администратором в файлах главного архива в текстовом формате. Текстовый формат представления состояния базы данных значительно упрощает процедуры вставки, модификации или удаления записей.

Текстовый файл содержит последовательность записей, которые располагаются в строках, завершаемых нажатием клавиши <Enter>, выполняющей перевод строки (LF) и возврат каретки (CR). Для размещения информации на нескольких строках используются скобки.

В табл. 9.2 представлены значения специальных символов системы DNS.

Таблица 9.2. *Значения специальных символов системы DNS*

Символы	Значение
.	Отдельно стоящая точка обозначает текущий домен
@	Отдельно стоящий символ "@" в поле **NAME** обозначает текущий исходный домен
( )	Скобки используются для размещения поля **DATA** на нескольких строках
*	Заменяет любую последовательность символов
;	Комментарий. Вся информация, размещенная между этим символом и концом строки, игнорируется

Примечание

В записях ресурсов доменное имя, не заканчивающееся точкой, считается относительным. При обработке оно прибавляется к текущему домену. Поэтому, когда задается полное имя, его необходимо заканчивать точкой.

Общая структура файла выглядит следующим образом:

```
<domain-name><RR> [;<comment>]
<blank><RR> [;<comment>]
<blank>[;<comment>]
$INCLUDE <file-name> [<domain-name>] ;[<comment>]
```

Примечание

Квадратные скобки в структуре "[ ]" означают факультативность параметров, описание других ключевых слов выходит за рамки данной книги.

К элементам общей структуры относятся:

- `<blank>` — пустая строка, символы "пробела" или табуляции.
- `<domain-name>` — имя домена — владельца записи. Как правило, в текстовом файле запись (строка) RR начинается с идентификатора владельца данной записи. Если поле `domain-name` пустое, то в качестве него используется последнее заданное в предыдущих записях поле `domain-name`, т. е. предполагается, что данная запись относится к предыдущему имени домена (как правило, для удобства чтения добавляется несколько пробелов и делается выравнивание столбцов).
- `$INCLUDE` — вставляет имя файла имен в текущий файл имен (и может содержать имя домена, который описан в добавленном файле).
- `<RR>` — информационная запись файла имеет следующий формат:

  ```
  [<TTL>] [<class>] <type> <RDATA>
  [<class>] [<TTL>] <type> <RDATA>
  ```

  - запись может начинаться с поля `TTL` и поля класса. Как правило, эти параметры для всех записей одного файла (зоны) принимают одно и то же значение и определяются только один раз — в записи определения зоны, располагающейся в начале файла имен;
  - далее следуют поля типа записи и данных;
  - поле `TTL` записывается как целое число. Для того чтобы избежать неопределенности при синтаксическом разборе, мнемоники типа и класса различаются, `TTL` всегда представляет собой число, а мнемоника типа всегда последняя из этих трех полей;
  - завершает строку поле данных ресурса.

Ниже приведен пример текстового представления части зоны "дерева" домена ISI.EDU.

```
NS      A.ISI.EDU.
        NS      VAXA
        MX      20   VAXA
A       A       26.3.0.103
VAXA    A       10.2.0.27
        A       128.9.0.33


$INCLUDE <SUBSYS>ISI-MAILBOXES.TXT
```

где файл `<SUBSYS>ISI-MAILBOXES.TXT` может содержать, например, следующее:

```
MOE     MX      A.ISI.EDU.
LARRY   MX      A.ISI.EDU.
STOOGES MX      MOE
```

```
        MX      CURLEY
```

В данном примере не указан класс IN и поле TTL, которые подразумеваются одинаковыми. В следующем примере показаны два адреса домена **XX.LCS.MIT.EDU**, принадлежащие различным классам с полем TTL = 17777.

```
XX.LCS.MIT.EDU. 17777   IN   A    10.0.0.44
                17777   CH   A    MIT.EDU. 2420
```

Имена и псевдонимы

В больших системах сетевые ресурсы часто имеют несколько различных имен. Например, имена **C.ISI.EDU** и **USC-ISIC.ARPA** принадлежат одному и тому же хосту, а почтовые ящики **Mockapetris@C.ISI.EDU**, **Mockapetris@B.ISI.EDU** и **PVM@ISI.EDU** адресуют один и тот же объект.

Для определения нескольких имен одного объекта используется тип записи CNAME. Этот тип определяет псевдоним объекта, к которому он относится. Ясно, что псевдоним не может иметь какой-либо другой спецификации, поскольку данные для псевдонима объекта и самого объекта не могут различаться.

Присутствие типа CNAME предполагает выполнение сервером имен определенных дополнительных действий. Например, если сервер не смог найти какую-либо информацию по указанному имени, он проверяет записи типа CNAME данного класса. Если у указанного в запросе имени есть псевдоним — поиск повторяется по данному псевдониму (при повторном проходе записи типа CNAME уже не просматриваются).

Например, если сервер имен обрабатывает запрос адреса имени **FUN.MAN.ARPA**, а база данных содержит следующие записи:

```
FUN.MAN.ARPA   IN CNAME       JOLLY.EDU
JOLLY.EDU      IN A       10.0.0.12
```

обе эти записи возвращают в ответ на запрос адреса как по имени **FUN.MAN.ARPA**, так и по имени **JOLLY.EDU** значение 10.0.0.12.

Примечание

Имена доменов, указывающие на имя другого домена, должны использовать только реальное имя, а не псевдоним. Например, имя адреса домена **JOLLY.EDU**, описанного в примере, должно быть записано в следующем виде:

```
12.0.0.10.IN-ADDR.ARPA       IN     PTR    JOLLY.EDU
```

Запросы. Стандартные и инверсные запросы

Запросы представляют собой сообщения, которые отправляются серверу имен и на которые он должен ответить. Запросы передаются через дейта-

граммы протокола UDP (порт 53) или соединения TCP (порт 53). Выбор протокола осуществляется в зависимости от требований операции к скорости и достоверности передаваемых данных.

Например, протокол UDP используется для построения стандартных запросов сети Интернет, но не используется для операций перемещения зоны, поскольку в первом случае важна скорость выполнения операции, а во втором — надежность соединения для передачи данных. Кроме того, протокол UDP позволяет установить более гибкий контроль интервалов повторной передачи потерянных пакетов. Протокол TCP используется, например, когда сервер одновременно работает с несколькими соединениями или клиентская часть требует для передачи данных установки достоверного соединения.

Ответ на запрос либо содержит требуемую информацию, либо переадресует запрос на другой сервер имен, либо сообщает о возникновении ошибки обработки или передачи запроса.

Запросы передаются в виде сообщений специального формата. Сообщение состоит из заголовка, который содержит определенные управляющие поля, такие как поле кода операции (**OPCODE**), типа запроса, статуса запроса и др., и четырех секций для параметров записей RR, содержание которых зависит от кода операции запроса (рис. 9.13).

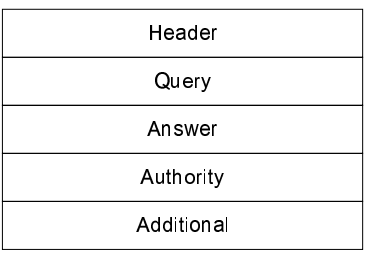


Рис. 9.13. Структура сообщения DNS-протокола

Header (Заголовок). Содержит управляющие поля.

Query (Запрос). Содержит имя и другие параметры запроса.

Answer (Ответ). Содержит RR ответа на запрос.

Authority (Права). Содержит RR, описывающие другие полномочные для работы с данной RR серверы имен.

Additional (Дополнительные секции). Содержат дополнительную информацию.

На рис. 9.14 показана структура заголовка сообщения DNS-протокола.

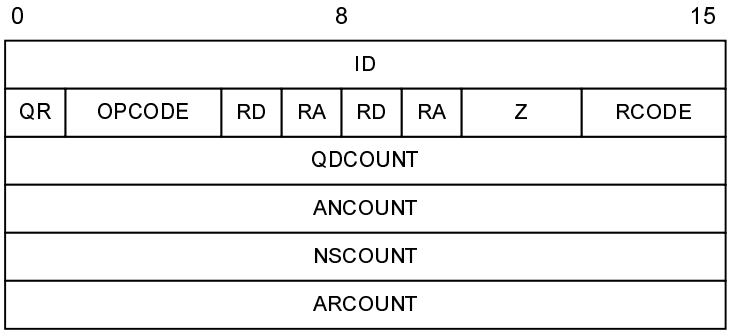


Рис. 9.14. Структура заголовка сообщения DNS-протокола

В заголовке сообщения DNS-протокола содержатся следующие поля и признаки:

- Поле **ID** (16 бит). Идентификатор цикла запрос-ответ. Идентификатор запроса копируется в поле сообщения ответа, которое, в свою очередь, может быть запросом следующей итерации первоначального запроса.
- Признак **QR** (1 бит). Флаг запроса или ответа (0/1).
- **OPCODE** (4 бита). Тип запроса (простой/инверсный/запрос статуса).
- **AA** (1 бит). Флаг уполномоченного ответа.
- Признак **TC** (1 бит). Флаг обрезания сообщения, которое выполняется в том, случае, если размер исходного сообщения превысил значение MTU канала передачи данных.
- Признак **RD** (1 бит). Флаг предпочтительного выполнения рекурсии при обработке запроса сервером имен *(см. разд. "Механизмы и алгоритмы обслуживания запросов" гл. 9)*.
- Признак **RA** (1 бит). Флаг возможности выполнения рекурсии при обработке запроса сервером имен.
- Поле **Z** (3 бита).Зарезервировано для дальнейшего использования.
- Поле **RCODE** (4 бита). Код статуса ответа.
- Поле **QDCOUNT** (16 бит). Количество записей в секции запроса.
- Поле **ANCOUNT** (16 бит). Количество записей в секции ответа.
- Поле **NSCOUNT** (16 бит). Количество записей в секции уполномоченных серверов NS.
- Поле **ARCOUNT** (16 бит). Количество записей в дополнительной секции.

Структура секции **Query** сообщения DNS-протокола представлена на рис. 9.15. Стандартный запрос (секция **Query**) содержит имя домена, информацию на

который мы хотим получить (**QNAME**), тип запроса (**QTYPE**), класс запроса (**QCLASS**). Поля **QTYPE** и **QCLASS** по 16 бит каждое, содержат коды типа и класса записи RR, информацию о которой мы хотим получить.

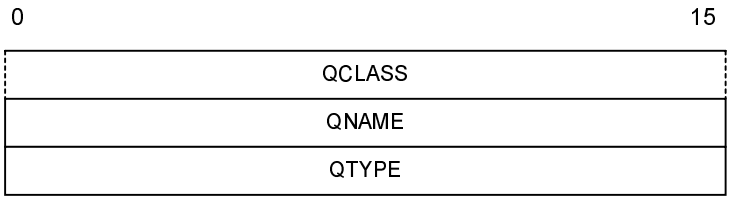


Рис. 9.15. Структура секции **Query** сообщения DNS-протокола

Используя параметры имени домена (хоста), типа и класса, сервер имен ищет в таблице базы данных соответствующие записи.

Например, хост-отправитель почты может сделать запрос о наличии в домене **ISI.EDU** почтовых серверов, например, для отправки почты в этот домен. Параметры запроса будут следующими: QNAME=ISI.EDU, QTYPE=MX, QCLASS=IN. В ответ хост получит пакет, содержащий в секции ответа (**Answer**) структуру записи RR (см. рис. 9.6), например, со следующей информацией:

```
ISI.EDU.   MX   10   MARS.ISI.EDU.
           MX   10   POMPA.ISI.EDU.
```

дополнительная секция ответа (**Additional**) может содержать IP-адреса узлов доменов:

```
 MARS.ISI.EDU.    A    10.2.0.27
                  A   128.9.0.33
 POMPA.ISI.EDU.   A    10.1.0.52
                  A   128.9.0.32
```

Кроме обычных запросов существуют так называемые инверсные запросы. Их отличие от стандартных состоит в том, что они определяют имя домена (или доменов) по характеристикам определенного ресурса. Например, если стандартный запрос определяет записи RR типа SOA, соответствующие имени домена, то соответствующий инверсный запрос определяет имя домена по параметрам этой записи.

Механизм инверсных запросов используется, например, для установления соответствия IP-адресов и имен хостов сети Интернет. Для соотнесения IP-адресов и имен хостов в сети Интернет используется специальный домен **IN-ADDR.ARPA**. Записи в домене **IN-ADDR.ARPA** состоят из суффикса **IN-ADDR.ARPA** и четырех предшествующих ему меток. Каждая метка представляет собой байт интернет-адреса. Расположены метки в порядке, обратном расположению байтов интернет-адреса. Например, если интернет-адрес домена 10.2.0.52, то в домене **IN-ADDR.ARPA** ему будет соответствовать

запись 52.0.2.10.IN-ADDR.ARPA. Такое обратное представление адреса позволяет определять управляющие зоны и шлюзы для классов сетей.

Например, если домен **IN-ADDR.ARPA** содержит информацию о ISI-шлюзе между сетью 10 и 26 и имеет адреса 10.2.0.22 и 26.0.0.103, то база данных будет содержать следующие записи:

```
10.IN-ADDR.ARPA.            PTR GW.ISI.EDU.
26.IN-ADDR.ARPA.            PTR GW.ISI.EDU.
22.0.2.10.IN-ADDR.ARPA.     PTR GW.ISI.EDU.
103.0.0.26.IN-ADDR.ARPA.    PTR GW.ISI.EDU.
```

При использовании сервиса инверсных запросов следует иметь в виду, что, поскольку домен **IN-ADDR.ARPA** и обычный домен данного хоста или шлюза расположены в различных зонах, возможно, что данные в этих доменах будут не согласованы, кроме того, шлюзы, как правило, имеют в разных доменах различные имена.

Предназначение механизма обратных запросов и домена **IN-ADDR.ARPA** состоит в том, чтобы по IP-адресу хоста можно было быстро определить домен и управляющие элементы домена, где расположен этот хост.

Все серверы имен должны понимать и распознавать как стандартные, так и инверсные запросы. Ясно, что инверсные запросы не могут гарантировать полноту и уникальность возвращаемой информации даже внутри определенного домена. Эти запросы, как правило, используются при инсталляции и тестировании различных компонентов сервера имен.

Серверы имен

Серверы имен управляют базами данных имен и адресов домена. База данных разделена на секции, называемые зонами, которые распределяются для обслуживания между различными серверами имен. Основной задачей серверов имен является обработка запросов на основе информации локальных зон, расположенных на данном хосте, или переадресация запроса на серверы, располагающие требуемой информацией.

Алгоритм сервера имен во многом зависит от сетевой среды работы, операционной системы хоста и от структуры данных, которыми сервер управляет. Основной единицей базы данных пространства имен является зона.

Понятие зоны. Деление DNS-пространства на зоны

Зона представляет собой полную спецификацию какой-либо части пространства имен домена. К данной зоне, как правило, имеют доступ несколько серверов имен. В соответствии с соглашением, каждая зона должна

быть доступна, по меньшей мере, с двух серверов. Один сервер имен обычно поддерживает работу одной или более зон.

База данных домена может быть разбита на зоны двумя способами: по классам и "срезам" пространства имен.

- *Деление по классам*. Для какого-либо класса строится и инсталлируется отдельная база данных. Поскольку пространство имен одно и то же для всех классов, структуры данных для отдельных классов можно рассматривать как параллельные "деревья" пространства имен. Ясно, что данные узлов этих "деревьев" будут зависеть от класса, которому они принадлежат. Причиной построения нового класса, как правило, является необходимость в новом формате данных или независимом (от уже существующего) пространстве имен.
- *Деление по "срезам"*. Внутри отдельного класса "срез" в пространстве имен можно сделать между любыми двумя соседними узлами. После построения такого "среза" каждая группа пространства имен будет представлять собой отдельную зону. Такой "срез" в пространстве имен может проходить в разных классах по различным местам пространства имен, части новых зон могут принадлежать различным серверам имен и т. д.

Эти правила деления подразумевают, что каждая зона имеет по крайней мере один узел и имя домена, пространством имен которого она владеет. Кроме того, в соответствии со структурой "дерева", каждая зона имеет своего "родителя" — зону, расположенную в системе иерархии ближе к корню "дерева". Имя зоны-родителя часто используется для определения имени данной зоны.

Безусловно, деление пространства имен таким образом, что каждый домен располагался бы в отдельной зоне или так, чтобы все узлы домена располагались бы в отдельной зоне, было удобно с точки зрения единого администрирования. Однако вместо этого база данных чаще всего разделена по организационному принципу, т. е. каждая организация, работающая с определенной частью "дерева", желает контролировать и изменять свою часть самостоятельно.

Информация, хранящаяся в определенной зоне, состоит из четырех основных частей:

1. Данные о правах узлов внутри зоны.
2. Данные, определяющие узел родителя зоны.
3. Данные о подзонах данной зоны.
4. Данные о правах доступа к серверам имен подзон данной зоны.

Вся эта информация хранится в виде RR-записей, т. е. данная зона может быть полностью описана в терминах записи RR. Зоны могут очень просто перемещаться между серверами имен, например, через механизм перемеще-

ния соответствующих записей RR, посредством обмена RR-сообщениями или передачей по FTP текстовых файлов главного архива.

Информация, находящаяся под управлением данной зоны, состоит из всех записей RR, прикрепленных ко всем узлам от вершины узла данной зоны к узлам-потомкам зоны.

Для управления зоной особенно важными являются записи RR, описывающие вершину зоны. Эти записи могут быть двух типов: записи серверов имен — по одному в записи RR и одна запись RR типа SOA, описывающая параметры работы зоны. Записи типа NS обозначают срезы данной зоны. Поскольку серверы имен всегда ассоциированы с границами зоны, записи RR типа NS могут располагаться только в узлах, которые являются родительскими каких-либо зон. Среди записей, образующих зону, записи RR типа NS могут располагаться как в вершине зоны, так и в ее конце, но никак не посередине.

Примечание

Структура зон содержит всю необходимую информацию для установления соединения с серверами имен своих подзон. Однако для обращения к серверам имен подзон необходимо знать IP-адреса этих серверов. В частности, если имя сервера имен само хранится в подзоне, мы можем столкнуться с ситуацией, когда запись RR типа NS говорит, что, для того чтобы получить адрес сервера, нам необходимо обратиться к серверу, адрес которого мы хотим выяснить. Для разрешения подобных недоразумений зона содержит так называемые "приклеенные" записи, которые не являются частью зоны и содержат адреса серверов подзоны.

Механизмы и алгоритмы обслуживания запросов

Одним из назначений сервера имен является обработка запросов. Существуют два способа обработки запросов сервером.

- *Нерекурсивный способ.* Для ответа сервер использует только локальную информацию. Ответное сообщение содержит либо требуемую информацию, либо ошибку, либо ссылку на соседний сервер. Данный способ должен поддерживаться всеми серверами имен.
- *Рекурсивный способ.* В этом состоянии сервер имен работает также в роли программы разрешения (resolve) и в ответном сообщении может возвратить либо ошибку, либо требуемую информацию, но никогда — ссылку на другой сервер имен. Данный способ не обязан поддерживаться серверами имен. Для работы с этим способом необходимо специальное программное обеспечение как на хосте сервера имен, так и на машине клиента. Этот способ используется в следующих случаях:
  - Обработчик ответов не может обрабатывать ответы, содержащие ссылки на другие серверы.
  - Запрос должен быть обработан только определенными серверами.

Алгоритм обработки запросов, используемый сервером имен, зависит от операционной системы хоста и структуры данных, используемой для хранения записей RR. Далее приведен алгоритм (рис. 9.16), предполагающий, что записи RR организованы в виде структуры дерева, по одному "дереву" на каждую зону.

1. Установить или снять флаг использования рекурсии (зависит от настроек сервера). Если флаг рекурсивной обработки установлен, переходим на шаг 5, иначе — на шаг 2.
2. Ищем зону ближайшего родителя QNAME (Query Name — имя требуемого ресурса). Если нашли, то переходим на шаг 3, иначе — на шаг 4.
3. Просматриваем (сверху вниз) метки зоны. Просмотр останавливается, если происходит одно из следующих событий:
   - Найдена запись QNAME. Если поиск остановился на записи типа CNAME (псевдонима), но QTYPE (Query Type — тип требуемого ресурса) не совпадает с типом CNAME (нам не нужен псевдоним), тогда копируем запись типа CNAME в сообщение ответа, изменяем QNAME на значение псевдонима CNAME и возвращаемся на шаг 1. Иначе, копируем все записи RR, соответствующие QTYPE в сообщение ответа и переходим на шаг 6.
   - Поиск вышел за пределы зоны — мы получаем ссылку. Это случается, когда мы наталкиваемся на запись RR типа NS, обозначающую срез или конец данной зоны. Копируем запись RR типа NS в секции полномочий (**Authority**) сообщения ответа, помещаем в дополнительные (**Additional**) секции сообщения адреса соответствующих серверов имен (используя "приклеенные" записи) и переходим на шаг 4.
   - Метка не соответствует записи (метка не существует), проверяем присутствие символа "*". Если символ "*" не обнаружен, проверяем, является ли искомое имя QNAME настоящим или псевдонимом CNAME. Если имя настоящее — возвращаем ошибку поиска. Выходим. Если символ "*" обнаружен, проверяем поле QTYPE. Копируем записи RR в соответствующее поле QTYPE в сообщение ответа. Переходим на шаг 6.
4. Просматриваем страницы кэша. Если QNAME найдена, копируем все записи RR с соответствующими QTYPE в сообщение ответа. Переходим на шаг 6.
5. Для ответа на запрос используем локальную программу разрешения имен (или ее копию). Помещаем в сообщении ответа все результаты поиска, включая промежуточные записи типа CNAME.
6. Используя только локальные данные, добавляем некоторые часто используемые записи RR (например, адреса соответствующих доменов) в дополнительные секции сообщения. Выход (окончание алгоритма).

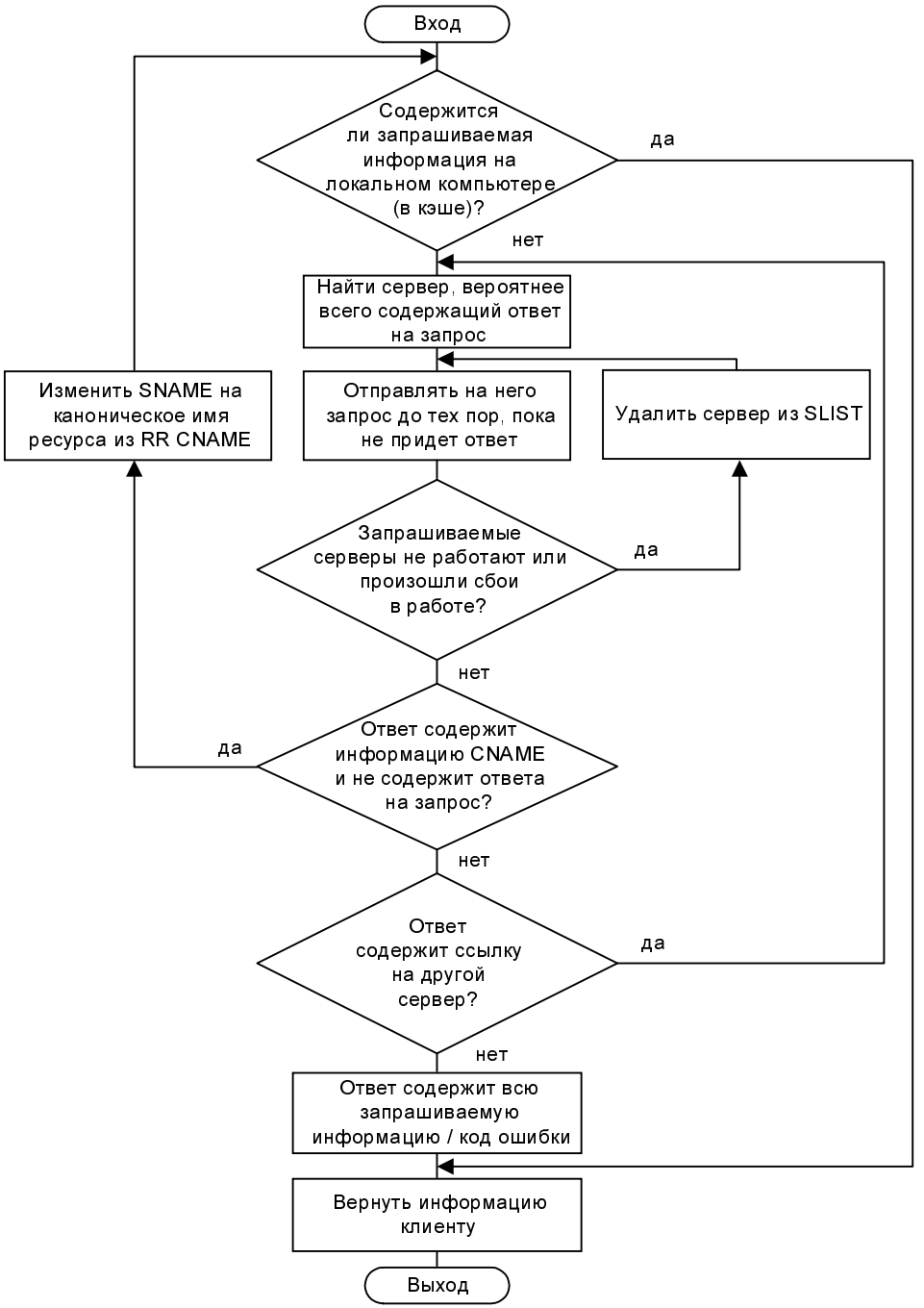


Рис. 9.16. Алгоритм обработки запросов сервером имен

Инсталляция и перемещение зон

Одной из обязанностей администратора зон является инсталляция зон на всех серверах имен, которые уполномочены для работы с этими зонами. При изменениях параметров зоны на одном из серверов эти изменения должны быть тиражированы на все серверы имен. Тиражирование или перемещение зон может быть сделано как по протоколу FTP, используя механизм передачи текстовых файлов архива, так и через протокол DNS.

Модель автоматического перемещения или обновления зон построена на том, что один из серверов имен является основным (master) в данной зоне. Все изменения на других зависимых (slave) серверах координируются в соответствии с изменениями главных файлов архива основного сервера.

После редактирования файлов главного архива администратор дает указание серверу загрузить новую зону (или новые параметры зоны). Серверы имен, которые координируют свои данные по данным основного сервера имен, периодически (через определенные промежутки времени) проверяют возможные изменения данных основного сервера и копируют их к себе (в том числе и копию новой зоны).

Для обнаружения изменений зависимые серверы проверяют поле **SERIAL** записи типа SOA зоны основного сервера. Какое бы изменение не было сделано, поле **SERIAL** записи типа SOA зоны изменится. Оно может или просто увеличиться, или измениться в соответствии с датой и временем изменения файла главного архива (master file). Поле **SERIAL** служит идентификатором версии настроек зоны, и по его значению можно определить, какая из двух копий зоны более поздняя.

Механизмом периодических запросов зависимых серверов управляют несколько параметров записи RR типа SOA — **REFRESH**, **RETRY**, **EXPIRE**.

Сразу после загрузки новой зоны зависимый сервер ждет интервал времени в секундах, количество которых указано в поле **REFRESH**, затем делает проверочный запрос на совпадение серийных номеров зоны данного и основного серверов имен. Если запрос не проходит (возвращается ошибка и др.), запрос будет повторяться через количество секунд, содержащееся в поле **RETRY**. Ответом на проверочный запрос является запись RR типа SOA основного сервера зоны. Если значения полей серийного номера зоны основного и зависимого сервера совпадают, то следующий запрос будет опять сделан через количество секунд, указанное в поле **REFRESH**. Если зависимый сервер обнаружит, что превышено значение **EXPIRY** (время существования копии зоны), копия зоны должна быть помечена как устаревшая и уничтожена.

Если ответ на проверочный запрос показал, что зона была изменена, зависимый сервер обязан сделать запрос на перемещение зоны целиком (AXFR).

Ответом на этот запрос будет последовательность сообщений. Первое и последнее сообщения должны содержать данные вершины узла зоны. Между ними располагаются сообщения с RR-зоны. Зависимый сервер должен обработать этот поток сообщений и построить собственную копию данной зоны.

Каждый зависимый сервер выполняет операцию синхронизации зон относительно основного сервера, но он также может выполнить эти операции относительно других зависимых серверов. Это может быть использовано, например, при сбоях работы сети и невозможности связаться с основным сервером или сбоях в работе основного сервера.

Программы разрешения имен

Программы разрешения имен (resolve) представляют собой интерфейс для работы пользовательских программ с серверами имен домена.

В простейшем случае программа разрешения получает запрос от пользовательской программы (например, почтового агента, TELNET или FTP) в виде системного вызова или вызова из подпрограммы. Требуемая информация возвращается в формате, понятном пользовательской программе и совместимом с форматом данных локального хоста.

Программы разрешения расположены на той же самой машине, что и программы, запрашивающие этот сервис. Но для работы программ разрешения необходимо обращаться к серверам имен на других хостах, и время ответа этих программ может варьироваться от миллисекунд до нескольких секунд. Поэтому одной из важных особенностей программ разрешения является возможность устранения сетевых задержек ответов, используя механизм кэширования предыдущих результатов запросов. Этот механизм позволяет разделять доступ и совместно использовать кэш данной программы разрешения имен различными процессами, пользователями и хостами сети.

Интерфейс разрешения имен

Интерфейс клиентской программы с программой разрешения имен в значительной степени зависит от программного обеспечения локального хоста, но этот интерфейс обязан содержать три стандартные части, к которым относятся:

1. *Трансляция имен хостов в адреса хостов.* Эта функция построена на основе функций, работающих с файлом HOSTS.TXT, т. е. в определенной строке этого файла указано имя домена и соответствующий ему IP-адрес. При работе с протоколом DNS эта функция строит запрос по имени домена записи RR типа A.
2. *Трансляция адресов хостов в имена хостов.* Эта функция при работе с файлом HOSTS.TXT использует тот же механизм: в определенной строке этого файла определен IP-адрес и соответствующее ему имя домена. При построении DNS-запроса по адресу используется суффикс

"IN-ADDR.ARPA" типа PTR. Например, запрос на имя хоста с адресом 1.2.3.4 выглядит как запрос записи RR типа PTR для имени "4.3.2.1.IN-ADDR.ARPA".

3. *Функция просмотра.* Эта функция позволяет строить произвольные запросы базы данных имен и адресов и не поддерживается в ранних версиях системы. Запрос может быть построен на основе параметров QNAME, QTYPE или QCLASS. Ответ содержит все записи, удовлетворяющие условиям запроса.

Эти функции возвращают или требуемые данные в определенном формате (записи RR базы данных), или код ошибки (например, имя не зарегистрировано в базе данных), или сообщение, что информация не найдена (например, имя домена существует, но нет данных запрашиваемого типа).

Функциональность

Рассмотрим алгоритм работы программы разрешения имен (рис. 9.17) со следующими исходными параметрами:

- SNAME — имя искомого домена;
- STYPE —тип запроса QTYPE
- SCLASS —класс запроса QCLASS
- SLIST — структура, описывающая серверы имен и зону, которые в данный момент просматривает программа разрешения. Эта структура содержит информацию о серверах имен, которые, предположительно, могут содержать искомые данные.
- SBELT — структура, схожая с SLIST. Она инициализируется из файла конфигурации и содержит список серверов, которые должны использоваться, когда программа разрешения имен не имеет никакой информации для выбора сервера имен.
- CACHE — эта структура хранит результаты предыдущих запросов.

Верхний уровень алгоритма запросов содержит четыре шага:

1. Просмотреть информацию на локальном компьютере (как правило, это содержание кэша) и, если информация найдена, вернуть ответ клиенту.
2. Найти сервер, вероятнее всего содержащий ответ на запрос.
3. Отправлять на него запрос до тех пор, пока не придет первый ответ.
4. Проанализировать ответ:
   - если ответ содержит всю запрашиваемую информацию или содержит имя ошибки, поместить эти данные в кэш и вернуть клиенту;

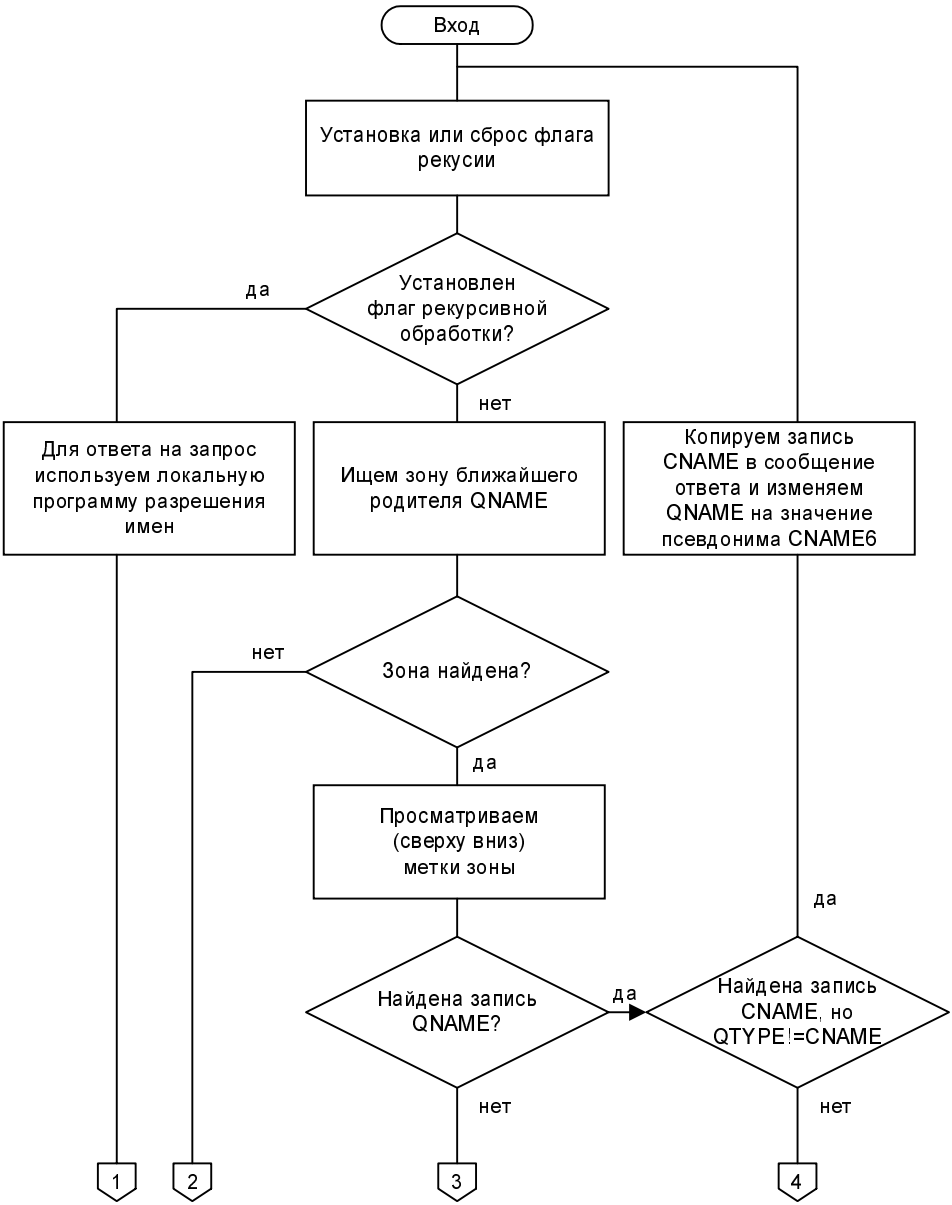


Рис. 9.17. Алгоритм работы программы разрешения имен, если ответ содержит ссылку на другой сервер, поместить эту информацию в кэш и перейти к шагу 2 *(начало)*

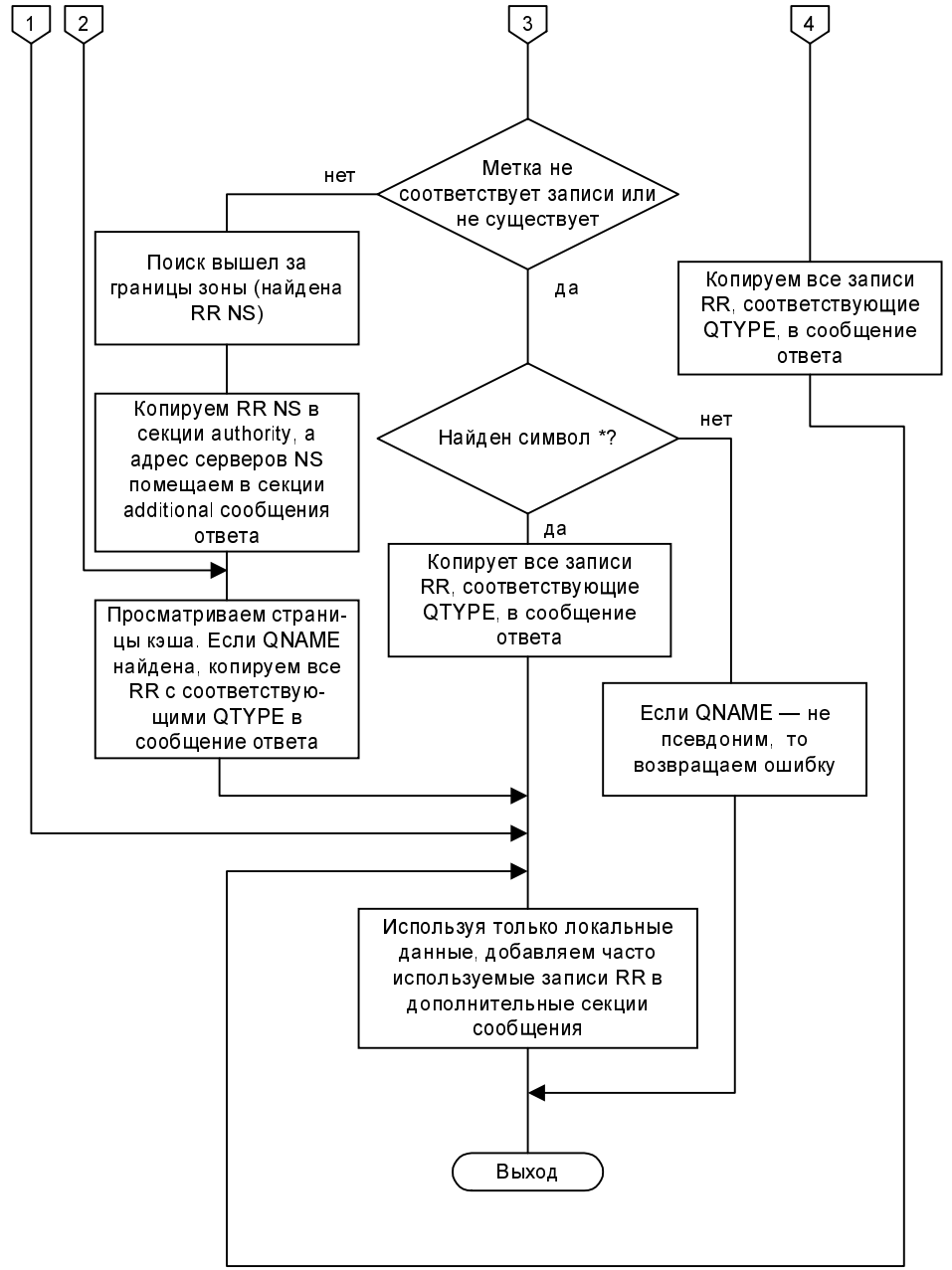


Рис. 9.17. Алгоритм работы программы разрешения имен, если ответ содержит ссылку на другой сервер, поместить эту информацию в кэш и перейти к шагу 2 *(окончание)*

- если ответ содержит информацию типа CNAME и не содержит ответа на запрос, поместить информацию типа CNAME в кэш, изменить SNAME на каноническое имя ресурса из записи RR типа CNAME и перейти к шагу 1;
- если ответ содержит информацию о том, что запрашиваемые серверы не работают или произошли еще какие-либо сбои в работе, удалить сервер из SLIST и перейти к шагу 3.

Рассмотрим эти шаги алгоритма более подробно.

- На шаге 1 поиск данных производится в кэше компьютера. Если данные находятся в кэше, то считается, что при нормальной работе этого достаточно для формирования ответа. Некоторые программы разрешения можно настроить так, чтобы они не позволяли пользовательским программам использовать кэшированные данные, однако в условиях нормальной работы этого делать не рекомендуется.
- На шаге 2 программа разрешения ищет сервер имен, которому можно было бы отправить запрос. Стратегия поиска выглядит так: сначала просматриваются записи RR локального сервера, начиная с SNAME, затем просматриваются серверы-родители SNAME, затем родители этих родителей и т. д. до корня. Например, если SNAME = Mockapetris.ISI.EDU, сначала мы ищем записи сервера Mockapetris.ISI.EDU типа NS, затем ISI.EDU, затем EDU, и наконец "." (корень). Записи RR типа NS содержат имена хостов зоны, расположенной рядом или выше SNAME. Копируем этот список в SLIST и, используя локальные данные, определяем адреса этих хостов.

  В некоторых случаях информация адреса недоступна, тогда программа может запустить параллельный процесс программы разрешения имен, который будет пытаться определить адреса по уже известным данным и по мере получения требуемой информации передавать ее программе-родителю. Если поиск записей RR NS завершился неудачно, программа разрешения имен для формирования структуры SLIST использует содержание вспомогательной структуры SBELT, т. е. если программа разрешения не знает, к какому серверу имен ей обращаться, она должна использовать информацию из файла настройки, в котором содержится список "пожарных" серверов. Хотя подобная ситуация относится к исключительным ситуациям, из файла конфигурации, как правило, выбираются какие-либо два сервера из корневых серверов и два сервера из серверов домена. Такой алгоритм построен для увеличения производительности работы системы, поскольку корневые серверы предоставляют

быстрый доступ ко всему пространству имен, а локальные серверы имеют более быстрый доступ к сети данного домена.

- На шаге 3 запросы будут отправляться до тех пор, пока не будет получен какой-либо ответ (организуется периодический опрос по всем адресам серверов).
- На шаге 4 производится анализ ответов (синтаксический разбор поступивших данных). Программа должна следить за тем, чтобы ответы содержали метку (идентификатор) сообщения запроса.

  Как правило, на запрос от уполномоченного сервера приходит один ответ, содержащий либо сообщение об ошибке, либо запрашиваемые данные. Данные передаются пользовательской программе и, если поле **TTL** больше нуля, помещаются в кэш для использования в дальнейшем.

  Если ответ содержит сообщение о перенаправлении запроса, программа разрешения должна проверить, что рекомендуемый для обращения сервер "ближе", чем серверы из списка SLIST. Если нет, ответ считается недействительным и отбрасывается. В том случае, если рекомендуемый сервер "ближе", запись RR типа NS сервера и его адреса кэшируются, записываются в SLIST и процесс поиска возобновляется.

  Если ответ содержит CNAME, а тип запроса не совпадает с CNAME, процесс поиска возобновляется с параметрами ответа (каноническим именем) CNAME.

Пример построения и инсталляции DNS

Рассмотрим пример построения зоны DNS.

На рис. 9.18 уполномоченный сервер каждого домена показан рядом со своим доменом в круглых скобках. Корневые серверы: **C.ISI.EDU**, **SRI-NIC.ARPA** и **A.ISI.EDU**. Домен **MIL** обслуживают серверы **SRI-NIC.ARPA** и **A.ISI.EDU**. Домен **EDU** обслуживают **SRI-NIC.ARPA** и **C.ISI.EDU**.

Примечание

Серверы могут обслуживать как непрерывные, так и разделенные (разрывные) зоны. На данной схеме сервер **C.ISI.EDU** работает в непрерывной зоне с корневым и **EDU** доменами. Сервер **A.ISI.EDU** работает с непрерывной зоной в корневом и **MIL** доменах, но также обслуживает и разрывную зону **ISI.EDU**.

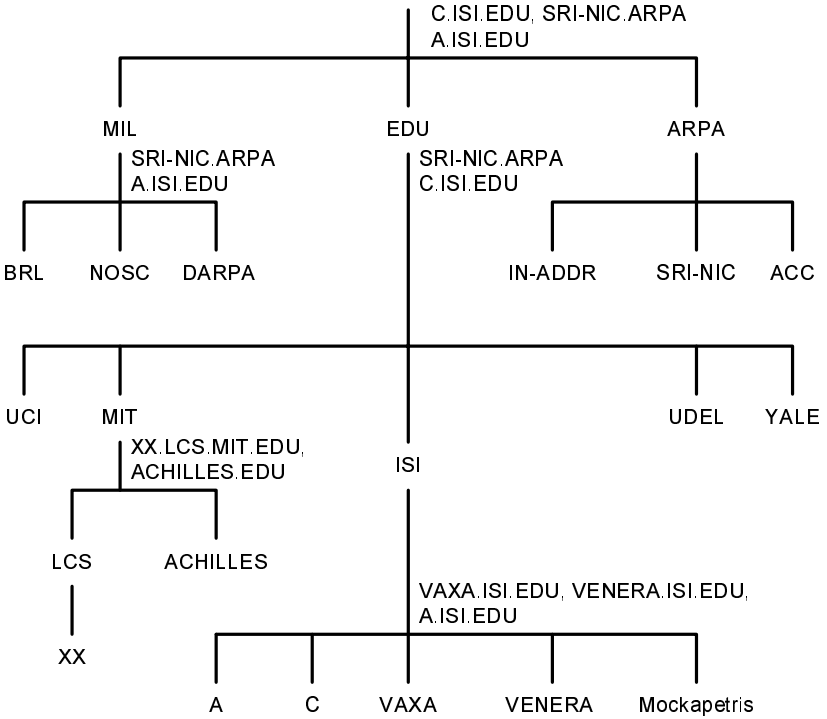


Рис. 9.18. Пример построения зоны с отдельным администрированием корневой зоны и зон **MIL**, **EDU** и **ISI.EDU**

Рассмотрим сервер имен **C.ISI.EDU**, который обслуживает в классе IN корневой домен, домены **MIL** и **EDU**. Данные его зон имеют следующий вид:

```
        IN      SOA     SRI-NIC.ARPA. HOSTMASTER.SRI-NIC.ARPA. (
                        870611   ;serial
                        1800     ;refresh
                        300      ;retry
                        604800   ;expiry
                        86400)   ;minimum TTL
                NS      A.ISI.EDU.
                NS      C.ISI.EDU.
                NS      SRI-NIC.ARPA.
MIL.    86400   NS      SRI-NIC.ARPA.
        86400   NS      A.ISI.EDU.
EDU.    86400   NS      SRI-NIC.ARPA.
        86400   NS      C.ISI.EDU.
```

```
SRI-NIC.ARPA.    A       26.0.0.73
                 A       10.0.0.51
                 MX      0 SRI-NIC.ARPA.
                 HINFO   DEC-2060 TOPS20
ACC.ARPA.        A       26.6.0.65
                 HINFO   PDP-11/70 UNIX
                 MX      10 ACC.ARPA.
USC-ISIC.ARPA.   CNAME   C.ISI.EDU.
73.0.0.26.IN-ADDR.ARPA.  PTR    SRI-NIC.ARPA.
65.0.6.26.IN-ADDR.ARPA.  PTR    ACC.ARPA.
51.0.0.10.IN-ADDR.ARPA.  PTR    SRI-NIC.ARPA.
52.0.0.10.IN-ADDR.ARPA.  PTR    C.ISI.EDU.
103.0.3.26.IN-ADDR.ARPA. PTR    A.ISI.EDU.
A.ISI.EDU. 86400 A       26.3.0.103
C.ISI.EDU. 86400 A       10.0.0.52
```

Здесь приведено содержание текстового файла главного архива. Все записи, за исключением записи RR типа SOA, укладываются в одну строчку. Класс всех записей зоны должен быть один и тот же, поэтому только первая запись RR типа SOA для зоны содержит информацию класса. Далее расположены записи RR типа NS уполномоченных серверов доменов **MIL** и **EDU** с записями так называемых "приклеенных" адресов (эти записи не являются частью зоны).

К корневому домену присоединены четыре записи RR: типа SOA, определяющая корневую зону, и три типа NS, которые содержат параметры серверов имен корневой зоны. Данные записи RR типа SOA описывают управляющие параметры зоны. Данные зоны инсталлированы на хосте **SRI-NIC.ARPA**, и главную часть зоны составляет домен **HOSTMASTER@SRI-NIC.ARPA**. Когда сервер имен загружает информацию о зоне, он устанавливает минимальное значение TTL-записей зоны в значение поля **MINIMUM** (здесь оно равно 86 400 секунд, или 1 день), это означает, что все управляющие данные зоны не могут иметь значение поля **TTL** ниже данного.

Записи RR типа NS доменов **MIL** и **EDU** устанавливают границу между корневой зоной и зонами **MIL** и **EDU**.

Примечание

В данном примере нижние (дочерние) зоны обслуживаются серверами имен, которые также обслуживают и корневую зону.

Далее, в качестве еще одного примера, приведена часть файла главного архива зоны **EDU**:

```
EDU.  IN SOA SRI-NIC.ARPA. HOSTMASTER.SRI-NIC.ARPA. (
                        870729 ;(serial)
```

```
                        1800   ;(refresh) 30 minutes
                        300    ;(retry) 5 minutes
                        604800 ;(expire)
                        86400  ;(minimum) of a day
                        )
                NS SRI-NIC.ARPA.
                NS C.ISI.EDU.
UCI             172800 NS ICS.UCI
                172800 NS ROME.UCI
ICS.UCI         172800 A 192.5.19.1
ROME.UCI        172800 A 192.5.19.31
...
UDEL.EDU.       172800 NS LOUIE.UDEL.EDU.
                172800 NS UMN-REI-UC.ARPA.
LOUIE.UDEL.EDU. 172800 A 10.0.0.96
                172800 A 192.5.39.3
...
MIT.EDU.          43200 NS XX.LCS.MIT.EDU.
                  43200 NS ACHILLES.MIT.EDU.
XX.LCS.MIT.EDU.   43200 A 10.0.0.44
ACHILLES.MIT.EDU. 43200 A 18.72.0.8
```

Примеры построения запросов и ответов серверов имен

На рис. 9.19 приведена структура сообщения DNS-запрос.

Header	OPCODE = SQUERY
Question	QNAME = SRI-NIC.ARPA., QCLASS = IN, QTYPE = A
Answer	<empty>
Authority	<empty>
Additional	<empty>

Рис. 9.19. Формат DNS-сообщения запроса и значения полей сообщения

Структура ответного сообщения от сервера имен **C.ISI.EDU** представлена на рис. 9.20.

Header	OPCODE = SQUERY, RESPONSE, AA
Question	QNAME = SRI-NIC.ARPA., QCLASS = IN, QTYPE = A
Answer	SRI-NIC.ARPA. 86400 IN A 26.0.0.73 86400 IN A 10.0.0.51
Authority	<empty>
Additional	<empty>

Рис. 9.20. Формат DNS-сообщения ответа и значения полей сообщения ответа

Заголовок (Header) сообщения ответа выглядит так же, как и заголовок запроса, за исключением того, что установлен бит **RESPONSE** (ответ) и бит **AA** (ответ пришел от уполномоченного сервера). Поле **Question** сообщения ответа выглядит так же, как и поле **Question** запроса. Поле данных сообщения ответа содержит запрашиваемые адреса.

Если бы тот же самый запрос пришел от другого сервера, который не является уполномоченным сервером домена **SRI-NIC.ARPA**, ответ выглядел бы так, как показано на рис. 9.21.

Заголовок этого сообщения не содержит бита **AA**, и параметр **TTL** имеет другое значение. Можно предположить, что эти данные получены не из зоны, а из кэша, и время нахождения данных в кэше составляет разницу между значением параметра **TTL**, установленным в зоне и полученным (8640−1777). Порядок записей в сообщении ответа RR не имеет значения.

Header	OPCODE = SQUERY, RESPONSE
Question	QNAME = SRI-NIC.ARPA., QCLASS = IN, QTYPE = A
Answer	SRI-NIC.ARPA. 1777 IN A 10.0.0.51 1777 IN A 26.0.0.73
Authority	<empty>
Additional	<empty>

Рис. 9.21. Формат DNS-сообщения ответа от неуполномоченного DNS-сервера и значения полей сообщения ответа

Рассмотрим запрос QNAME=SRI-NIC.ARPA, QTYPE=*. Этот запрос аналогичен предыдущему, только тип возвращаемых данных может принимать

различные значения. Тогда от хоста **C.ISI.EDU** мы получим ответ, представленный на рис. 9.22.

Header	OPCODE = SQUERY, RESPONSE, AA		
Question	QNAME = SRI-NIC.ARPA., QCLASS = IN, QTYPE = *		
Answer	SRI-NIC.ARPA.	86400 IN A	26.0.0.73
		A	10.0.0.51
		MX	0 SRI-NIC.ARPA.
		HINFO	DEC2060 TOPS20
Authority	<empty>		
Additional	<empty>		

Рис. 9.22. Формат DNS-сообщения ответа на запрос всех ресурсов, ассоциированных с определенным именем и значения полей сообщения ответа

Если тот же самый запрос будет направлен на сервер имен, который не уполномочен для работы с **SRI-NIC.ARPA**, ответы будут такими, как представлено на рис. 9.23 и рис. 9.24.

Header	OPCODE = SQUERY, RESPONSE		
Question	QNAME = SRI-NIC.ARPA., QCLASS = IN, QTYPE = *		
Answer	SRI-NIC.ARPA.	86400 IN A	26.0.0.73
		A	10.0.0.51
Authority	<empty>		
Additional	<empty>		

Рис. 9.23. Формат DNS-сообщения ответа от неуполномоченного сервера имен на запрос всех ресурсов, ассоциированных с определенным именем, и значения полей сообщения ответа

Header	OPCODE = SQUERY, RESPONSE				
Question	QNAME = SRI-NIC.ARPA., QCLASS = IN, QTYPE = *				
Answer	SRI-NIC.ARPA.	1290	IN	HINFO	DEC-2060 TOPS20
Authority	<empty>				
Additional	<empty>				

Рис. 9.24. Формат DNS-сообщения ответа от другого неуполномоченного сервера имен на запрос всех ресурсов, ассоциированных с определенным именем, и значения полей сообщения ответа

Оба ответа не содержат бита **AA**. Кроме того, эти два сообщения содержат различные значения параметра **TTL**, а это значит, что оба отвечающих сервера поместили эти данные в кэш в различное время и первый кэшировал ответы с типом **QTYPE**=A, а второй — с типом HINFO.

Рассмотрим запрос QNAME=SRI-NIC.ARPA, QTYPE=MX. Такой запрос может быть сделан при поиске почтовых серверов. Ответ сервера **C.ISI.EDU** будет такой, как показано на рис. 9.25.

Ответ содержит секции типа MX записи RR. Кроме того, дополнительные секции пакета содержат информацию об адресах почтовых серверов.

Составим запрос QNAME=SIR-NIC.ARPA, QTYPE=A, в котором пользователь ввел неверное имя домена. Ответ от **C.ISI.EDU** будет таким, какой показан на рис. 9.26.

Сообщение указывает, что такое имя не существует (выставлен флаг ошибки — **RCODE**). Запись начала зоны типа SOA, содержащаяся в поле **Authority**, указывает, что имя домена, указанное в запросе, не существует в данной зоне типа SOA как минимум 86 400 секунд — это срок обновления информации зоны.

Header	OPCODE = SQUERY, RESPONSE, AA
Question	QNAME = SRI-NIC.ARPA., QCLASS = IN, QTYPE = MX
Answer	SRI-NIC.ARPA. 86400 IN MX 0 SRI-NIC.ARPA.
Authority	<empty>
Additional	SRI-NIC.ARPA. 86400 IN A 26.0.0.73 A 10.0.0.51

Рис. 9.25. Формат DNS-сообщения ответа на запрос поиска почтовых серверов — (типа MX) и значения полей сообщения ответа

Header	OPCODE = SQUERY, RESPONSE, AA, RCODE = NE
Question	QNAME = SRI-NIC.ARPA., QCLASS = IN, QTYPE = A
Answer	<empty>
Authority	SOA SRI-NIC.ARPA. HOSTMASTER.SRI-NIC.ARPA. 870611 1800 300 604800 86400
Additional	<empty>

Рис. 9.26. Формат DNS-сообщения ответа на запрос адреса несуществующего ресурса и значения полей сообщения ответа

Отправим запрос QNAME=BRL.MIL, QTYPE=A на сервер **C.ISI.EDU**. Ответное сообщение будет иметь вид, показанный на рис. 9.27.

Header	OPCODE = SQUERY, RESPONSE		
Question	QNAME = BRL.MIL, QCLASS = IN, QTYPE = A		
Answer	<empty>		
Authority	MIL.	86400 IN NS	SRI-NIC.ARPA
		86400 NS	A.ISI.EDU
Additional	A.ISI.EDU.	A	26.3.0.103
	SRI-NIC.ARPA	A	26.0.0.73
		A	10.0.0.51

Рис. 9.27. Формат DNS-сообщения ответа на запрос адреса ресурса неуполномоченного сервера имен и значения полей сообщения ответа

Поле ответа данного сообщения не содержит информации. Отвечающий сервер **C.ISI.EDU** не уполномочен для работы с доменом **MIL**, и он возвращает информацию об уполномоченных для работы с доменом **MIL** серверах имен — **A.ISI.EDU** и **SRI-NIC.ARPA**.

Примеры работы программы разрешения имен

Следующие примеры иллюстрируют работу программы разрешения имен на компьютере клиента. Предполагаем, что программа разрешения начинает работу с очищенным кэшем (как бывает после перезагрузки компьютера), хосты системы расположены по адресам 26.x.x.x, и структура SBELT содержит следующую информацию:

```
match count = -1
SRI-NIC.ARPA.    26.0.0.73       10.0.0.51
A.ISI.EDU.       26.3.0.103
```

Эта информация определяет серверы, к которым можно обращаться за информацией. Параметр `match count = -1` обозначает "удаление" этих серверов (этот параметр играет роль на последующих стадиях алгоритма разрешения).

1. Предположим, что первый запрос пришел от программы локального почтового клиента, который хочет отправить почту по адресу **PVM@ISI.EDU**. Почтовый клиент запрашивает записи типа MX домена **ISI.EDU**.
2. Программа разрешения просматривает кэш и ищет записи RR типа MX для **ISI.EDU**. Но поскольку кэш пуст, она организует опрос серверов имен и ищет записи NS доменов **ISI.EDU**, **EDU** и корневого. Предпо-

ложим, что и этот опрос ничего не дал. Тогда программа обращается за информацией к структуре SBELT — она копирует ее в структуру SLIST.

3. Теперь программа разрешения должна выбрать один из трех адресов серверов имен. Поскольку сеть локального хоста 26.x.x.x, выбор делается между 26.0.0.73 и 26.3.0.103 (как правило, берется первый, по порядку следования, адрес). Затем по выбранному адресу отправляется запрос (рис. 9.28).

Header	OPCODE = SQUERY
Question	QNAME = ISI.EDU., QCLASS = IN, QTYPE = MX
Answer	<empty>
Authority	<empty>
Additional	<empty>

Рис. 9.28. Формат DNS-сообщения запроса серверов типа MX и значения полей сообщения запроса

4. Если ответ не приходит через определенный промежуток времени, запрос отправляется на другой адрес (при этом запрос может быть отправлен и по ранее использованному адресу).

Предположим, что от сервера **SRI-NIC.ARPA** ответ получен (рис. 9.29).

Header	OPCODE = SQUERY, RESPONSE			
Question	QNAME = ISI.EDU., QCLASS = IN, QTYPE = MX			
Answer	<empty>			
Authority	ISI.EDU.	172800	IN NS	VAXA.ISI.EDU.
			NS	A.ISI.EDU.
			NS	VENERA.ISI.EDU.
Additional	VAXA.ISI.EDU.	172800	A	10.2.0.27
		172800	A	128.9.0.33
	VENERA.ISI.EDU.	172800	A	10.1.0.52
		172800	A	128.9.0.32
	A.ISI.EDU.	172800	A	26.3.0.103

Рис. 9.29. Формат DNS-сообщения ответа на запрос серверов типа MX от неуполномоченного сервера и значения полей сообщения ответа

Программа разрешения кэширует ответ и на его основе строит новый SLIST, который будет выглядеть следующим образом (описание алгоритма

изменения параметра расстояния "match count" выходит за рамки данной книги):

```
match count = 3
A.ISI.EDU.       26.3.0.103
VAXA.ISI.EDU.    10.2.0.27        128.9.0.33
VENERA.ISI.EDU.  10.1.0.52        128.9.0.32
```

Теперь программа разрешения делает запрос по искомому типу MX. Ответ будет выглядеть примерно так, как показано на рис. 9.30.

После получения ответа, программа разрешения добавляет эту информацию в кэш и возвращает запись RR типа MX клиенту.

Header	OPCODE = SQUERY, RESPONSE, AA			
Question	QNAME = ISI.EDU., QCLASS = IN, QTYPE = MX			
Answer	ISI.EDU.	MX 10 VENERA.ISI.EDU. MX 20 VAXA.ISI.EDU		
Authority	<empty>			
Additional	VAXA.ISI.EDU.	172800	A	10.2.0.27
		172800	A	128.9.0.33
	VENERA.ISI.EDU.	172800	A	10.1.0.52
		172800	A	128.9.0.32

Рис. 9.30. Формат DNS-сообщения ответа на запрос по типу MX и значения полей сообщения ответа

Утилита NSLOOKUP

Утилита NSLOOKUP (Name server look up) входит в состав почти всех пакетов программного обеспечения для работы с семейством протоколов TCP/IP. Она предоставляет пользователю возможность просматривать состояние баз данных системы DNS, организовывать поиск определенных записей баз данных системы DNS, определять IP-адреса или имена ресурсов систем DNS удаленных хостов. Например, пользователь может просмотреть список почтовых хостов или серверов имен указанного домена. Утилита NSLOOKUP чаще всего используется сетевыми и системными администраторами для настройки параметров работы локальных сетей и прикладного программного или сетевого обеспечения.

Для обмена с удаленными (или локальными) базами данных системы DNS утилита NSLOOKUP использует механизм "запросов и ответов"

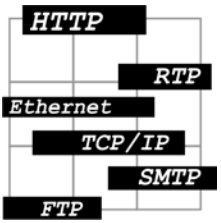
DNS-протокола, параметры запросов пользователь может установить самостоятельно.

При использовании утилиты пользователь в командной строке определяет имя или адрес DNS-сервера, устанавливает параметры запроса и получает ответ. Ниже представлен пример поиска записей почтовым сервером (тип MX) на сервере **sun.com**.

```
C:\>nslookup
Default Server:  team1.mcs.marinform.ru
Address:  195.5.151.50

> www.sun.com
Server:  team1.mcs.marinform.ru
Address:  195.5.151.50
Name:    www.sun.com
Addresses:  192.9.9.100, 192.9.48.5

> set type=MX
> sun.com
Server:  team1.mcs.marinform.ru
Address:  195.5.151.50

sun.com preference = 40, mail exchanger = mars.sun.com
sun.com preference = 50, mail exchanger = dollar.usec.Sun.COM
sun.com preference = 60, mail exchanger = franc.usec.Sun.COM
sun.com preference = 5, mail exchanger = saturn.Sun.COM
sun.com preference = 10, mail exchanger = mercury.Sun.COM
sun.com preference = 20, mail exchanger = venus.Sun.COM
mars.sun.com    internet address = 192.9.22.1
dollar.usec.Sun.COM     internet address = 192.9.51.3
franc.usec.Sun.COM      internet address = 192.9.51.4
saturn.Sun.COM  internet address = 192.9.25.2
mercury.Sun.COM internet address = 192.9.25.1
venus.Sun.COM   internet address = 192.9.25.5
> exit
```

Все реализации утилиты NSLOOKUP поддерживают стандартный набор команд для построения запросов. Список возможных опций запросов, реализованных в конкретной версии утилиты NSLOOKUP, можно получить командой HELP.

Глава 10



Протоколы электронной почты — SMTP, POP3, IMAP4

В этом разделе будет рассмотрена работа наиболее распространенного приложения сети Интернет — электронной почты (e-mail) и наиболее популярных протоколов, которые используются этим приложением. Скорость, эффективность и простота использования концепции e-mail сделали этот сервис наиболее распространенным в сети Интернет. Сообщения электронной почты e-mail могут передаваться и достигать своего места назначения через весь земной шар за считанные минуты. Пользователи Интернет ласково называют обычную почту "улиткой" из-за ее относительной медлительности.

Протокол SMTP

Основным протоколом передачи электронной почты является SMTP (Simple Mail Transfer Protocol, Простой протокол передачи почты). Протокол прикладного уровня SMTP обеспечивает передачу сообщений электронной почты между произвольными узлами сети Интернет. Существует огромное число коммерческих и свободно распространяемых программных пакетов для работы с e-mail, использующих SMTP-сервис. Среди них такие популярные программы, как Microsoft Mail, Microsoft Exchange и Lotus cc:Mail.

SMTP представляет собой независимый от транспортной подсистемы протокол, для работы которого необходим только транспортный канал передачи потока данных. Собственные механизмы промежуточного хранения и обеспечения надежной доставки позволяют протоколу SMTP использовать различные транспортные службы. Сообщения протокола SMTP могут передаваться по любому транспортному каналу, удовлетворяющему требованиям передачи данных SMTP через различные сети или группы сетей, например, TCP, X.25 и др.

Протокол SMTP обеспечивает как передачу сообщений в адрес одного получателя, так и тиражирование нескольких копий для передачи в разные адреса. Протокол SMTP может быть использован для передачи не только текстовых сообщений, но и двоичной информации, рисунков, исполняемых файлов и др. Двоичная информация, находящаяся в сообщении, перед

отправкой специальным образом форматируется и, как правило, кодируется в 7-битный вид. Современные расширения протокола SMTP — ESMTP (Enhanced SMTP) позволяют передавать данные в 8-битном виде. Форматирование и кодирование разнородной пользовательской информации производится программным обеспечением оболочки электронной почты.

После того как пользователь закончил формирование своего сообщения, т. е. вставил необходимые картинки, присоединил к текстовому сообщению дополнительные файлы или исполняемые программы, сообщение помещается в определенную структуру — конверт сообщения. Конверт сообщения содержит информацию о структуре и составе передаваемых данных, дату формирования сообщения, имена и адреса отправителя, имена и адреса получателя или группы получателей, темы сообщения и др.

По сути, конверт сообщения является блоком данных , который структурирует данные, сформированные приложением, и представляет их в стандартном виде протокола передачи сообщений — SMTP.

Прежде чем переходить к рассмотрению работы протокола передачи почты — SMTP, необходимо коротко остановиться на описании структур и форматов данных, в которые укладываются разнородные пользовательские данные — текст, изображения, звуки и т. п. при передаче по электронной почте.

Формат почтовых сообщений MIME

В данном разделе рассмотрен формат почтовых сообщений и стандарт MIME (Multipurpose Internet Mail Extensions), предназначенный для работы с составными и нестандартными сообщениями электронной почты.

К сожалению, рамки данной книги не позволяют подробно разобрать все типы почтовых сообщений, разновидности кодировок и представлений данных. Доступное и подробное изложение этой темы может составить несколько объемистых томов. В данном разделе будут кратко изложены наиболее распространенные типы форматов и их взаимодействие между собой.

Любое сообщение состоит из конверта сообщения и тела сообщения. Конверт содержит информацию, необходимую для доставки и обработки сообщения. Тело сообщения включает информацию, которую отправитель передает получателю. Иными словами, конверт представляет собой каркас, на который накладываются данные пользователя. Конверт и данные в сообщении тесно взаимосвязаны друг с другом, все почтовые системы для построения конверта используют информацию тела сообщения. Структура конверта, как правило, задается локальным почтовым программным обеспечением. Поэтому, прежде чем отправлять данные в определенном конверте, нужно убедиться, что получатель этих данных сумеет извлечь их из конверта.

В простейшем случае конверт сообщения состоит только из заголовка. Заголовок отделен от тела сообщения пустой строкой. Заголовок состоит из нескольких строк, содержащих информацию о сообщении.

Каждая строка заголовка содержит имя поля заголовка и данные поля заголовка. Имя поля отделено от данных символом ":". Названия полей должны состоять из символов ASCII от 32 до 126. Каждая строка заканчивается символами "CR" и "LF".

Примечание

Далее для упрощения записи при описании полей знак ":" будет опущен.

Некоторые поля сообщения должны иметь фиксированную структуру, например, поля `Date`, `From`, другие поля могут иметь произвольную структуру, например, поля `Subject`, `Comments` и др.

Рассмотрим состав и структуру основных полей сообщения на следующем простейшем примере:

```
From:      "Somebody M.S." <mike@joke.ru>
To:         zol@fun.ru
Date:       Wed, 10 Sep 1997 16:17:50 +0300
Subject:    Hello
Message-ID: <some.string@joke.ru>

Hello!
How do you do?
```

Как вы видите, сообщение довольно немногословно. Большая его часть представляет собой заголовок. Разберем подробнее состав и структуру полей конверта сообщения.

- Поле `From` содержит адрес отправителя. Как правило, адрес автора сообщения устанавливается в параметрах настройки программы почтового клиента. Если бы отправитель поручил кому-либо другому отправить за него сообщение, например, своему секретарю, сообщение содержало бы поле `Sender`. В этом поле был бы указан реальный отправитель сообщения, т. е. идентификатор пользователя, который вошел в почтовую систему и отправил данное письмо, пользуясь, например, настройками программы клиента автора сообщения. Например, если бы сообщение было отправлено секретарем **lisa@joke.ru**, оно содержало бы следующее:

  ```
  From: mike@joke.ru (mike@joke.ru)
  Sender: lisa@joke.ru
  ```

  Поле `From` может содержать несколько адресов отправителей, от имени которых было отправлено сообщение, но не должно содержать групповой адрес, например:

```
From: mike@joke.ru,
      vlad@joke.ru
Sender: lisa@joke.ru
```

- Поле `To` содержит адрес(а) получателя(ей) — кому предназначено данное сообщение. Любое сообщение должно содержать либо поле `To`, либо поле `Cc`, содержащее адреса получателей копий данного сообщения, например:

```
To:  Oden@Mad-Host,
     Irving@Other-Host
Cc:  Simmons@bra.x400.icl.co.uk, Blob@ultra.x400.icl.uk
```

- Поле `Date` содержит дату отправки сообщения почтовой системой. Поле даты содержит дату, время и часовую зону. Формат даты также зависит от почтовой системы.
- Поле `Subject` содержит тему данного сообщения, заполняется пользователем самостоятельно, в соответствии с содержанием информационной части сообщения.
- Поле `Message-ID` содержит уникальный идентификатор сообщения. Он используется для ссылок на данное сообщение других сообщений и идентификации частей данного сообщения. Состав идентификатора определяется типом почтовой системы и, как правило, состоит из строки символов и адреса хоста отправителя.
- В сообщении может быть поле `Reply-To` — где указан почтовый ящик, куда следует направлять ответы на данное сообщение. Например, если отправитель, имеющий адрес **mike@joke.ru**, захочет, чтобы ответ на его сообщение получил его секретарь, он может указать:

```
Reply-To: lisa@joke.ru
```

  Ответы на сообщение, в заголовке которого есть поле `Reply-To`, отправляются по указанному в этом поле адресу. Если этого поля нет, ответы направляются по адресу, указанному в поле `From`.

Минимальный заголовок сообщения должен содержать поля: `Date`, `From`, `To` или `Cc`. Все остальные поля не являются обязательными в формате заголовка сообщения и их присутствие и содержимое непосредственно зависит от типа используемой почтовой системы.

При передаче сообщения через промежуточные почтовые серверы к заголовку достраиваются записи `Received`, содержащие поля с адресами и временем обработки сообщения промежуточными серверами, например:

```
Received: from ns.joke.ru (ns.joke.ru [199.86.42.18])
by bulldozer.arcom.spb.su (8.8.5/t/97-Mar-14) with ESMTP id QAA10700
```

```
for <zol@joke.ru>; Wed, 10 Sep 1997 16:24:02 +0400 (MSD)
```

Заголовок также может содержать запись типа и версии программного обеспечения клиента (`X-Mailer`), комментарии (`Comments`), приоритетность (`Priority`) и др., например:

```
Priority: normal
X-mailer: Pegasus Mail v3.22
```

Очередность полей заголовка определяется почтовой системой, но, как правило, они размещаются следующим образом: `Received`, `Date`, `From`, `Subject`, `Sender`, `Reply-To`, `To`, `Cc`, `Comment`, `X-Mailer`, `Message-ID` и др.

На начальном этапе развития электронной почты было вполне достаточно передавать между пользователями только текстовые сообщения, которые содержали только символы из набора ASCII. Если пользователю требовалось отправить изображение или исполняемый файл, он кодировал его символами ASCII и вставлял в текст сообщения.

С ростом популярности электронной почты и расширением возможностей компьютеров и линий передачи, назрела необходимость передавать по почте тексты, написанные на разных национальных языках, изображения, исполняемые файлы, видео и другие бинарные данные в унифицированном формате. Старая схема передачи таких данных становилась все более неудобной из-за того, что:

- Разные клиенты работали с различными типами кодирования, что часто приводило к невозможности прочитать полученные данные адресатом.
- Не была определена структура размещения и идентификации типа закодированных данных. Чтобы понять, что представляет собой полученная информация, ее надо было вынуть из сообщения и декодировать.
- В результате было сильно затруднено, а реально и невозможно, развитие программных средств просмотра данных сообщения "на лету", т. е. чтобы программное обеспечение клиента могло автоматически распознавать тип данных и представлять их пользователю сразу в момент просмотра сообщения.

В общем, все это было очень неудобно, особенно, если вам приходится постоянно работать с электронной почтой, например, прочитывать и отвечать на несколько десятков электронных писем в день.

Поэтому, для более удобной работы с составными и нестандартными (не ASCII) сообщениями и был разработан стандарт MIME (Multipurpose Internet Mail Extensions, Многоцелевое расширение электронной почты).

Стандарт MIME не заменяет, а расширяет существующий до сих пор способ форматирования сообщений. MIME представляет собой новый формат представления данных, который позволяет почтовому клиенту удобный и гибкий интерфейс работы с электронной почтой. MIME — это, образно говоря, новый конверт, в который вы можете положить разнообразный текст, графические изображения, видеофрагменты и отправить адресату, который,

в свою очередь, может быстро и без проблем воспользоваться всей полученной информацией. Сообщение построено с использованием MIME, если в его заголовке присутствуют следующие поля:

- Поле `MIMI-Version`. Стандарт MIME идентифицируется в почтовом сообщении полем `MIME-Version`, которое содержит строку версии MIME расширения данного сообщения.

  Например: `MIME-Version: 1.0`.

- Поле `Content-Type` указывает состав сообщения:

  - `text` — сообщение содержит текстовую информацию в виде последовательности символов из набора, указанного параметром `charset` (`us-ascii`, `iso-8859-1`, `koi8-r`, `windows-1251` и др.).

    Например: `Content-Type: text/plain; charset="iso-8859-1"`

  - `multipart` — означает, что данное сообщение состоит из нескольких отдельных блоков, каждый из которых описывает свой состав самостоятельно. Данный тип имеет параметр boundary, который содержит строку-разделитель частей сообщения.

    Например: `Content-Type: multipart/mixed; boundary="this is a separator"`

  - `application`, `image`, `audio`, `video` — означают, что сообщение содержит двоичные данные определенного типа.

    Например: `Content-Type: application/msword; name="file.doc"` или `Content-Type: image/gif; name="pic.gif"`

  - `message` — сообщение содержит другое сообщение.

    Например, строка `Content-Type: message/RFC-822` указывает, что далее следует сообщение, отформатированное в соответствии с правилами, сформулированными в документе RFC-822.

Примечание

RFC-822 — это основополагающий документ форматов почтовых сообщений. Он описывает только самые основы структуры почтовых сообщений без каких-либо расширений. Если указано, что сообщение сформировано на основании RFC-822, то это значит, что данное сообщение составлено в каноническом виде.

- Поле `Content-Transfer-Encoding` содержит идентификатор типа кодировки, используемой в данном сообщении или его части: `base64`, `quoted-printable`, `7bit`, `8bit`, `binary` или др.

  Особенность типов кодировки: `7bit`, `8bit`, `binary` состоит в том, что они не подразумевают никакого кодирования. Эти типы только обозначают, что данные представлены в определенном виде. `7bit` — данные содержат только ASCII-символы, `8bit` и `binary` — указывают на то, что

данные содержат не только ASCII-символы. Разница между типами `8bit` и `binary` состоит в ограничении длины строки данных. У типа `8bit` строка ограничена длиной строки SMTP-протокола — 1000 байт, а у типа `binary` таких ограничений нет.

Примечание

Кодировка "8bit", как правило, используется для передачи текстовых сообщений на различных национальных языках, а кодировка "binary" — для передачи графики, видео и т. п. Для передачи данных в этих кодировках серверы SMTP должны уметь работать с 8-битными данными, т. е. поддерживать расширения протокола SMTP —ESMTP.

Кодировки "base64" и "quoted-printable" преобразуют последовательность символов данных сообщения из набора 00-FF в последовательность символов ASCII.

Кодировка "quoted-printable", как правило, предназначена для кодирования текстов на различных национальных языках, содержащих символы из второй части таблицы кодировки (7F-FF). Кодировка "quoted-printable" удобна тем, что символы верхней части таблицы кодировки (00-7F) представляются в своем естественном виде. Это делает латинский текст письма хоть немного более удобным для чтения (чего, к сожалению, не скажешь о кириллице). Описание алгоритма кодирования quoted-printable не входит в рамки данной книги, однако для понимания принципов его работы достаточно знать следующее:

- каждый символ сообщения может быть представлен в виде символа "=" и двузначного шестнадцатеричного кода символа. Например, символ "=" может быть представлен как '=3D';
- символы, десятичные коды которых находятся в диапазонах от 33 до 60 и от 62 до 126, представляются в виде ASCII-символов, т. е. символ "1" в кодировке "quoted-printable" будет '1';
- остальные правила касаются представления знаков переноса строки, пробела и т. п. Все это вы можете найти, например, в документе RFC-1341.

Кодировка "base64" используется, как правило, для представления "нечитаемой" части сообщения — изображений, бинарных файлов, видео и др. Алгоритм кодирования чем-то напоминает процедуру сжатия (или архивации) данных наоборот. Описание алгоритма кодирования "base64" не входит в рамки данной книги, однако, следует отметить, что для понимания принципов его работы достаточно знать следующее:

- битовый поток разбивается на сегменты по 24 бита, которые, в свою очередь, делятся на четыре части по 6 бит;
- каждая такая часть кодируется одним из 64 ASCII-символов (отсюда название — "base64") в соответствии с таблицей, в которую входят

прописные и заглавные буквы английского алфавита, цифры и знаки "+" и "/".

Кодировки "base64", "quoted-printable", "7bit" поддерживаются практически всеми почтовыми серверами.

Рассмотрим теперь возможности и спецификацию MIME на примере следующей части сообщения:

```
...

Date: 03 Jun 1997 10:28:00 +0300
Mime-Version: 1.0
From: john@team.fun.ru (john nuttor)
Subject: Hello!
To: zol@team1.mcs.tsi.ru (Sergey Zolotov)
Content-Type: multipart/mixed;
boundary="Boundary_3393cd7a29_Level0"

--Boundary_3393cd7a29_Level0
Content-Type: text/plain; charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable

=DE=D4=CF-=D0=CF=D0=C1=CC=CF, =C4=CF=D5=DE=C9=D7=C1=D4=D8,
=C9 =D6=C4=C1=
=D4=D8, =CB=CF=C7=C4=C1 =C5=C7=CF =D0=C5=D2=C5=CD=C1=CE=D1=D4.

--Boundary_3393cd7a29_Level00
Content-Type: application/octet-stream;name="Register.url"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="Register.url"

W0ludGVybmV0U2hvcnRjdXRdDQpVUkw9aHR0cDovL3d3dy50ZWxlY29td2ViLmNvbS9yZW
dpc3Rlci5odG0NCg0K
--Boundary_3393cd7a29_Level0--
```

Тип структуры данного сообщения определен как `multipart` с подтипом `mixed`. Это означает, что сообщение состоит из нескольких частей разных форматов и на порядок представления частей не накладываются никакие ограничения. Данный тип сообщения обязан содержать параметр `boundary`, который представляет собой строку, служащую разделителем частей сооб-

щения. Таким разделителем в данном сообщении является строка `Boundary_3393cd7a29_Level0`. Она делит сообщение на две части.

- Первая часть типа `text/plain` содержит закодированное текстовое сообщение в кодировке `charset="iso-8859-1"`. Тип кодирования `quoted-printable`. Это означает, что после декодирования первой части ее текст будет иметь кодировку `iso-8859-1` и содержать символы вне набора ASCII, что следует из общего вида закодированной информации. Эта часть представляет собой обычное сообщение, которое мы читаем в окне текста сообщения программы почтового клиента.
- Вторая часть сообщения имеет тип `application/octet-stream` — обозначение типа информации в двоичных кодах, который программа почтового клиента не сумела сопоставить ни с одним из известных ей типов. Тип имеет название: параметр `name`, который указывает, что данные содержат URL (Uniform Resourse Locator, Универсальный указатель ресурсов) какого-то объекта. Текст второй части сообщения закодирован в `base64`, на что указывает строка `Content-Transfer-Encoding: base64`. Эта часть содержит дополнительный параметр `Content-Disposition`, который определяет тип представления этой части данных. Тип представления данных `attachment` с параметром `filename=Register.url` означает, что вторая часть нашего сообщения будет обозначена программой почтового клиента как присоединенный файл с именем `Register.url`.

Теперь совсем не сложно представить, каким должно быть сообщение, чтобы оно содержало, например, текст с картинкой.

Первую часть сообщения можно оставить без изменения (разве что только изменить текст):

```
...
Content-Type: multipart/mixed;
boundary="Boundary_example_boundary_picture"

--Boundary_example_boundary_picture
Content-Type: text/plain; charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable

Look
```

а вторую представить так:

```
-- Boundary_example_boundary_picture
Content-Type: image/jpg
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="look.jpg"

<... picture...>
--Boundary_3393cd7a29_Level0--,
```

где <...picture...> — закодированное при помощи алгоритма "base64" содержание картинки из файла look.jpg. Текст сообщения сокращен до одного слова: Look, поскольку в кодировке quoted-printable символы ASCII передаются в своем "естественном" виде. Если вы хотите, чтобы картинка появлялась в самом тексте сообщения, достаточно установить параметр Content-Disposition: inline. Это означает, что программа почтового клиента разместит картинку рядом с текстом письма, т. е. пользователю не нужно будет предпринимать какие-либо дополнительные действия для того, чтобы увидеть картинку.

В разобранном выше примере сообщение состоит из двух частей. Однако каждая из описанных частей может быть самостоятельным сообщением и передаваться отдельно. Кроме того, каждая часть сообщения может включать в себя несколько других частей и т. д., образуя своеобразную иерархию вложенности. Манипулируя параметрами MIME, вы можете составлять практически любые конструкции объектов и передавать их по электронной почте.

Рамки данной книги не позволяют более детально остановиться на обсуждении значений MIME-параметров и их комбинаций. Теперь, после знакомства с составом и форматами передаваемых сообщений, перейдем к рассмотрению протокола передачи этих сообщений.

Взаимодействие компонентов SMTP

Протокол SMTP использует следующую модель взаимодействия: по запросу пользователя SMTP-отправитель (sender-SMTP) устанавливает двусторонний канал с SMTP-получателем (receiver-SMTP). SMTP-получателем может быть как хост назначения почтового сообщения, так и какой-либо промежуточный хост. Команды протокола SMTP генерируются отправителем и отправляются SMTP-получателю, который, в свою очередь, отправляет ответы обработки полученных команд SMTP-отправителю.

Схема работы протокола SMTP изображена на рис. 10.1. Простейший алгоритм работы протокола выглядит следующим образом:

1. После установления канала SMTP-соединения по любому из транспортных протоколов SMTP-отправитель посылает команду MAIL, идентифицирующую атрибуты отправителя почты, например, его адрес. Если SMTP-получатель может принять почтовое сообщение, он отправляет в ответ команду OK.
2. После этого SMTP-отправитель отправляет команду RCPT, идентифицирующую атрибуты получателя почты, например, адрес почтового ящика. Если SMTP-получатель готов принять почту в данный почтовый ящик, он отвечает командой OK, если нет, он отвечает отказом принять почту в указанный почтовый ящик. Если отправитель указал несколько почтовых

ящиков, в которые следует поместить сообщение, то SMTP-получатель может отказать части из них, при этом транзакция соединения не заканчивается.

3. SMTP-отправитель отправляет данные SMTP-получателю. Если получатель успешно принял все данные, он отправляет команду `OK`.

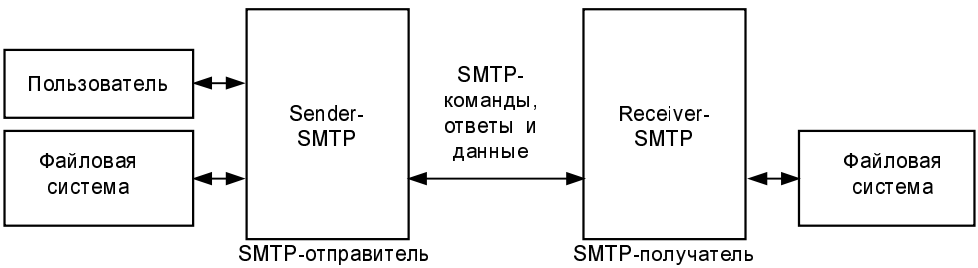


Рис. 10.1. Схема работы SMTP-протокола

Протокол SMTP поддерживает несколько механизмов передачи почты: напрямую от хоста пользователя-отправителя к хосту пользователя-получателя, когда два хоста соединены между собой напрямую через один и тот же транспортный сервис; или через серверы протокола SMTP (т. е. хосты — посредники), когда отправитель и получатель не могут соединиться напрямую.

Аргументы команды `MAIL` содержат имя домена отправителя почты, т. е. SMTP-адрес почтового ящика — обратный адрес. Аргументы команды `RCPT` содержат адрес почтового ящика получателя сообщения. Эти параметры не изменяются в процессе передачи сообщения и используются SMTP-серверами, например, для возврата недоставленных сообщений.

SMTP-команды имеют строго определенный синтаксис. Ответы обработки команд имеют численные коды. Команды и ответы представляют собой строки из символов ASCII. Команды и ответы не чувствительны к регистру символов (чего нельзя сказать об именах пользовательских почтовых ящиков). Если транспортный сервис не поддерживает 8-битный канал, каждый из 7-битных символов передается со старшим (нулевым) битом = 0.

Организация информационного обмена

В процессе информационного обмена приложения SMTP-клиент (отправитель) и SMTP-сервер (получатель) обмениваются специальными сообщениями — командами протокола SMTP.

Транзакция SMTP-отправки почты состоит из трех шагов. Начало транзакции задает команда `MAIL`, идентифицирующая отправителя, затем следует

серия команд RCPT, определяющих почтовые ящики получателей, затем подается команда DATA для передачи текста сообщения. Завершает передачу данных командная строка закрытия транзакции.

- Формат команды MAIL выглядит следующим образом:

  MAIL <SP> FROM:<reverse-path> <CRLF>

  где <SP> — символ пробела, <CRLF> — символы возврата каретки и перевода строки, <reverse-path> — имя почтового ящика отправителя сообщения. Эта команда говорит SMTP-получателю, что открывается новая транзакция передачи почты и получатель должен очистить все старые таблицы и буферы данных для приема нового сообщения. В случае возникновения ошибки SMTP-получатель отправляет по адресу <reverse-path> уведомление о невозможности открытия транзакции. Если SMTP-транзакция открыта успешно, отправителю возвращается команда "согласен 250 раз" — 250 OK, где 250 — код возврата успешного выполнения команды.

  Все ответы SMTP-серверов или получателей имеют код, который обозначает статус обработки команды. Структура кодов строго определена, обработка конкретной команды может вернуть только строго обусловленные для нее коды возврата. Все коды возврата построены по определенной структуре и образуют сложную иерархическую систему. Каждая цифра или комбинация цифр кода возврата несет в себе информацию о породивших ее условиях. К сожалению, описание структуры кодов возврата SMTP-сообщений выходит за рамки данной книги. Но если вы хотите программировать какой-либо SMTP-сервис, вам необходимо тщательно изучить структуру кодов возврата. Подробное описание этой структуры вы можете найти в документах RFC, например, RFC-821.

- Вторым шагом процесса передачи почтового сообщения будет команда RCPT:

  RCPT <SP> TO:<forward-path> <CRLF>

  где <forward-path> — имя почтового ящика адресата. Данная команда может повторяться несколько раз для различных адресов получателей сообщения. Если какой-либо из получателей неизвестен в передающей системе, возвращается код 550 — "на указанный адрес почта не может быть отправлена — пользователь не существует". Если пользователь найден, отправителю возвращается код 250 OK.

- На третьем шаге транзакции передается команда DATA:

  DATA <CRLF>

  На эту команду SMTP-получатель возвращает код 354, означающий готовность принимать данные. После этого отправитель начинает передавать текст сообщения: заголовок и тело сообщения. Каждая строка сообщения заканчивается символом <CRLF>. При получении строки

окончания передачи текста — последовательности <CRLF>.<CRLF>, получатель возвращает код 250 OK. Это означает, что все данные приняты. В противном случае получателем будет передан код ошибки неверного завершения транзакции.

Примечание

Необходимо помнить, что длина строки данных SMTP-сообщения, вместе с символами возврата каретки и перевода строки <CRLF> не должна превышать 1000 байтов.

Ниже приведен пример транзакции передачи сообщения, использующей описанные выше команды. Пример показывает передачу почты от пользователя **Smith**, домена **Alpha.ARPA** пользователям домена **Beta.ARPA** (в данном и следующих примерах введены следующие обозначения: S: — SMTP-отправитель, R: — SMTP-получатель).

```
S: MAIL FROM:<Smith@Alpha.ARPA>
R: 250 OK
S: RCPT TO:<Jones@Beta.ARPA>
R: 250 OK
S: RCPT TO:<Green@Beta.ARPA>
R: 550 No such user here
S: RCPT TO:<Brown@Beta.ARPA>
R: 250 OK
S: DATA
R: 354 Start mail input; end with <CRLF>.<CRLF>
S: hey hey hey...
S:... how do you do, Beta ?
S: <CRLF>.<CRLF>
R: 250 OK
```

В соответствии с представленной SMTP-транзакцией, почтовое сообщение будет направлено пользователям **Jones** и **Brown**. Пользователь **Green** не получит это сообщение, поскольку для него не организован почтовый ящик в домене **Beta**.

Может случиться так, что адрес почтового ящика (<forward-path>) содержит неверную информацию или находится в другом домене, а SMTP-получатель знает правильный адрес. Тогда SMTP-получатель информирует об этом SMTP-отправителя двумя способами.

- 251 User not local; will forward to <forward-path>. Это ответное сообщение говорит отправителю, что SMTP-получатель знает почтовый ящик адресата сообщения <forward-path> и берется самостоятельно переправить туда сообщение.

- `551 User not local; please try <forward-path>`. Этот ответ говорит отправителю, что SMTP-получатель знает почтовый ящик получателя сообщения по адресу `<forward-path>`, но не берется переправить его туда самостоятельно и рекомендует это сделать отправителю.

Например, описанный выше сценарий мог развиваться следующим образом:

```
S: RCPT TO:<Green@Beta.ARPA>
R: 251 User not local; will forward to <Green@Batman.ARPA>
```

или

```
S: RCPT TO:<Green@Beta.ARPA>
R: 551 User not local; please try <White@Gamma.ARPA>
```

Для открытия простейшей SMTP-транзакции отправки и получения почтового сообщения необходимо, чтобы между SMTP-отправителем и SMTP-получателем были установлены:

1. Транспортное соединение, например, TCP или UDP, или X.25.
2. SMTP-соединение.

SMTP-соединение устанавливается между отправителем и SMTP-получателем по готовому транспортному каналу. Этот процесс включает в себя обмен сообщениями открытия SMTP-канала.

```
HELO <SP> <domain> <CRLF>
```

Команда `HELO` отправляется инициирующим открытие соединения хостом. Как правило, это хост отправителя, который идентифицирует себя через параметр `<domain>`. Ответом является код `250` с именем отвечающего домена. Например, после того, как установлено транспортное соединение с хостом SMTP-получателя, происходит следующее:

```
R: 220 OH-UNIX.ARPA Simple Mail Transfer Service Ready
S: HELO AH-XENIX.ARPA
R: 250 OH-UNIX.ARPA
```

Первая строчка приходит на хост SMTP-отправителя (он же инициатор соединения) и сообщает о готовности протокола SMTP к работе.

Во второй строчке отправитель передает команду установления SMTP-канала и открывает транзакцию.

В третьей строчке SMTP-получатель сообщает об успешном открытии транзакции.

Для того чтобы завершить сеанс протокола SMTP, достаточно отправить команду `QUIT`. Например:

```
S: QUIT
R: 221 AH-UNIX.ARPA Service closing transmission channel
```

Для того чтобы прервать текущую SMTP-транзакцию и очистить все буферы отправителя и получателя данной транзакции (т. е. перезапустить SMTP-

соединение), используется команда RSET. Она может применяться, например, при зацикливаниях квитирующих сообщений доставки.

Кроме описанных выше команд, протокол SMTP поддерживает и другие команды, которые могут быть полезны при передаче почты, но не входят в минимальный список обязательных команд и могут не выполняться на всех SMTP-серверах:

- Команда VRFY. Аргументом этой команды является имя почтового ящика или пользователя. Команда проверяет наличие указанного пользователя или почтового ящика в текущем домене и либо возвращает сообщение об отсутствии информации, либо сообщает полные атрибуты почтового ящика, либо переадресовывает поиск в другой домен. Ниже представлены несколько вариантов выполнения команды VRFY:

```
S: VRFY Smith
R: 250 Fred Smith <Smith@USC-ISIF.ARPA>

S: VRFY Smith
R: 251 User not local; will forward to <Smith@USC-ISIQ.ARPA>

S: VRFY Jones
R: 550 String does not match anything.

S: VRFY Jones
R: 551 User not local; please try <Jones@USC-ISIQ.ARPA>

S: VRFY Gourjygiolz
R: 553 User ambiguous.
```

- Команда EXPN. Аргументом этой команды является строка-идентификатор списка адресатов домена. Это может быть имя файла, содержащее список адресатов, или какой-либо другой идентификатор списка, определенный в системе. По этой команде SMTP-получатель отправляет список адресатов домена или сообщение о невозможности по каким-либо причинам выполнить данную команду (может быть, из-за отсутствия достаточных прав). Например:

```
S: EXPN Example-People
R: 250-Fred Fonebone <Fonebone@USC-ISIQ.ARPA>
R: 250-Sam Q. Smith <SQSmith@USC-ISIQ.ARPA>
R: 250-Quincy Smith <@USC-ISIF.ARPA:Q-Smith@ISI-VAXA.ARPA>
R: 250-<joe@foo-unix.ARPA>

S: EXPN Executive-Washroom-List
R: 550 Access Denied to You.
```

Примечание

Команды VRFY и EXPN не входят в минимальный список обязательных команд и могут не выполняться на SMTP-серверах.

Протокол SMTP может доставлять сообщения не только в почтовые ящики пользователя, но и позволяет обмениваться сообщениями между терминалами пользователей (т. е. в момент передачи сообщения на принимающем хосте получатель является *активным* пользователем). Для этих целей используются команды SEND, SOML и SAML — эти команды также не входят в минимальный список обязательных команд и могут не выполняться на SMTP-серверах. Все эти команды имеют ту же структуру, что и команда MAIL. Отличия заключаются в том, что:

- команда SEND отправляет сообщение только на терминал получателя;
- команда SOML (Send Or MaiL) отправляет сообщение на терминал получателя, если он активен, или оставляет сообщение в почтовом ящике получателя, если терминал получателя не активен;
- команда SAML (Send And MaiL) отправляет сообщение как на терминал (если он активен), так и в почтовый ящик получателя.

Поскольку речь зашла о работе в терминальном режиме, необходимо упомянуть еще о двух командах, которые могут быть очень полезны при такой работе.

- Команда TURN может быть использована для изменения функций приложений протокола SMTP (отправитель — на получатель, получатель — на отправитель). Так, если SMTP-отправитель (A) отправляет команду TURN и получатель (B) отвечает 250 OK, то A становится SMTP-получателем, а B — SMTP-отправителем.
- Команда HELP. Эта команда особенно полезна при работе с SMTP-сервером в терминальном режиме. Она возвращает SMTP-отправителю информацию о поддерживаемых командах сервером протокола SMTP. Если в качестве аргумента будет передано имя какой-либо команды, ответом будет подробное описание предназначения и синтаксиса интересующей команды.

```
HELP [<SP> <string>] <CRLF>
```

Ретрансляция сообщений

В огромной сети Интернет очень редко удается отправить сообщение адресату напрямую. Как правило, используются SMTP-серверы, которые выполняют роль промежуточных пунктов пересылки сообщений. SMTP-серверы принимают всю поступившую на них почту (если администратором SMTP-сервера не установлены ограничения) и затем самостоятельно переправляют ее адресату. Этот процесс и называется ретрансляцией сообщений.

SMTP-серверы выбирают путь сообщения по своему усмотрению в зависимости от параметров настройки, скорости доступа и т. д. Протокол SMTP позволяет отправителю сообщения самостоятельно указывать путь передачи сообщения, устанавливая в качестве параметров команды отправки промежуточные SMTP-серверы.

Параметр `<forward-path>` команды `RCPT` может содержать не только имя почтового ящика получателя, но и путь к этому почтовому ящику через SMTP-серверы, например:

```
S: RCPT TO:<@ONE,@TWO:Jones@THREE>
```

где **ONE**, **TWO**, **THREE** — имена хостов (**ONE** и **TWO** — имена промежуточных хостов, **Jones@THREE** — имя почтового ящика адресата на хосте **THREE**). Механизм ретрансляции сообщений состоит в следующем:

1. Сообщение передается SMTP-серверу, стоящему первым на пути передачи (хост **ONE**). После принятия этим сервером сообщения он перемещает свой идентификатор из пути передачи и ставит его в строку обратного пути `<reverse-path>` (определенный в команде `MAIL`).
2. Затем сервер, получивший данное сообщение, становится SMTP-отправителем и передает сообщение по адресу хоста, стоящего первым в пути передачи (это будет теперь хост **TWO**). Теперь это сообщение с измененными параметрами `<forward-path>` и `<reverse-path>` отправляется на следующий по маршруту хост.
3. Далее все повторяется, пока сообщение не дойдет до адресата (т. е. параметр `<forward-path>` станет равным пустой строке).

Примечание

Заметим, что идентификаторы отправителя `<reverse-path>` и получателя `<forward-path>` никак не отражаются на параметрах сообщения `To`, `From` и `Cc`.

Если SMTP-сервер получил сообщение, но позднее (например, при его отправке) обнаружил, что путь передачи указан неверно (или произошли какие-либо сбои сети и т. д.) и данное сообщение не может быть отправлено, он обязан построить "сообщение уведомления" об ошибке и отправить его отправителю сообщения (указанному в параметре `<reverse-path>`). Для предотвращения зацикливания уведомляющих сообщений сообщение об ошибочном сообщении передается SMTP-сервером без имени (идентификатора) отправителя, т. е. без параметра `<reverse-path>`. Вместо него передается пустая строка:

```
MAIL FROM:<>
```

Информация об отправителе этого сообщения помещается в поле `From` конверта сообщения.

Рассмотрим теперь несколько примеров выполнения транзакций SMTP.

Примеры выполнения транзакций протокола SMTP

Простейший пример SMTP-транзакции — передача сообщения трем адресатам.

```
R: 220 BBN-UNIX.ARPA Simple Mail Transfer Service Ready
S: HELO USC-ISIF.ARPA
R: 250 BBN-UNIX.ARPA
S: MAIL FROM:<Smith@USC-ISIF.ARPA>
R: 250 OK
S: RCPT TO:<Jones@BBN-UNIX.ARPA>
R: 250 OK
S: RCPT TO:<Green@BBN-UNIX.ARPA>
R: 550 No such user here
S: RCPT TO:<Brown@BBN-UNIX.ARPA>
R: 250 OK
S: DATA
R: 354 Start mail input; end with <CRLF>.<CRLF>
S: Hey!...
S:... Bye!
S:.
R: 250 OK
S: QUIT
R: 221 BBN-UNIX.ARPA Service closing transmission channel
```

Адресату **Green@BBN-UNIX.ARPA** почта не была доставлена.

Пример прерванной SMTP-транзакции выглядит так:

```
R: 220 MIT-Multics.ARPA Simple Mail Transfer Service Ready
S: HELO ISI-VAXA.ARPA
R: 250 MIT-Multics.ARPA
S: MAIL FROM:<Smith@ISI-VAXA.ARPA>
R: 250 OK
S: RCPT TO:<Jones@MIT-Multics.ARPA>
R: 250 OK
S: RCPT TO:<Green@MIT-Multics.ARPA>
R: 550 No such user here
S: RSET
R: 250 OK
```

```
S: QUIT
R: 221 MIT-Multics.ARPA Service closing transmission channel
```

В этом случае почта адресатам не передана.

Пример ретрансляции сообщения от **JQP@MIT-AI.ARPA** на **Jones@BBN-VAX.ARPA** через **USC-ISIE.ARPA** выглядит следующим образом:

- Шаг 1: обработка сообщения на хосте **USC-ISIE.ARPA**.

```
R: 220 USC-ISIE.ARPA Simple Mail Transfer Service Ready
S: HELO MIT-AI.ARPA
R: 250 USC-ISIE.ARPA
S: MAIL FROM:<JQP@MIT-AI.ARPA>
R: 250 OK
S: RCPT TO:<@USC-ISIE.ARPA:Jones@BBN-VAX.ARPA>
R: 250 OK
S: DATA
R: 354 Start mail input; end with <CRLF>.<CRLF>
S: Date: 2 Nov 97 22:33:44
S: From: John Q. Public <JQP@MIT-AI.ARPA>
S: Subject:  Hello
S: To: Jones@BBN-Vax.ARPA
S:
S: Hello! I am John.
S:.
R: 250 OK
S: QUIT
R: 221 USC-ISIE.ARPA Service closing transmission channel
```

- Шаг 2: доставка адресату **Jones@BBN-VAX.ARPA**.

```
R: 220 BBN-VAX.ARPA Simple Mail Transfer Service Ready
S: HELO USC-ISIE.ARPA
R: 250 BBN-VAX.ARPA
S: MAIL FROM:<@USC-ISIE.ARPA:JQP@MIT-AI.ARPA>
R: 250 OK
S: RCPT TO:<Jones@BBN-VAX.ARPA>
R: 250 OK
S: DATA
R: 354 Start mail input; end with <CRLF>.<CRLF>
S: Received: from MIT-AI.ARPA by USC-ISIE.ARPA ;
   2 Nov 97 22:40:10 UT
S: Date: 2 Nov 97 22:33:44
```

```
S: From: John Q. Public <JQP@MIT-AI.ARPA>
S: Subject:  Hello
S: To: Jones@BBN-Vax.ARPA
S:
S: Hello! I am John.
S:.
R: 250 OK
S: QUIT
R: 221 USC-ISIE.ARPA Service closing transmission channel
```

Использование протокола SMTP для передачи 8-битных данных

В предыдущих разделах мы рассматривали использование протокола SMTP для передачи 7-битных сообщений. Такие сообщения поддерживаются всеми SMTP-серверами. Поэтому, если информация, которую необходимо передать, содержит не только ASCII-символы, а отправитель не знает, какая почтовая система используется у его адресата, он, как правило, кодирует всю передаваемую информацию в 7-битный вид. Например, если и отправитель, и получатель поддерживают стандарт MIME, объекты сообщения можно представить в кодировках "quoted-printable" или "base64".

Однако если SMTP-сервер адресата поддерживает расширения протокола SMTP (ESMTP) для передачи 8-битных данных в формате "MIME 8bit", отправитель может не беспокоиться о кодировании и передавать по SMTP все 8 бит.

Для того чтобы проверить, поддерживает ли SMTP-сервер расширения "MIME 8bit", отправьте команду приветствия `EHLO` — это прототип команды `HELO` для протокола ESMTP. Ответом на это приветствие должны быть строчки `250 OK` и `250 8BITMIME`. Например: (символы "`C:`" и "`S:`" в данном случае используются для обозначения SMTP-клиента и сервера соответственно)

```
C: EHLO <client.domain>
S: 250-<server.domain> says hello
S: 250 8BITMIME
```

Примечание

В протоколе ESMTP команда `MAIL` имеет дополнительный аргумент: `BODY`, который может принимать значения "7BIT" или "8BITMIME".

Например, команда `MAIL` может быть сформирована с аргументом "BODY=8BITMIME". Если на эту команду сервер ответит кодом "250", это

будет означать, что он готов к приему 8-битных данных. Ниже представлен пример сценария передачи 8-битных данных по протоколу ESMTP:

```
C: EHLO ymir.claremont.edu
S: 250-dbc.mtview.ca.us says hello
S: 250 8BITMIME
C: MAIL FROM:<ned@ymir.claremont.edu> BODY=8BITMIME
S: 250 <ned@ymir.claremont.edu>... Sender and 8BITMIME ok
C: RCPT TO:<mrose@dbc.mtview.ca.us>
S: 250 <mrose@dbc.mtview.ca.us>... Recipient ok
C: DATA
S: 354 Send 8BITMIME message, ending in CRLF.CRLF.
...
C:.
S: 250 OK
```

Однако при выборе такого способа передачи необходимо помнить, что остаются в силе ограничения на размер строки SMTP-сообщения, которая составляет 1000 байтов.

Для преодоления ограничения на размер строки и передачи по SMTP типа "binary" рисунков, видео и др. больших массивов бинарных данных может быть использовано расширение протокола SMTP, часто называемое как CHUNKING.

CHUNKING-сервис использует вместо старой команды `DATA` расширенную команду — `BDAT`. Данное расширение поддерживает как типы "7bit", "8BITMIME" так и "BINARYMIME", который и позволяет передавать большие объемы двоичных данных.

Для того чтобы узнать, поддерживает ли данный SMTP-сервер сервис CHUNKING, используется команда приветствия `EHLO`. Если ответ содержит строку `250 CHUNKING`, это значит, что SMTP-сервер может работать с командой `BDAT`. Команда `BDAT` — это простое расширение функциональных возможностей команды `DATA`, т. е. вы можете без каких-либо ограничений при отправке 7-битных данных вместо команды `DATA` использовать команду `BDAT`.

Примечание

Нельзя использовать команды `DATA` и `BDAT` в одной SMTP-транзакции.

Команда `BDAT` имеет следующий формат:

```
BDAT chunk-size [LAST]
```

где `<chunk-size>` — размер порции передаваемых данных, а ключевое слово `LAST` указывает, что следующая порция информации — последняя.

Для того чтобы отправить данные в формате "BINARYMIME", команда отправки — MAIL должна содержать параметр BODY=BINARYMIME. Затем, используя команду BDAT, передавайте вашу информацию. Ниже представлен пример передачи большого бинарного файла (R: — получатель, S: — отправитель SMTP-сообщения).

```
R: <wait for connection on TCP port 25>
S: <open connection to server>
R: 220 server.esmtp.us SMTP service ready
S: EHLO client.edu
R: 250-server.esmtp.us says hello
R: 250-PIPELINING
R: 250-BINARYMIME
R: 250 CHUNKING
S: MAIL FROM:<one@client.edu> BODY=BINARYMIME
S: RCPT TO:<two@server.esmtp.us>
R: 250 <one@client.edu>... Sender and BINARYMIME ok
R: 250 <two@server.esmtp.us>... Recipient ok
S: BDAT 100000
S: (First 10000 octets of canonical MIME message data)
S: BDAT 324 LAST
S: (Remaining 324 octets of canonical MIME message data)
R: 250 100000 bytes received
R: 250 Message OK, 100324 octets received
S: QUIT
R: 221 Goodbye
```

Взаимодействие SMTP с протоколом TCP

SMTP — это протокол прикладного уровня. Он использует протокол TCP как транспортный протокол, обеспечивающий достоверность и надежность канала соединения. Как правило, SMTP-канал использует TCP-соединение между SMTP-отправителем — машиной пользователя, на которой порт SMTP-соединения может быть выбран произвольно, и SMTP-сервером, который, по умолчанию, использует порт 25.

TCP-соединение поддерживает передачу 8-битных символов. Если протокол SMTP использует для передачи 7-битные ASCII-символы, каждый из них передается по TCP-каналу как 8 бит со старшим битом, равным 0. Передаются 8-битные SMTP-данные без изменений.

SMTP-команды и данные инкапсулируются в поле данных протокола TCP в соответствии с обычными механизмами инкапсуляции стека протоколов TCP/IP.

Протокол POP3

Как правило, небольшим организациям невыгодно держать у себя систему передачи сообщений (message transport system).

Для решения этой проблемы был разработан почтовый протокол для работы в офисе — POP (Post Office Protocol). Его наиболее распространенный вариант — POP3 (Протокол почтового отделения версия 3). Этот протокол позволяет рабочим станциям динамически получать доступ к своим почтовым ящикам, расположенным на сервере, предназначенном для обслуживания электронной почты в данной организации.

Протокол POP3 — это простейший протокол для работы пользователя с содержимым своего почтового ящика. Он позволяет только забрать почту из почтового ящика сервера на рабочую станцию клиента и удалить ее из почтового ящика на сервере. Всю дальнейшую обработку почтовое сообщение проходит на компьютере клиента.

POP3-сервер не отвечает за отправку почты и используется только как универсальный почтовый ящик для группы пользователей. Когда пользователю необходимо отправить сообщение, он должен установить соединение с каким-либо SMTP-сервером и отправить туда свое сообщение по протоколу SMTP. Этот SMTP-сервер может быть тем же хостом, где работает POP3-сервер, а может располагаться совсем в другом месте (в другом домене или, вообще говоря, где угодно в сети Интернет).

Как правило, при обмене электронной почтой небольшие организации могут использовать для получения своей корреспонденции POP3-сервер, установленный на каком-либо компьютере в офисе, а для отправки почты по протоколу SMTP — один из хорошо доступных SMTP-серверов.

Основные принципы протокола POP3

Приложение, выполняющее функцию POP3-сервера, как правило, использует 110-й TCP-порт, который находится в режиме ожидания входящего соединения. Когда клиент хочет воспользоваться POP3-сервисом, он просто устанавливает TCP-соединение с портом 110 этого хоста. После установления соединения сервер POP3 отправляет подсоединившемуся клиенту приветственное сообщение. После этого клиент и сервер начинают обмен командами и данными. По окончании обмена POP3-канал закрывается.

Команды POP3 состоят из ключевых слов, состоящих из ASCII-символов, и одним или несколькими параметрами, отделяемыми друг от друга символом "пробела" — <SP>. Все команды заканчиваются символами "возврата каретки" и "перевода строки" — <CRLF>. Длина ключевых слов не превышает четырех символов, а каждого из аргументов может быть до 40 символов.

Ответы POP3-сервера на команды состоят из строки статус-индикатора, ключевого слова, строки дополнительной информации и символов завершения строки — <CRLF>. Длина строки ответа может достигать 512 символов. Строка статус-индикатора принимает два значения: положительное (+OK) и отрицательное (–ERR). Любой сервер POP3 обязан отправлять текст строки статус-индикатора в верхнем регистре, тогда как другие команды и данные могут приниматься или отправляться как в нижнем, так и в верхнем регистрах.

Ответы POP3-сервера на некоторые команды могут составлять несколько строк. В этом случае строки разделены символами <CRLF>. Последнюю строку информационной группы завершает строка, состоящая из символа "." (код — 046) и <CRLF>, т. е. последовательность <CRLF>.<CRLF>.

POP3-сессия состоит из нескольких частей. Как только открывается TCP-соединение и POP3-сервер отправляет приветствие, сессия должна быть зарегистрирована — состояние авторизации клиента (AUTHORIZATION state). Процесс авторизации включает определение подлинности абонента и установление соответствия его прав доступа запрошенным сервисам. Для выполнения этой стадии клиент должен зарегистрироваться в POP3-сервере, т. е. ввести свой идентификатор и пароль.

После этого сервер предоставляет клиенту его почтовый ящик и открывает для данного клиента транзакцию — состояние начала транзакции обмена (TRANSACTION state). На этой стадии клиент может считать и удалить почту своего почтового ящика.

После того как клиент заканчивает работу (передает команду QUIT), сессия переходит в состояние UPDATE — завершение транзакции. В этом состоянии POP3-сервер закрывает транзакцию данного клиента (на языке баз данных — операция COMMIT) и закрывает TCP-соединение.

В случае получения неизвестной, неиспользуемой или неправильной команды POP3-сервер должен ответить отрицательным состоянием индикатора.

POP3-сервер может использовать в своей работе таймер контроля времени соединения. Этот таймер отсчитывает время "бездействия" ("idle") клиента в сессии от последней переданной команды. Если время сессии истекло, сервер закрывает TCP-соединение, не переходя в состояние UPDATE (иными словами, откатывает транзакцию или, на языке баз данных, — выполняет ROLLBACK).

Примечание

Не забывайте, что POP3-сервер может обслуживать группу клиентов, которые, возможно, присоединяются по коммутируемой линии, и, следовательно, необходимо иметь средство автоматического регулирования времени соединения. По спецификации POP3-таймер контроля состояния "idle" должен быть установлен на промежуток времени не менее 10 минут.

Организация информационного обмена

При открытии TCP-соединения POP3-клиентом, POP3-сервер отправляет сообщение приветствия. Далее во всех примерах POP3-протокола используются следующие обозначения: `C:` —клиент, `S:` — сервер POP3:

```
S:  +OK POP3 server ready
```

Теперь POP3-сессия находится в состоянии авторизации (AUTHORIZATION), и клиент должен зарегистрировать себя на POP3-сервере. Это может быть выполнено либо с помощью команд `USER` и `PASS` — ввод открытых пользовательского идентификатора и пароля (именно этот способ используется чаще), либо командой `APOP` — авторизация цифровой подписью, на базе секретного ключа. Любой POP3-сервер должен поддерживать хотя бы один из механизмов аутентификации.

- Команда `USER` имеет следующий формат:

  ```
  USER name
  ```

  Аргументом `name` является строка, идентифицирующая почтовый ящик системы. Этот идентификатор должен быть уникальным в почтовой системе POP3-сервера. Если ответом на эту команду является строка индикатора `+OK`, клиент может отправлять команду `PASS` — ввод пароля или `QUIT` — завершить сессию. Если ответом является строка `-ERR`, клиент может либо повторить команду `USER`, либо закрыть сессию. Примеры использования команды:

  ```
  C: USER frated
  S: -ERR sorry, no mailbox for frated here
  ```

  или

  ```
  C: USER mrose
  S: +OK mrose is a real hoopy frood
  ```

  Примечание

  Сервер может вернуть отрицательный ответ, если почтовый ящик существует, но по каким-либо причинам не доступен.

- Команда `PASS` используется только после положительного ответа на команду `USER`:

  ```
  PASS string
  ```

  Аргументом команды является строка пароля данного почтового ящика. После получения команды `PASS` POP3-сервер, на основании аргументов команд `USER` и `PASS`, определяет возможность доступа к заданному почтовому ящику. Если POP3-сервер ответил `+OK`, это означает, что аутентификация клиента прошла успешно и он может работать со своим почтовым ящиком, т. е. сессия переходит в состояние TRANSACTION. Если POP3-

сервер ответил –ERR, то либо был введен неверный пароль, либо не найден указанный почтовый ящик:

```
C: USER mrose
S: +OK mrose is a real hoopy frood
C: PASS secret
S: -ERR maildrop already locked
```

или

```
C: USER mrose
S: +OK mrose is a real hoopy frood
C: PASS secret
S: +OK mrose's maildrop has 2 messages (320 octets)
```

- Команда аутентификации пользователя APOP не входит в список обязательно реализуемых команд POP3-сервера. Эта команда предоставляет значительно больший (по сравнению с командами USER или PASS) уровень защиты авторизации пользователя при открытии сессии AUTHORIZATION и используется только тогда, когда к обеспечению конфиденциальности доступа к информации почтовых ящиков предъявляются повышенные требования. Эта команда может быть передана клиентом POP3-сервера после получения приветственного сообщения или после ошибки обработки команд USER/PASS.

```
APOP name digest
```

Аргументами команды являются: name — имя пользователя (то же, что и в команде USER), digest — шифрованная (по алгоритму MD5) строка пароля. Применяемый здесь алгоритм необратимого шифрования для построения секретного ключа использует открытый ключ и временную метку. Временные метки передаются хосту клиента вместе с сообщением приветствия. Например, для UNIX-машин временная метка может иметь вид: <process-ID.clock@hostname>, где process-ID — это идентификатор процесса, clock — состояние таймера на момент установления соединения, hostname — имя компьютера POP3-сервера. Этот механизм позволяет достичь очень высокой степени защищенности. Далее показан пример работы команды APOP.

```
S: +OK POP3 server ready <1896.697170952@dbc.mtview.ca.us>
C: APOP mrose c4c9334bac560ecc979e58001b3e22fb
S: +OK maildrop has 1 message (369 octets)
```

Алгоритм на основании открытого ключа "tanstaaf" и временной метки <1896.697170952@dbc.mtview.ca.us> построил шифрованную строку c4c9334bac560ecc979e58001b3e22fb.

К командам состояния AUTHORIZATION может относиться команда закрытия POP3-сессии — QUIT, если она была отправлена в режиме

AUTHORIZATION (например, при вводе неправильного пароля или идентификатора пользователя).

Эта команда отправляется без аргументов и всегда имеет единственный ответ `+OK`, например:

```
C: QUIT
S: +OK dewey POP3 server signing off
```

После того как клиент успешно прошел процедуру аутентификации в POP3-сервере и POP3-сервер "закрыл" определенный почтовый ящик только для использования данным клиентом (для тех, кто работал с базами данных, это называется EXCLUSIVE ACCESS LOCK), POP3-сессия переходит в режим TRANSACTION, и клиент может начать работу со своей почтой.

- Команда `STAT` (без аргументов) используется для просмотра состояния текущего почтового ящика.

  В ответ POP3-сервер возвращает строку, содержащую количество и общий размер в байтах сообщений, которые клиент может получить с POP3-сервера. Сообщения, помеченные на удаление, не учитываются. Формат ответа: `+OK nn mm`, где `nn` — количество сообщений, `mm` — их общий объем:

  ```
  C: STAT
  S: +OK 2 320
  ```

  В этом примере POP3-сервер сообщает, что в данном почтовом ящике находятся два сообщения общим объемом 320 байт.

  После того как POP3-сервер открыл почтовый ящик, он присваивает каждому сообщению номер и устанавливает его размер в байтах. Первому сообщению присваивается число 1, второму — 2 и т. д. Далее во всех командах, относящихся к сообщениям, POP3-сервер ссылается на сообщения по их номерам и указывает их размер только в десятичном виде.

- Команда `LIST` может передаваться как с аргументом `msg` — номером сообщения, так и без аргумента:

  ```
  LIST [msg]
  ```

  Если команда содержит аргумент, и сообщение с указанным номером существует, ответом на нее будет "информационная строка", которая содержит номер сообщения и размер сообщения в байтах. Если аргумент не указан — ответом будет список информационных строк обо всех сообщениях в данном почтовом ящике. Сообщения, помеченные на удаление, не фигурируют в этом списке:

  ```
  C: LIST
  S: +OK 2 messages (320 octets)
  ```

```
S: 1 120
S: 2 200
S:.
```

или

```
C: LIST 2
S: +OK 2 200
```

или

```
C: LIST 3
S: -ERR no such message, only 2 messages in maildrop
```

- Следующая команда — команда `RETR` — используется для передачи клиенту запрашиваемого сообщения:

```
RETR msg
```

Аргумент команды `msg` — номер сообщения. Если запрашиваемого сообщения нет, возвращается отрицательный индикатор `-ERR`.

```
C: RETR 1
S: +OK 120 octets
S: <text message>
S:.
```

- После получения сообщение, как правило, помечается на удаление из почтового ящика, при этом используется команда `DELE`:

```
DELE msg
```

Аргумент команды— номер сообщения. Сообщения, помеченные на удаление, реально удаляются только после закрытия транзакции, на стадии UPDATE.

```
C: DELE 1
S: +OK message 1 deleted
```

или

```
C: DELE 2
S: -ERR message 2 already deleted
```

- Для проверки состояния соединения с POP3-сервером используется команда `NOOP`. При активном соединении ответом на нее будет положительный индикатор `+OK`:

```
C: NOOP
S: +OK
```

- Для отката транзакции внутри сессии используется команда `RSET` (без аргументов). Если пользователь случайно пометил на удаление какие-либо сообщения, он может убрать эти пометки, отправив эту команду:

```
C: RSET
S: +OK maildrop has 2 messages (320 octets)
```

После того как пользователь проделал в режиме TRANSACTION все необходимые действия, он должен перейти в режим UPDATE. Для этого клиент POP3 отправляет команду `QUIT`. По этой команде все сообщения, помеченные на удаление, реально уничтожаются в почтовом ящике. Если сессия TRANSACTION завершается каким-либо другим способом (сбой сервера, сети или др.) никаких изменений в состоянии почтового ящика не происходит.

- POP3-сервер может поддерживать еще несколько команд, которые предоставляют пользователю большую свободу в манипулировании сообщениями, но которые не входят в список обязательных для реализации команд, т. е. ваш POP3-сервер может их и не поддерживать. Одной из таких команд является команда `TOP`:

  ```
  TOP msg n
  ```

  По этой команде пользователь может получить `n` первых строк сообщения с номером `msg`. POP3-сервер по запросу клиента отправляет заголовок сообщения, затем пустую строку, затем требуемое количество строк сообщения (если количество строк в сообщении меньше указанного в параметре `n`, пользователю передается все сообщение).

  ```
  C: TOP 1 10
  S: +OK
  S: <header>
  S: <blank>
  S: <message body>
  S:.
  ```

Различные расширения POP3-протокола могут включать в себя и другие команды или поддерживать расширения описанных команд.

Пример использования протокола POP3

Ниже приведен пример использования протокола POP3 для приема сообщений из почтового ящика mrose.

```
S: <wait for connection on TCP port 110>
C: <open connection>
S:    +OK POP3 server ready
C:    USER mrose
S:    +OK mrose is a real hoopy frood
C:    PASS secret
S:    +OK mrose's maildrop has 2 messages (320 octets)
C:    STAT
```

```
S:    +OK 2 320
C:    LIST
S:    +OK 2 messages (320 octets)
S:    1 120
S:    2 200
S:    .
C:    RETR 1
S:    +OK 120 octets
S:    <the POP3 server sends message 1>
S:    .
C:    DELE 1
S:    +OK message 1 deleted
C:    RETR 2
S:    +OK 200 octets
S:    <the POP3 server sends message 2>
S:    .
C:    DELE 2
S:    +OK message 2 deleted
C:    QUIT
S:    +OK dewey POP3 server signing off (maildrop empty)
C:  <close connection>
S:  <wait for next connection>
```

Простота протокола POP, которая послужила росту его популярности вначале, обернулась затем отсутствием гибкости и невозможностью выполнять другие операции управления почтовыми ящиками. На смену POP3 пришло новое поколение протоколов работы с электронной почтой — протоколы IMAP.

Протокол IMAP4

В этом разделе будет рассмотрена наиболее популярная версия протокола серии IMAP (Internet Message Access Protocol, Протокол доступа к электронной почте Internet), а именно IMAP4 (IMAP version 4).

Протокол IMAP4 позволяет клиентам получать доступ и манипулировать сообщениями электронной почты на сервере.

Существенным отличием протокола IMAP4 от протокола POP3 является то, что протокол IMAP4 поддерживает работу с системой каталогов (или папок) сообщений. Этот протокол позволяет пользователю управлять каталогами (папками) удаленных сообщений так же просто, как если бы они располагались на его собственном компьютере. Протокол IMAP4 позволяет клиенту

создавать, удалять и переименовывать почтовые ящики, проверять наличие новых сообщений и удалять старые. Благодаря тому, что протокол IMAP4 поддерживает механизм уникальной идентификации каждого сообщения в почтовой папке клиента, он позволяет читать из почтового ящика только сообщения, удовлетворяющие определенным условиям или их части, менять атрибуты сообщений и перемещать отдельные сообщения.

Структура папок в значительной степени зависит от типа почтовой системы, но в любой системе у каждого клиента есть специальный каталог `INBOX`, куда попадают поступающие ему сообщения.

Основные принципы протокола IMAP4

Протокол IMAP4 работает поверх транспортного протокола, который обеспечивает надежный и достоверный канал передачи данных между клиентом и IMAP4-сервером. При работе по протоколу TCP, протокол IMAP4 использует 143-й порт. Команды и данные протокола IMAP4 передаются по транспортному протоколу в том виде, в каком их отправляет сервер или пользователь.

Принцип передачи данных по протоколу IMAP4 такой же, как и у других подобных протоколов. Сначала клиент и сервер обмениваются приветствиями. Затем клиент отправляет на сервер команды и данные. Сервер, соответственно, передает клиенту ответы на обработку команд и данных. После завершения обмена канал закрывается.

Если сервер использует таймер контроля времени соединения, он должен быть установлен не менее чем на 30-минутный интервал "пассивности" клиента. Таким образом, если сервер в течение 30 минут получает хотя бы одну команду, таймер сбрасывается.

Весь обмен данными между клиентом и сервером организован в виде строк, завершающихся символами возврата каретки и перевода строки, либо в виде последовательности байтов заданной длины. Каждая команда клиента начинается с идентификатора или тега команды. Тег, как правило, представляет собой короткую строку, состоящую из букв и цифр, (например, `A0001`, `A0002` и т. д.). Тег является уникальным идентификатором данной команды клиента. Ответы сервера или следующие команды клиента могут ссылаться на данную команду по ее тегу.

Каждая команда клиента начинается с новой строки. В тех случаях, когда команда передает поток данных заданной длины или когда команда требует ответа с сервера, для того чтобы продолжить работу (например, при аутентификации), она может занимать несколько строк.

Строки данных, передаваемые с сервера в ответ на команду клиента, могут не содержать тег, а содержать символ "`*`". Это означает, что они являются промежуточными строками потока данных ответа, а идентификатор их

команды содержится в последней строке потока. В такой поток данных не может вклиниться другая команда.

Если сервер обнаружил ошибку в команде, он отправляет уведомление `BAD` клиенту с тегом неправильной команды. Если команда успешно обработана — возвращается уведомление `OK` с тегом команды. Если команда вернула отрицательный результат, например, в случае невозможности выполнить данную команду — возвращается уведомление `NO` с тегом невыполненной команды.

Важной особенностью протокола IMAP является то, что взаимодействие клиента с сервером не строится по принципу "запрос-ответ", в котором каждая из сторон "ходит" по очереди. Клиент может отправить новую команду на сервер, не дожидаясь ответа на предыдущую команду. Естественно, это правило не распространяется на ситуацию, когда эти команды взаимосвязаны или когда ответ одной влияет на результат другой. Сервер может обрабатывать несколько команд одновременно и отвечать на каждую из них по ее окончании. При этом ответ на более позднюю команду может поступить раньше, поэтому ответ сервера всегда содержит тег той команды, к которой он относится.

Для работы в таком режиме клиент и сервер должны фиксировать весь поток данных обмена, поскольку как сервер, так и клиент в своих запросах и ответах могут ссылаться на команды и данные, введенные на предыдущих стадиях сессии обмена.

Для того чтобы обеспечить гибкость и многофункциональность операций работы с сообщениями, почтовые системы протокола IMAP присваивают сообщениям определенные атрибуты.

Атрибуты сообщений IMAP-системы

Каждое сообщение в почтовой системе для работы с протоколом IMAP имеет уникальный идентификатор, по которому можно получить доступ к этому сообщению. Уникальный идентификатор UID (Unical IDentifier) представляет собой 32-битное число, которое идентифицирует сообщение в данной папке. Уникальные идентификаторы сообщений сохраняются от сессии к сессии и могут использоваться, например, для синхронизации каталогов мобильных пользователей.

Каждая папка в системе также имеет уникальный действительный идентификатор (UIDVALIDITY). Вместе с идентификатором UID сообщения эта пара образует 64-битное число, однозначно идентифицирующее каждое сообщение. Если идентификатор UID сообщения сохраняется постоянным, то UIDVALIDITY данной папки в текущей сессии должен быть больше, чем в предыдущей сессии.

Кроме уникального идентификатора, сообщение в IMAP-системе имеет порядковый номер, т. е. все сообщения в данном почтовом ящике последовательно нумеруются. Если в почтовый ящик добавляется новое сообщение, ему присваивается номер на 1 больше количества сообщений в почтовом ящике. При удалении какого-либо сообщения из данной папки порядковые номера всех сообщений пересчитываются, поэтому порядковый номер сообщения может меняться во время сессии. Большинство команд протокола IMAP4 работают с порядковыми номерами сообщений, а не с идентификатором UID.

Помимо числовых идентификаторов, сообщениям назначаются флаги. Одни флаги могут быть действительны для данного сообщения постоянно от сессии к сессии, другие — только в данной сессии. К наиболее употребительным флагам, в частности, относятся:

- `\Seen` — обозначает, что данное сообщение было прочитано;
- `\Answered` — на сообщение был дан ответ;
- `\Deleted` — сообщение помечено на удаление;
- `\Draft` — формирование данного сообщения еще не завершено;
- `\Recent` — сообщение "только что" поступило в почтовый ящик, т. е. данная сессия — первая, которая может прочитать это сообщение. `\Recent` — пример флага, который не сохранится в следующей сессии.

Кроме того, на сервере протокола IMAP хранятся дата и время получения сообщения сервером. Например, если сообщение получено с использованием протокола SMTP, то фиксируется дата и время доставки по адресу назначения, общий размер сообщения, состав конверта (заголовка) сообщения (не путать с SMTP-конвертом), структура сообщения (MIME-структура).

Основные команды протокола IMAP4

Как уже было отмечено, IMAP4 — гибкий и многофункциональный протокол с широкими возможностями. Он обслуживает более 20 различных команд клиента по управлению состоянием своей почты. Подробное описание всех команд и ответов IMAP4-сервера вы можете найти, например, в документе RFC-2060. Далее будут описаны только некоторые из клиентских команд и на примерах их обработки показана общая схема организации взаимодействия клиента и IMAP4-сервера.

Протокол IMAP4 поддерживает текстовые команды и ответы сервера, т. е. ASCII-последовательности символов. Строка команды или данных завершается последовательностью символов возврата каретки и перевода строки. 8-битные данные, в соответствии со спецификацией MIME, не могут переда-

ваться по протоколу IMAP4 в "открытом" виде. Как правило, реализации протокола IMAP перед передачей двоичных данных кодируют их кодировкой "base64".

Так же, как и POP3-сервер, IMAP4-сервер выполняет команды в зависимости от одного из четырех состояний, в котором он находится:

1. В состоянии сервера вне аутентификации клиент должен для начала работы зарегистрироваться в сервере.
2. В состоянии аутентификации сервера клиент может выбрать для работы папку с сообщениями.
3. В состоянии обмена с почтовой папкой сервера клиент производит основную работу с сообщениями.
4. В состоянии отсоединения сервер завершает транзакцию клиента.

Далее при описании команд, символом `S:` будет обозначаться поток данных от MAP4-сервера , а символом `C:` — поток данных от клиента.

- Команда `LOGIN`. После того как по транспортному протоколу (например, TCP) было установлено соединение и от сервера пришла строка приветствия, клиент должен зарегистрироваться в системе. Для этого чаще всего используется команда `LOGIN`. Аргументом команды является строка с идентификатором и паролем клиента:

```
S:   * OK IMAP4rev1 Service Ready
C:   a001 login ali sesam
S:   a001 OK LOGIN completed
```

  Команда `LOGIN` передает пароль и идентификатор пользователя по сети в открытом виде. Если пользователю необходима защита информации своей почты, он может пользоваться командой `AUTHENTICATE`. Аргументом команды является строка, указывающая механизм аутентификации, которым желает воспользоваться данный пользователь. В зависимости от выбранного типа аутентификации строится дальнейший обмен между сервером и клиентом. Например, при использовании механизма шифрования KERBEROS аутентификация выглядит следующим образом:

```
S: * OK KerberosV4 IMAP4rev1 Server
C: A001 AUTHENTICATE KERBEROS_V4
S: + AmFYig==
C: BAcAQU5EUkVXLkNNVS5FRFUAOCAsho84kLN3/IJmrMG+25a4DT
   +nZImJjnTNHJUtxAA+o0KPKfHEcAFs9a3CL5Oebe/ydHJUwYFd
   WwuQ1MWiy6IesKvjL5rL9WjXUb9MwT9bpObYLGOKi1Qh
S: + or//EoAADZI=
C: DiAF5A4gA+oOIALuBkAAmw==
S: A001 OK Kerberos V4 authentication successful
```

- После регистрации в системе клиент должен выбрать каталог (папку) сообщений, с которым он будет работать. Выбор каталога осуществляется командой `SELECT`. Аргументом команды является имя почтового каталога:

```
C: A142 SELECT INBOX
S: * 172 EXISTS
S: * 1 RECENT
S: * OK [UNSEEN 12] Message 12 is first unseen
S: * OK [UIDVALIDITY 3857529045] UIDs valid
S: * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
S: * OK [PERMANENTFLAGS (\Deleted \Seen \*)] Limited
S: A142 OK [READ-WRITE] SELECT completed
```

  IMAP4-сервер, прежде чем подтвердить завершение обработки команды, передает клиенту атрибуты данного каталога. В показанном выше примере:

  - в папке `INBOX` — 172 сообщения (строка `* 172 EXISTS`);
  - из них одно только что поступившее (строка `* 1 RECENT`);
  - в папке есть непрочитанные сообщения, минимальный порядковый номер непрочитанного сообщения — 12 (строка `* OK [UNSEEN 12] Message 12 is first unseen`);
  - уникальный временный идентификатор папки `INBOX` в данной сессии — 3857529045 (строка `* OK [UIDVALIDITY 3857529045] UIDs valid`);
  - сообщения в данной папке могут иметь флаги, указанные в строке `FLAGS` (строка `* FLAGS (\Answered \Flagged \Deleted \Seen \Draft)`);
  - клиент может менять у сообщений флаги `\Deleted` и `\Seen` (строка `* OK [PERMANENTFLAGS (\Deleted \Seen \*)] Limited`);
  - клиент имеет права на запись и чтение сообщений из папки `INBOX` (строка `A142 OK [READ-WRITE] SELECT completed`).

- Команда `SELECT` устанавливает текущий каталог для работы клиента. Если пользователю необходимо получить информацию о состоянии какого-либо каталога, достаточно воспользоваться командой `EXAMINE` с именем каталога в качестве аргумента команды, например:

```
C: A932 EXAMINE bloop
S: * 17 EXISTS
...
```

Команда EXAMINE возвращает те же параметры, что и команда SELECT, а отличается от команды SELECT только тем, что открывает заданный почтовый ящик исключительно на чтение.

- Если необходимо запросить статус какой-либо папки, не меняя текущий каталог, можно воспользоваться командой STATUS. В качестве параметров данной команде придаются имя папки и тип запрашиваемой информации. В зависимости от указанного типа, команда может возвращать: количество сообщений в папке, количество новых сообщений, количество непрочитанных сообщений, папки, определяемой уникальным действительным идентификатором UIDVALIDITY, идентификатор UID следующего сообщения, которое будет добавлено в данную папку, например:

```
C: A042 STATUS blob (MESSAGES UNSEEN)
S: * STATUS blob (MESSAGES 231 UNSEEN 12)
S: A042 OK STATUS completed
```

- Чтобы получить список папок (подкаталогов), находящихся в определенной папке и доступных клиенту, можно воспользоваться командой LIST. Аргументами команды являются: имя каталога, список подкаталогов которого хотим получить (пустая строка — " " означает текущий каталог) и маска имен подкаталогов. Имена каталогов и маски имен подкаталогов могут интерпретироваться по-разному, в зависимости от реализации почтовой системы и структуры описания иерархии папок. Например, список папок, находящихся в корне, можно получить так:

```
C: A004 LIST "/" *
S: * LIST (\Noinferiors) "/" INBOX
S: * LIST (\Noinferiors) "/" OUTBOX
S: * LIST (\Noinferiors) "/" WasteBox
S: A004 OK LIST completed
```

Ответ сервера содержит список папок в соответствии с их положением в иерархии и флаги данных папок (флаг \Noinferiors означает, что внутри данной папки нет и не может быть построена иерархия).

- После получения информации на каталог пользователь может прочитать любое сообщение или определенную группу сообщений, часть сообщения или определенные атрибуты сообщения. Для этого используется команда FETCH. Аргументами данной команды являются порядковый номер сообщения и критерии запроса. Критерии содержат описание вида возвращаемой информации. Например, можно запросить части заголовков или UID-сообщений в папке, или сообщения, имеющие или не имеющие определенные флаги. Так, запрос заголовков сообщений, находящихся в папке INBOX с порядковыми номерами от 10 до 12, будет выглядеть следующим образом:

```
C: A654 FETCH 10:12 BODY[HEADER]
S: * 10 FETCH (BODY[HEADER] {350}
S: Date: Wed, 17 Jul 1996 02:23:25 -0700 (PDT)
S: From: man@globe.com
S: Subject: Hi
S: To: imap@world.edu
S: Message-Id: <B27397-0100000@world.edu>
S: MIME-Version: 1.0
S: Content-Type: TEXT/PLAIN; CHARSET=US-ASCII
S:
S:)
S: * 11 FETCH....
S: * 12 FETCH....
S: A654 OK FETCH completed
```

- После просмотра сообщения пользователь может сохранить его с другими флагами, добавить или удалить флаги сообщения (например, пометить данное сообщение на удаление). Для этого используется команда `STORE`. Аргументами команды являются: номера сообщений, идентификатор операции и перечень флагов. Например, операция добавления флага удаления (`\Deleted`) трем сообщениям выглядит следующим образом:

```
C: A003 STORE 2:4 +FLAGS (\Deleted)
S: * 2 FETCH FLAGS (\Deleted \Seen)
S: * 3 FETCH FLAGS (\Deleted)
S: * 4 FETCH FLAGS (\Deleted \Flagged \Seen)
S: A003 OK STORE completed
```

  Ответом на выполнение команды будут переданы строки новых флагов указанных сообщений.

- Пользователь также может организовать поиск сообщений по определенным критериям. Для этого используется команда `SEARCH`. Критерий поиска состоит из комбинации нескольких условий поиска, а результатом поиска будет множество сообщений, находящихся в пересечении указанных условий. Условия могут налагаться на состав, структуру тела или (и) заголовка сообщений, а также на флаги, размер, идентификаторы, периоды дат сообщений. Результатом работы команды является строка, состоящая из последовательных номеров сообщений, удовлетворяющих критерию поиска. Например, поиск всех непрочитанных сообщений, поступивших от "`Smith`" с 1.03.96, будет выглядеть так:

```
C: A282 SEARCH UNSEEN FROM "Smith" SINCE 1-Mar-1996
S: * SEARCH 2 84 882
S: A282 OK SEARCH completed
```

Результатом поиска будут сообщения с последовательными номерами 2, 84 и 882.

- Протокол IMAP4 позволяет не только искать и читать сообщения в каталогах, этот протокол позволяет добавлять, копировать и перемещать сообщения в каталоги. Добавление сообщения в папку можно осуществить командой APPEND:

```
C: A003 APPEND saved-messages (\Seen) {310}
C: Date: Mon, 7 Feb 1997 21:52:25 -0800 (PST)
C: From: Fred Foobar <foobar@Blurdybloop.COM>
C: Subject: afternoon meeting
C: To: mooch@owatagu.siam.edu
C: Message-Id: <B27397-0100000@Blurdybloop.COM>
C: MIME-Version: 1.0
C: Content-Type: TEXT/PLAIN; CHARSET=US-ASCII
C:
C: Hello Joe, do you think we can meet at 3:30 tomorrow?
C:
S: A003 OK APPEND completed
```

  Отметим, что команда APPEND не осуществляет доставку сообщения по указанному адресу, она только размещает в данном каталоге набор строк в виде сообщения.

  Если сервер протокола IMAP4 поддерживает 8-битные данные, можно добавлять тексты сообщений в 8-битной кодировке, иначе текст должен быть закодирован в одну из 7-битных кодировок.

- Команда COPY копирует сообщения с заданными порядковыми номерами в указанный каталог, например:

```
C: A003 COPY 2:4 MEETING
S: A003 OK COPY completed
```

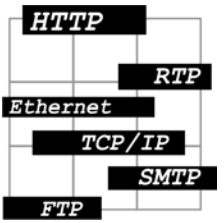
Пример взаимодействия IMAP4-клиента с сервером

Далее приведен простейший сценарий сессии взаимодействия IMAP4-клиента с сервером.

```
S:   * OK IMAP4rev1 Service Ready
C:   a001 login alladin sesam
S:   a001 OK LOGIN completed
C:   a002 select inbox
S:   * 18 EXISTS
```

```
S:   * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
S:   * 2 RECENT
S:   * OK [UNSEEN 17] Message 17 is the first unseen message
S:   * OK [UIDVALIDITY 3857529045] UIDs valid
S:   a002 OK [READ-WRITE] SELECT completed
C:   a003 fetch 12 body[header]
S:   * 12 FETCH (BODY[HEADER] {350}
S:   Date: Wed, 17 Jul 1996 02:23:25 -0700 (PDT)
S:   From: gray@cac.washington.edu
S:   Subject: IMAP4rev1 WG mtg summary and minutes
S:   To: imap@cac.washington.edu
S:   Message-Id: <B27397-0100000@cac.washington.edu>
S:   MIME-Version: 1.0
S:   Content-Type: TEXT/PLAIN; CHARSET=US-ASCII
S:
S:  )
S:   a003 OK FETCH completed
C:   a004 store 12 +flags \deleted
S:   * 12 FETCH (FLAGS (\Seen \Deleted))
S:   a004 OK +FLAGS completed
C:   a005 logout
S:   * BYE IMAP4rev1 server terminating connection
S:   a005 OK LOGOUT completed
```

Глава 11



Протокол передачи новостей — NNTP

NNTP (Network News Transport Protocol, Протокол передачи новостей) предназначен для тиражирования статей распределенной системы дискуссий UseNet.

NNTP — это протокол прикладного уровня. Он работает поверх транспортного протокола сети (например, TCP) и обеспечивает взаимодействие между участниками дискуссий UseNet по модели клиент-сервер.

В первое время развития в сети Интернет для передачи пакетов новостей, как, впрочем, и для передачи электронной почты, использовался протокол UUCP (UNIX to UNIX Copy). Этот сервис применяется и до сих пор, но в большинстве случаев вытесняется более быстрым и более гибким протоколом NNTP.

Система новостей UseNet

Интернет-сообщество уже много лет пользуется системой тиражирования объявлений, рекламы, сообщений и другой информации. Всю подобным образом тиражируемую информацию называют новостями — "news". Новости делятся на группы новостей — "newsgroups". Группы новостей организованы в определенном порядке, основанном на распределении дискуссий по темам, например, отдых, спорт, новости, информация, религия и др. Внутри каждой из этих групп может быть до нескольких тысяч подгрупп, которые, в свою очередь, обладают такой же иерархической структурой. Принцип этой организации подобен структуре размещения каталогов и подкаталогов на жестком диске.

К примеру, группа, обменивающаяся информацией об отдыхе (**rec**reation), использует ключ **rec** для построения иерархии более низкого уровня, такой как **rec.arts**, **rec.games**, **rec.pets**, **rec.sports**, **rec.travel** и т. д. Каждая из подгрупп, например, **rec.pets**, может быть, в свою очередь, разделена на более мелкие подгруппы, например, **rec.pets.dogs**, **rec.pets.cats** и др. Так организованные группы новостей позволяют пользователям, имеющим общие интересы, обмениваться новостями, отправлять свои статьи и отвечать на заметки других пользователей, обмениваться иллюстрациями и программным обеспечением для просмотра статей этой группы и т. п.

Топологически система UseNet представляет собой направленный граф. Каждый узел графа будет обозначать хост системы, а каждое ребро — путь передачи статей от одного хоста к другому. Систему UseNet можно представить себе как множество подсетей, по каждой из которых передаются статьи определенной конференции. Большинство соединений устроено таким образом, что статьи системы UseNet могут передаваться в обе стороны, однако на практике передача осуществляется только в одном направлении.

Механизм передачи статей заключается в следующем. Предположим, что какое-либо сообщение попадает в конференцию на одной машине, например, оно поступило от пользователя. Это сообщение принимается данным хостом в локальную базу данных и передается всем его соседям, которые "подписаны" по крайней мере на одну из конференций, к которой относится данная статья. Этот процесс будет продолжаться до тех пор, пока все хосты подсети данной конференции не будут располагать этой статьей.

Если каждому ребру графа системы UseNet поставить в соответствие идентификатор конференции, тогда отношение "подписки" топологически можно представить следующим образом: хост A "подписан" на конференцию хоста B, если между A и B существует ребро графа, принадлежащее подсети данной конференции.

Примечание

Иногда удается построить централизованную систему дискуссий на основе распределенной сетевой файловой системы. Безусловно, такой вариант будет хорошо работать, но только в однородной среде, когда все участники дискуссий имеют подобное аппаратное и программное обеспечение.

Важной частью алгоритма передачи новостей системы UseNet является предотвращение зацикливаний. Например, если хост A отправил статью хосту B, то хост B может, в свою очередь, отправить эту же статью обратно хосту A и т. д., поскольку передача статей может происходить как в одну, так и в другую сторону.

Одним из решений этой проблемы является отслеживание истории прохождения хостов данной статьей. Так, каждый хост, при прохождении через него статьи, фиксирует уникальный идентификатор этого сообщения (Message-ID). Если сообщение с тем же номером приходит на хост повторно, оно уже не принимается. Другой способ предотвращения петель состоит в фиксировании пути внутри самого сообщения. То есть сообщение никогда не будет отправлено на хост, зафиксированный в специальном поле (Path) заголовка статьи.

Способы тиражирования сообщений

Широко используются два основных метода тиражирования сообщений:

1. Отправка статей по электронной почте по спискам рассылки участников конференций.

2. Использование распределенной базы данных и серверов новостей системы UseNet.

Тиражирование новостей по почте в соответствии со списками рассылки реализуется благодаря использованию списков рассылки. Определенные хосты, например, компьютеры пользователей или почтовые хосты, выполняющие сервис тиражирования, хранят у себя списки адресов подписчиков на определенные дискуссии. Эти списки используются для отправки каждому из подписчиков копии тиражируемой для него информации — статей групп дискуссий. Статьи передаются от хоста к хосту как обычные почтовые сообщения, при этом каждый из хостов должен переправить всю поступившую информацию конференций на хосты, указанные в его списке рассылки. Эти хосты, в свою очередь, должны переправить полученную информацию следующим хостам, и т. д.

Этот способ хорошо работает, когда список подписчиков не превышает нескольких десятков адресов. При возрастании количества участников дискуссии данная схема будет требовать значительных сетевых ресурсов. При добавлении или удалении пользователей отдельную проблему для участников таких дискуссий может составить обновление списков подписки на серверах тиражирования. Кроме того, такой способ тиражирования предъявляет дополнительные требования к надежности работы серверов, на которых хранятся списки рассылки подписчиков конференций.

Другой способ тиражирования информации заключается в использовании системы серверов новостей и распределенной базы данных для хранения информационных сообщений. Этот метод используется в системе UseNet и объединяет множество различных сетей с разнообразным аппаратным и программным обеспечением.

Все статьи новостей данной подсети хранятся в одном месте, например, на главном хосте локальной сети или внешнем сервере с высокоскоростным доступом, откуда пользователи системы, расположенные на данной подсети, с помощью своего локального программного обеспечения самостоятельно выбирают статьи для чтения. Пользователи при этом используют технологию клиент-сервер и могут запрашивать только интересующие их статьи, а не копировать себе ненужную информацию.

Серверы новостей

Узел сети, предоставляющий пользователям возможность работать с хранящимися на нем статьями и управляющий доступом к новостям и обновлениям статей, называется *сервером новостей*. Используя протокол NNTP, серверы новостей обмениваются между собой статьями новостей. Механизм протокола NNTP позволяет серверам новостей выбирать, какие статьи следует принимать и передавать на другие серверы.

Клиент, инициирующий отправку, проверяет, существует ли на сервере группа новостей — конференция (Newsgroup), к которой относится данная статья, после чего она отправляется. Затем клиент запрашивает список новых статей, поступивших на сервер, на основании которого он может запросить новые статьи. В завершение клиент сообщает серверу о тех статьях, которые он уже имеет, для того чтобы сервер не отправлял эти статьи клиенту, как новые, если они поступят еще раз.

Для небольшого количества пользователей идеальной схемой построения обмена новостями была бы структура, состоящая из одного News-сервера и пользователей, обращающихся к нему для отправки или получения новых статей. В системах с большим количеством клиентов, например, масштаба университета или крупного предприятия, необходимо использовать так называемые промежуточные серверы новостей — *intermediate news server*.

Такой промежуточный сервер работает в каждом домене сети организации и отвечает за обеспечение посреднических действий между своими клиентами и главным News-сервером. В его обязанности, например, входит кэширование новых статей для обслуживания запросов подписчиков конференций, находящихся внутри домена. Для взаимодействия с системой UseNet клиент такой сети сначала соединяется со "своим" промежуточным сервером и читает размещенные на нем статьи новостей. Если сервер оказывается неработоспособным или у него нет требуемых статей, клиент может обратиться на сервера новостей более высокого уровня иерархии.

Пользователи небольших сетей, состоящих из нескольких компьютеров и не имеющих возможности или целесообразности устанавливать свой промежуточный сервер, как правило, обращаются к серверу, обслуживающему большее количество дискуссий или имеющему более доступный канал соединения. Статья новостей, попавшая на один из серверов, основной или промежуточный, должна быть разослана всем участникам данной дискуссии. Обязанность тиражирования статей между собой целиком лежит на серверах системы UseNet.

Формат статей и сообщений системы UseNet

После того как пользователь, используя локальное программное обеспечение для работы с системой новостей UseNet, сформировал свою статью, т. е. присоединил файлы, вставил рисунки и т. п., так же, как и в случае отправки обычного почтового сообщения, статья "заворачивается" в конверт системы UseNet, во многом похожий на конверт сообщений протокола SMTP.

Стандартный формат сообщений электронной почты определяет структуру самого сообщения, расположение и типы его частей, форматы кодировок *(см. разд. "Формат почтовых сообщений, MIME" гл. 10)*. Формат статьи

UseNet дополняет формат сообщений электронной полями, специфичными для системы передачи новостей. Текст и конверт почтового сообщения образуют единую структуру, которая используется для построения статьи UseNet.

Можно говорить, что формат статей UseNet представляет собой дополнительную структуру, позволяющую почтовым сообщениям тиражироваться через систему UseNet по протоколу NNTP.

Структура почтового сообщения подробно рассматривалась в разделе, посвященном протоколу SMTP. Поэтому, обсуждая структуру статьи UseNet, остановимся на описании только тех частей сообщения, которые добавляются системой UseNet. Типичный заголовок статьи UseNet выглядит следующим образом:

```
From: jerry@cock.att.com (Jerry Balls)
Path: cbosgd!mhuxj!mhuxt!cock!jerry
Newsgroups: news.announce
Subject: UseNet Etiquette -- Please Read
Message-ID: <642@cock.att.com>
Date: Fri, 19 Nov 97 16:14:55 GMT
Followup-To: news.misc
Expires: Sat, 1 Jan 98 00:00:00 -0500
Organization: AT&T Bell Laboratories

This is a trivial news!
```

Стандартное UseNet-сообщение (так же, как и почтовое сообщение) состоит из нескольких строк заголовка, отделенных от тела сообщения пустой строкой. Каждая строка состоит из ключевого слова, двоеточия ":", пробела, информации и символов завершения строки — "CR" и "LF". Далее для упрощения изложения при описании полей заголовков они будут изображаться без двоеточия. Некоторые из полей заголовка могу быть опущены. К таким полям относятся: `Followup-To`, `Expires`, `Reply-To`, `Sender`, `References`, `Control`, `Distribution`, `Keywords`, `Summary`, `Approved`, `Lines`, `Xref`, `Organization`. Обязательные поля должны присутствовать в заголовке каждой статьи: `From`, `Date`, `Newsgroups`, `Subject`, `Message-ID`, `Path`.

- Поля `From`, `Date`, `Subject`, `Message-ID` содержат те же параметры, что и аналогичные поля конверта почтового сообщения. Форматы этих полей в статьях системы UseNet зависят от программного обеспечения клиента для работы с системой.
- Поле `Newsgroups` заголовка статьи определяет группу и содержит название дискуссии или конференции, которой принадлежит данная статья. Одна и та же статья может принадлежать нескольким группам, тогда их названия перечисляются через запятую.

Все группы, упомянутые в поле Newsgroups, должны существовать. Если в поле Newsgroups поступившего сообщения содержатся имена недоступных для данного сервера или несуществующих групп дискуссий, эти имена при обработке статьи игнорируются.

- Поле Path содержит путь, пройденный сообщением для того, чтобы попасть в данную систему. Когда система переправляет сообщение с одного хоста на другой, она добавляет свое имя в поле Path. Имена хостов могут отделяться друг от друга любым знаком пунктуации за исключением точки — "." (поскольку точка используется в именах хостов). Например, так: Path: cbosgd!mhuxj!mhuxt или Path: cbosgd, mhuxj, mhuxt, или Path: @cbosgd.att.com,@mhuxj. att.com,@mhuxt. att.com. Как правило, имя следующего хоста добавляется в строчку слева, поэтому хост — инициатор сообщения всегда стоит в этой строке справа. Это нужно для обеспечения совместимости со старыми системами. Например, хост A получил сообщение с параметром Path: X!Y!Z. Если хост A отправляет его хосту B, то хост B получит статью с параметром Path: A!X!Y!Z, и т. д.

  Параметр Path не используется для адресации ответов в качестве почтового адреса (за исключением старых версий некоторых программ). Он предназначен для хранения и обработки информации маршрута данной статьи. Этот параметр играет очень важную роль при построении графа тиражирования данного сообщения и предотвращения "зацикливаний" передаваемых статей.

  Если, например, хост A отправляет статью хосту B, параметр Path статьи будет содержать имя хоста A. Поэтому хост B или какой-либо из следующих хостов, впоследствии обрабатывающих эту статью, не отправит ее обратно на хост A.

Среди факультативных полей заголовка статьи, можно выделить следующие поля, специфичные для статей системы UseNet:

- Поле Followup-To имеет тот же формат, что и поле Newsgroups. Если это поле присутствует, дополнительные статьи, относящиеся к данной статье, направляются в конференцию(ии), указанные в этом поле. Если поле отсутствует, дополнения к данной статье направляются в конференцию(ии), указанные в поле Newsgroups.

- Поле Expires содержит дату "истечения срока действия" сообщения. Поле может быть использовано для удаления статей, содержащих информацию, имеющую ограниченный срок действия (для того чтобы статьи, содержащие критичную по времени информацию, находились в сети заданное время). Это может относиться, например, к объявлению о дате какого-либо симпозиума или семинара. Если это поле не указано, используется значение, устанавливаемое локальным программным обес-

печением по умолчанию, например, по соображениям экономии дискового пространства. Пользователям настоятельно не рекомендуется использовать это поле, если в этом нет необходимости.

- Поле `References` содержит список идентификаторов сообщений (`Message-ID`), на которые ссылается данное сообщение. Это поле обязательно должно присутствовать во всех дополнительных сообщениях. Как правило, поле `Subject` ответных сообщений содержит префикс `Re`. Например:

  ```
  References: <AAckqIqWy5@jone.tamu.edu>
  Message-Id: <AARABLqq2H@lisa.silly.edu>
  Subject: Re: Hello
  ```

  Если исходное сообщение содержало поле `References`, то дополнительное сообщение должно содержать в этом поле значения `References` и значение поля `Message-ID` этого сообщения. Данное поле позволяет группировать статьи по тематическому принципу и организовывать временные иерархические структуры внутри конференции. Если строка `References` получается слишком длинной, локальное программное обеспечение может ее "обрезать", сохраняя количество ссылок, достаточное для идентификации потока обсуждения.

- Поле `Organization` содержит имя организации, к которой принадлежит отправитель или хост отправителя.

- Поле `Keywords` содержит ключевые слова-идентификаторы сообщения. Оно используется при поиске сообщения конкретного содержания.

- Поле `Summary` содержит краткое описание содержания статьи.

- Поле `Distribution` используется для ограничения распространения статей определенных конференций. Поле содержит идентификаторы зон распространения статьи. Например:

  ```
  Newsgroups: rec.auto,misc.forsale
  Distribution: nj,ny
  ```

  Данная статья будет передаваться подписчикам конференций "rec.auto" и "misc.forsale", расположенным в доменах New Jersy — **nj** и New York — **ny**. Эту же задачу можно было решить, если создать для подписчиков New York и New Jersy отдельные конференции, например, "nj.rec.auto", "ny.rec.auto", "nj.misc.forsale" и "ny.misc.forsale". Но если вам нужно отправить одну или несколько статей ограниченному кругу лиц, создавать для этого специальные конференции крайне неэффективно.

- Поле `Xref` содержит имя локального хоста без идентификатора домена, номер данной статьи и номер данной конференции в почтовой системе локального хоста. Например, если заголовок сообщения содержит следующие строки:

  ```
  ...
  Newsgroups: news.lists,news.groups
  ```

```
...
Xref: alibaba news.lists:461 news.groups:6378
```

Это означает, что данное сообщение имеет номер 461 в группе "news.list" и 6378 в группе "news.groups" локального хоста "alibaba". Эти значения могут быть использованы локальным программным обеспечением.

Управляющие сообщения

Если сообщение содержит строку `Control`, это означает, что данное сообщение является управляющим. Управляющие сообщения используются для обмена служебной информацией между UseNet-хостами (а не пользователями). Управляющие сообщения передаются с использованием того же самого механизма, как и обычные сообщения, т. е. через систему групп или конференций подписки.

Примечание

Для совместимости со старыми версиями системы сообщения адресованные группе "all.all.ctl", считаются управляющими. Если сообщение этой группы не содержит поля `Control`, поле предмета сообщения `Subject` для таких сообщений интерпретируется как параметр `Control`. Кроме того, если в сообщениях этой группы первые 4 символа поля `Subject` содержат подстроку `cmsg`, оставшаяся часть поля `Subject` интерпретируется как управляющее сообщение.

Поля `Control` заголовка сообщения содержат команду и параметры команды: первое слово представляет собой команду, остальные — параметры этой команды. Другие поля заголовка и тело сообщения тоже могут использоваться как потенциальные параметры команды. Например:

```
Control: cancel <any.string@host.domen>
```

Управляющие сообщения передаются автоматически программным обеспечением хоста системы UseNet, и отправлять управляющие сообщения вручную не рекомендуется. Сообщения о поступлении неправильно составленных управляющих сообщений и об ошибках их обработки передаются не отправителю сообщения, а администратору локальной системы, из которой сообщение было отправлено. Далее перечислим управляющие сообщения.

- Сообщение `cancel` позволяет пользователю удалять статьи из локальной системы уже после того, как статья была тиражирована. Параметр команды — идентификатор сообщения `Message-ID`.

  ```
  cancel <Message-ID>
  ```

  Отправить такую команду может только автор статьи или администратор локальной UseNet-системы. Обработчик данного управляющего сообщения, прежде чем его применить, сверяет параметры `Sender` и `From` управляющего и исходного сообщений.

- Сообщения `ihave` и `sendme`. Эти сообщения являются частью ihave/sendme-протокола, который позволяет одному хосту (A) сообщить другому хосту (B), что данное сообщение им получено — команда `ihave` (а другому хосту (B) запросить это сообщение — команда `sendme`).

  ```
  ihave <Message-ID list> [<remotesys>]
  sendme <Message-ID list> [<remotesys>]
  ```

  Например, хост (A) получил сообщение `<1234@vax.berkeley.edu>` и хочет передать его на хост (B). Тогда (A) отправляет через какую-либо из конференций, на которые подписан (B), хосту (B) управляющее сообщение `ihave <1234@vax.berkeley.edu>` A. Хост (B) отвечает, опять же через какую-либо из конференций, на которые подписан хост (A), управляющим сообщением `sendme <1234@vax.berkeley.edu>` B. После получения сообщения `sendme`, хост (A) отправляет хосту (B) данное сообщение. Эти команды позволяют хостам выбирать статьи для пересылки, тем самым уменьшая передаваемый поток новостей "вслепую". Однако, если размер сообщения ненамного превышает размер управляющего сообщения, пользоваться этим механизмом невыгодно. В таких случаях применяют групповой запрос сообщений. Для этого строится команда, в которой идентификаторы требуемых сообщений (`Message-ID`) содержатся не в поле заголовка, а в теле сообщения — по одному в каждой строке.

- Управляющие сообщения могут создавать или удалять группы (конференции) новостей.

  ```
  newgroup <groupname> [moderated]
  ```

  Это сообщение создает конференцию с именем `groupname`. Тело данного управляющего сообщения может содержать краткую аннотацию новой группы (предназначение, уровень и темы дискуссий и т. д.). Если указан аргумент `moderated`, то это означает, что у данной группы есть председательствующий, который следит за уровнем и темами дискуссий. По умолчанию новая конференция не имеет председательствующего. Статья новостей может быть отправлена в конференцию только после того, как данная конференция будет создана.

  ```
  rmgroup <groupname>
  ```

  Это сообщение удаляет конференцию с именем `groupname`.

- Управляющее сообщение `sendsys` (без аргументов) представляет собой запрос к UseNet-хосту на получение списка всех его соседей и групп статей новостей, которые он передает своим соседям. Ответ на запрос передается отправителю данного управляющего сообщения, указанному в поле `Reply-To`, или в поле `From` управляющего сообщения. Передаваемая в ответ информация общедоступна и должна быть отправлена по запросу автоматически или вручную, в той или иной форме. Например,

ответное сообщение может быть таким: (пользователь **mark** отправил запрос своему хосту):

```
From: cbosgd!mark  (Mark Horton)
Date: Sun, 27 Mar 83 20:39:37 -0500
Subject: response to your sendsys request
To: mark@home.att.com

Responding-System: cbosgd.att.com
cbosgd: osg,cb,btl,bell,world,comp,sci,rec,talk,misc,news,soc
ucbvax: world,comp,to.ucbvax
cbosg: world,comp,bell,btl,cb,osg,to.cbosg
cbosgb: osg,to.cbosgb
sescent: world,comp,bell,to.sescent:F:/usr/spool/outnews/sescent
```

Первая строчка тела сообщения содержит имя отвечающей системы, затем располагается информация ответа. Разделитель колонок — символ ":". Первая колонка содержит имя хоста соседа, вторая — имена конференций, в которые данный хост отправляет статьи. Третья и четвертая колонки не специфицированы, их интерпретация зависит от конкретной реализации программного обеспечения.

- Управляющее сообщение `version` (без аргументов) используется для запроса имени и версии программного обеспечения системы UseNet, используемой на запрашиваемом хосте.

Модели передачи статей

Как уже отмечалось выше, система UseNet для формирования и тиражирования своих статей использует почтовые сообщения. Поэтому для функционирования системы UseNet настоятельно рекомендуется, чтобы участвующие в обмене сообщениями хосты использовали совместимые почтовые системы. В противном случае система будет использовать только часть параметров, что существенно уменьшит возможности организации дискуссий.

Примечание

Например, поле `From` сообщений новостей нельзя будет использовать в его стандартном виде, если синтаксис имен хостов во взаимодействующих системах будет различным.

Системы, использующие для передачи статей дискуссий электронную почту (системы, использующие метод передачи по спискам рассылки), должны отличать отправку и получение статей дискуссий от обычных почтовых сообщений. В противном случае могут возникнуть проблемы, связанные, например, с идентификацией поля `From`, поскольку отправители почтового

сообщения и статьи могут не совпадать. Другая проблема может возникнуть с интерпретацией ошибочных сообщений, которые отличаются у почтовой системы и системы новостей.

Одним из решений этой проблемы является инкапсуляция статьи дискуссии, вместе с заголовком, в тело почтового сообщения. Для этого в начало каждой строчки статьи новостей, как тела, так и заголовка, вставляется буква "N". После этого статья записывается в тело почтового сообщения. Это гарантирует, что ни одна строчка статьи не будет интерпретироваться почтовым сервером как управляющая для почтового сообщения. Программа, которая получит такое сообщение, извлекает из него статью и передает ее соответствующей программе обработки статей дискуссий.

В таком формате сообщение будет выглядеть следующим образом:

```
Date: Mon, 3 Jan 95 08:33:47 MST
From: news@cbosgd.att.com
Subject: network news message
To: rnews@npois.att.com

NPath: cbosgd!mhuxj!harpo!utah-cs!sask!derek
NFrom: derek@sask.UUCP
NNewsgroups: misc.test
NSubject: test
NMessage-ID: <176@sask.UUCP>
NDate: Mon, 3 Jan 95 00:59:15 MST
N
NThis really is a test.
N
```

Подобная форма передачи статей позволяет отдельно задавать приоритеты почтовых сообщений и статей новостей.

Как правило, статьи новостей имеют небольшой размер, поэтому имеет смысл передавать их пакетами, по несколько статей в каждом. Такие пакеты имеют свой формат, понятный взаимодействующим хостам.

Например, пакет может быть построен таким образом, что каждому сообщению в пакете будет предшествовать строка `#!rnews <number>`, где `<number>` — длина статьи. Такой пакет будет выглядеть следующим образом:

```
#! rnews 239
From: jerry@eagle.att.com (Jerry Schwarz)
Path: cbosgd!mhuxj!mhuxt!eagle!jerry
Newsgroups: news.announce
Subject: UseNet Etiquette -- Please Read
```

```
Message-ID: <642@eagle.att.com >
Date: Fri, 19 Nov 96 16:14:55 EST
Approved: mark@cbosgd.att.com

Here is an important message about UseNet Etiquette.
#! rnews 234
From: jerry@eagle.att.com (Jerry Schwarz)
Path: cbosgd!mhuxj!mhuxt!eagle!jerry
Newsgroups: news.announce
Subject: Notes on Etiquette message
Message-ID: <643@eagle.att.com >
Date: Fri, 19 Nov 96 17:24:12 EST
Approved: mark@cbosgd.att.com

There was something I forgot to mention in the last message.
```

В данном примере, символ "#" используется как идентификатор, а строка `rnews` задает формат пакета.

Примечание

Некоторые сети позволяют работать с механизмом удаленных процедур. В таких сетях статьи новостей могут передаваться через удаленные вызовы процедур серверов новостей (например, через стандартные потоки ввода/вывода UUCP-соединения). В этом случае важно, чтобы статьи отправлялись через достоверное соединение, т. е., если во время работы удаленной системы произойдет сбой, поток передачи должен восстановиться при восстановлении ее работоспособности.

Протокол NNTP, описание команд

Сервер новостей системы UseNet использует протокол NNTP для взаимодействия с другими серверами. Алгоритм протокола NNTP во многом похож на алгоритм протокола SMTP. Протокол поддерживает канал обмена новостями и простейший интерфейс с распределенной базой данных новостей. При использовании в качестве транспортного протокола TCP протокол NNTP, как правило, использует порт 119.

Команды и ответы протокола состоят из ASCII-символов. Если соединение позволяет передавать 8-битные данные, восьмой (старший) бит устанавливается равным нулю. Команды могут содержать параметры, которые отделяются пробелами или символами табуляции. Каждая строка может содержать только одну команду (с параметрами или без), должна заканчиваться парой символов "CR" и "LF" и быть длиной не более 512 байтов.

После обработки очередной команды сервер новостей возвращает статус ее обработки. Ответ сервера представляет собой строку, состоящую из числового кода обработки и текста (который может быть либо комментарием, либо возвращаемой по команде информацией).

Примечание

Числовые коды ответов сервера новостей организованы по иерархическому принципу и обозначают состояние сервера после выполнения команды. Описание кодов возврата выходит за рамки данной книги. Подробное описание структуры системы кодов вы можете найти, например, в документе RFC-977.

Итак, после того как клиент установил TCP-соединение с сервером новостей, сервер отвечает приглашением к работе по протоколу NNTP (далее символом `S:` будем обозначать сервер новостей, а символом `C:` — клиента). Например:

```
S: 200 newsvax news server ready — posting ok
```

Код `200` означает, что данный сервер разрешает клиенту передавать ему новые статьи. Код ответа — `201`, обозначает, что сервер не разрешает клиенту передавать статьи:

```
S: 201 badvax netnews server ready -- no posting allowed
```

Теперь клиент может начать работу. Далее будут описаны наиболее употребительные в ежедневной работе команды протокола NNTP.

- Команда `LIST` возвращает список групп сервера новостей и сопутствующую информацию. Например:

  ```
  C: LIST
  S: 215 list of newsgroups follows
  S: net.wizards 10125 10011 y
  S: net.idiots 00100 00001 n
  S:...
  S: <groupname> <last> <first> p
  ```

  где `<groupname>` — имя группы новостей, `<last>` и `<first>` — номера последней и первой статьи в группах, `p` — флаг, показывающий, может ли клиент отправлять свои статьи в эти группы (`n` — не может, `y` — может).

  Некоторые конференции могут быть не доступны для чтения. Например, статьи группы должны быть определенным образом зашифрованы или могут передаваться в группу только через председательствующего группы — модератора. Эта возможность не зависит от общих настроек сервера, а определяется параметрами конкретной группы.

- Для того чтобы получить статьи из конкретной группы, пользователь должен к ней подсоединиться. Это делается с помощью команды `GROUP`.

В качестве параметра этой команды задается название группы новостей. Например:

```
C: GROUP net.deneg
S: 211 104 10011 10125 net.deneg group selected
   (there are 104 articles on file, from 10011 to 10125)
```

где 211 — код успешного открытия группы (если код = 411, такой группы нет), число 104 — количество статей в группе, 10011 — номер первой статьи группы, 10125 — номер последней статьи. После выполнения этой команды создается указатель на текущую статью, который по умолчанию будет указывать на первую статью в группе. Если пользователь не знает названия группы, то он может взять его из списка, возвращаемого командой LIST.

- После того как выбрана группа новостей, пользователь может прочитать статью из этой группы. Для этого можно использовать команды: ARTICLE, BODY, HEAD. В качестве аргументов команды используют либо номер статьи, либо ее уникальный идентификатор (Message-ID). Последний может использоваться, например, для предотвращения повторного чтения одной и той же статьи. Разница между этими командами заключается в том, что команда ARTICLE возвращает как заголовок, так и тело статьи, команда HEAD возвращает только заголовок, а BODY — только тело сообщения. Например:

```
C: ARTICLE <87623@baz.net>
S: 220 <87623@baz.net> All of article follows
S:... text...

C: BODY 402
S: 220 402 <4105@ucbvax.arpa> Article retrieved, body follows
S:... text...

C: HEAD 404
S: 220 404 <mik@vax.net> Article retrieved, head follows
S:... text...
```

- Сообщения в группе можно читать последовательно, не обращаясь к номеру статьи. Для этого требуется установить указатель на текущую статью. Это делает команда STAT. Аргументом команды является или номер статьи в группе, или уникальный идентификатор сообщения (Message-ID). После того как указатель установлен, вы можете читать текст статей, например, командой BODY, не указывая номера. Для того чтобы передвинуть указатель на следующую статью, следует подать

команду NEXT без аргументов, соответственно, на предыдущую статью — LAST. Например:

```
C: STAT 110
S: 223 110 <23445@svax.com> article retrieved — statistics
C: BODY
S: 222 110 <23445@svax.com> article retrieved — body follows
S:... text...
C: NEXT
S: 223 113 <21495@nuke.uucp> article retrieved — statistics
```

- Пользователь может получить от сервера новостей список групп, которые были созданы позднее заданной даты. Этот сервис предоставляется командой NEWGROUPS. Аргументами команды NEWGROUPS являются дата и время в формате "YYMMDD HHMMSS", где YY — год, MM — месяц, DD — день, HH — часы, MM — минуты, SS — секунды; кроме того, флаг отсчета времени (локальное или по GMT (Greenwich Mean Time, время по Гринвичу)) и маска имен групп. Например, запрос списка групп, созданных на данном сервере с 15 мая 1995 года по GMT и имеющих в своем названии строку net, выглядит следующим образом:

```
C: NEWGROUPS 950515 000000 GMT <net>
S: 235 New newsgroups since 950515 follow
S: net.fluff
S:...
```

- Пользователь может получить от сервера новостей список уникальных идентификаторов статей, которые были созданы в указанной группе позднее заданной даты. Этот сервис предоставляется командой NEWNEWS. Аргументами команды являются дата и время в формате "YYMMDD HHMMSS", флаг отсчета времени (локальное или по GMT), маска имени группы новостей. Маска имени в данной команде позволяет использовать символ джокера ("*") и символ отрицания ("!"). Например, запрос списка статей, полученных всеми группами новостей данного сервера после 16 июня 1997 года, выглядит следующим образом:

```
C: NEWNEWS * 970616 000000
S: 230 New news since 970616 000000 follows
S: <1772@foo.UUCP>
S:...
```

- Команда IHAVE информирует сервер, что пользователь имеет статью с идентификатором <Message-ID>, который указывается в качестве аргумента данной команды. Например:

```
C: IHAVE <4105@vax.com>
S: 435 Already seen that one, where you been?
```

```
C: IHAVE <4109@vax.com>
S: 335 News to me! <CRLF.CRLF> to end.
C: (sends article)
C:.
S: 235 Article transferred successfully.  Thanks.
```

Если сервер уже имеет данную статью, он ответит, что копия статьи у него уже есть. Если указанной статьи нет, сервер запросит ее отправить, и пользователь может отправить статью. При отправке клиент должен передать всю статью целиком, включая заголовок и тело сообщения, в том же формате, в каком он получает статьи от данного сервера. Передача сообщения завершается строчкой, содержащей символ "." (т. е. последовательностью <CRLF>.<CRLF>).

Ситуация, когда сервер новостей не имеет сообщения, в то время как оно находится у его клиента, может возникнуть, например, если клиент работает с несколькими серверами, которые еще не успели или не могут по каким-либо причинам обменяться данными сообщениями.

- Для того чтобы отправить в конференцию новую статью, пользователь должен сначала убедиться, что данная конференция доступна для записи новых статей и ввести команду POST. Сервер должен ответить кодом готовности принять сообщение (340). После этого пользователь может отправить статью, включая заголовок и тело сообщения, в том же формате, в каком он получает статьи от данного сервера. Передача сообщения завершается последовательностью "<CRLF>.<CRLF>". Например:

```
C: POST
S: 340 Continue posting; Period on a line by itself to end
C:...
C:.
S: 240 Article posted successfully.
```

- Завершает сессию обмена команда QUIT:

```
C: QUIT
S: 205 VAX closing connection. Goodbye.
```

Пример передачи статьи по протоколу NNTP

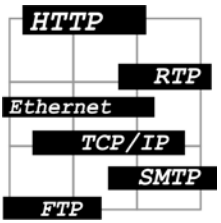
Далее представлен простейший пример передачи статьи по протоколу NNTP:

```
S: 200 atvax news server ready — posting ok
C: GROUP net. wizards
S: 211 104 10011 10125 net.wizards group selected
   (there are 104 articles on file, from 10011 to 10125)
```

```
C: STAT 10110
S: 223 10110 <23445@svax.ARPA> article retrieved — statistics
   only (article 10110 selected, its message-id is
   <23445@svax.ARPA>)
C: BODY
S: 222 10110 <23445@svax.ARPA> article retrieved — body follows
S:... text body...
S:.
C: NEXT
S: 223 10113 <21495@nudebch.uucp> article retrieved — statistics
   only (article 10113 was next in group)
C: ARTICLE 10120
S: 220 10120 <4105@ucbvax.ARPA> Article retrieved, text follows
S:... text article...
S:.
C: QUIT
S: 205 goodbye.
```

Глава 12

Протокол HTTP



Протокол HTTP (Hypertext Transfer Protocol, Протокол передачи гипертекста) представляет собой протокол прикладного уровня и обеспечивает возможность доступа к разнообразной информации, размещенной в сети WWW (World Wide Web). Протокол HTTP обладает высокопроизводительными механизмами тиражирования информации мультимедийных систем, независимо от типа представления данных. Протокол построен по объектно-ориентированной технологии и может использоваться для решения различных задач, например, работы с серверами имен или управления распределенными информационными системами.

Способность хранить и представлять данные разнообразных форматов (изображения, видео, аудио) делает сеть WWW уникальным средством размещения информации. Сегодня на многочисленных WWW-серверах размещаются колоссальные объемы самой разнообразной информации, что, в конечном счете, и является причиной роста популярности самой сети Интернет.

С 1990 года протокол HTTP используется информационной системой WWW в качестве одного из основных протоколов.

Сервисы сети WWW

Протокол HTTP позволяет получать доступ к информационным ресурсам и сервисам WWW-серверов. Для унификации доступа к многофункциональным ресурсам сети WWW-серверы поддерживают комплекс интерфейсов, позволяющих структурировать уровни и методы доступа к сетевым ресурсам. По сути, каждый из интерфейсов представляет собой объект сети со своими методами и своей структурой. Для поиска и отображения информации, размещенной в сети WWW, применяются специальные приложения, называемые Web-браузер. Согласованное взаимодействие этих объектов и составляет все разнообразие сети WWW.

- URI (Uniform Resource Identifier, Идентификатор ресурса), URL (Uniform Resource Locator, Местонахождение ресурса), URN (Uniform Resource Name, Имя ресурса) — разные аспекты идентификации одного и того же сервиса, определяющие тип, метод доступа и расположение

узла сети, на котором находятся ресурс, доступный через сеть Интернет. Этот сервис состоит из трех частей.

- *Схема*. Идентифицирует тип сервиса, через который можно получить доступ к сервису, например FTP или WWW-сервер.
- *Адрес*. Идентифицирует адрес (хост) ресурса, например, **www.ripn.net**.
- *Имя или путь доступа*. Идентифицирует полный путь к ресурсу на выбранном хосте, который мы хотим использовать для доступа к ресурсу, например, **/home/images/image1.gif**.

Например, файл `readme.txt`, расположенный на сайте Microsoft (WWW-сервере), представляет собой ресурс с идентификатором: **http://www.microsoft.com/readme.txt**. Это означает, что для обращения к ресурсу должен использоваться протокол HTTP, (схема доступа отделена двоеточием **":"** и указывает название использованного протокола), следующие два слэша отделяют адрес сервера **www.microsoft.com**; а также имя файла **/readme.txt**.

Примечание

Как правило, когда имеют в виду компьютер, на котором расположен ресурс, используют значение URL или URN, а когда обозначают ресурс полностью (тип, хост, путь) используют URI. Видимо, большой беды в том нет, если вы используете одно обозначение вместо другого, но обязательно поясняйте, что оно значит в контексте.

Идентификатор URI может содержать не только имя ресурса, но и параметры, необходимые для его представления. Имя ресурса отделяется от строки параметров символом "?". Строка параметров состоит из символьных групп с постоянной структурой (лексем), разделяемых символом "&". Каждая такая лексема состоит из имени параметра и его значения, разделенных символом "=". Символы лексем, не входящие в набор символов ASCII, заменяются знаком "%" и шестнадцатеричным значением этого символа. Например, символ пробела " " заменяется значением "%20". Для указанного ресурса вся строка параметров является одним строковым параметром, поэтому тип, порядок следования или уникальность имен отдельных параметров строки не существенны. Например:

http://www.exe.com/bin/scrshell.run?in=10&go=ok%20and%20ok&event=1&event=2

Этот идентификатор URI содержит 4 параметра, три из которых — числовые, а два имеют одно имя. Анализ и разбор значений отдельных параметров целиком возлагается на идентификатор URI, в данном примере на ресурс scrshell.run.

- *HyperText Markup Language* (HTML) — это язык описания информации, хранимой в сети WWW. HTML-файл содержит специальные коды,

обозначающие присоединенную графическую, видео и аудио информацию или исполняемые коды среды отображения информации (Web-браузер — Java Script, Java Classes.) Когда Web-браузер получает доступ к этому файлу, он сначала интерпретирует закодированную в HTML-файле информацию, а затем в соответствующей форме представляет эту информацию пользователю.

Буквы "HT" в названии протокола HTML обозначают "HyperText" — основную концепцию размещения информации в сети WWW. Документы HyperText содержат специальные связи, которые называются *гиперссылками (hyperlinks)* и размещаются в тексте документа. Гиперссылки позволяют пользователю не только переходить от одной части этого документа к другой, но и обращаться к идеологически связанным документам, размещенным в сети WWW.

Примечание

Система файлов помощи Windows представляет собой пример использования механизмов гиперссылок. Щелкнув мышью на выделенном или подчеркнутом слове, пользователь может свободно перемещаться по всей системе файлов помощи, даже между разными файлами.

Управляющими конструкциями языка HTML являются *теги*. Теги представляют собой ключевые слова, заключенные в угловые скобки. Ключевыми в данном случае являются обычные слова английского языка, обозначающие ту или иную команду, которая должна быть применена к тексту, обрамленному тегами. Например:

```
<font color="red">This is a red color</font color>
```

Строка `This is a red color` будет выделена в Web-браузер красным цветом. HTML — это очень простой язык описания состояния объектов, но его подробное обсуждение выходит за рамки данной книги. Вы сможете найти множество материалов по последним модификациям стандартов и описания HTML на WWW-узле **http://www.w3.org**.

- *Common Gateway Interface* (CGI) — это стандарт расширения функциональности WWW, позволяющий WWW-серверам выполнять программы, аргументы которых может определять пользователь. Выше было отмечено, что WWW-серверы позволяют запрашивать статические HTML Web-страницы и просматривать их с помощью приложений Web-браузер. Интерфейс CGI расширяет возможности пользователя и позволяет ему выполнять программы, ассоциированные с данной Web-страницей, предоставляя таким образом возможность получения динамической информации из WWW-сервера. Например, пользователь такого WWW-сервера может получить самую последнюю информацию о погоде, выполнив программу, которая запрашивает прогноз погоды на текущий момент из базы данных.

Интерфейс CGI в основном играет роль шлюза между WWW-сервером и внешними исполняемыми программами. Он получает запрос от пользователя, передает его внешней программе и затем возвращает результаты пользователю через построенную динамически Web-страницу. При этом построенные Web-страницы могут коренным образом отличаться друг от друга, поскольку они формируются в прямой зависимости от параметров, определяемых пользователем.

Механизм интерфейса CGI также является универсальным и может передавать данные между любыми WWW-серверами. Поскольку интерфейс CGI основан на исполняемых файлах, нет ограничений на тип программы, которая будет в нем исполняться. Программа может быть написана на любом из языков программирования, позволяющем создавать исполняемые модули, а именно на C/C++, Fortran, Pascal, Visual Basic или PowerBuilder. CGI-программа также может быть написана с использованием командных языков операционных систем, таких как Perl или Shell.

В настоящее время широко используется технология активных серверных страниц ASP (Active Server Pages). По сути, эта технология представляет применение того же самого стандарта CGI, только на уровне объектно-ориентированного подхода к построению Web-страниц.

Примечание

Для языков Java и JavaScript приложение Web-браузер представляет операционную систему или среду, в которой они выполняются, а Web-страница является ресурсом, выделенным для их работы. Эти языки не строят Web-страницу по данным пользователя, а используют ее как платформу для своих действий и действий пользователя.

Принципы построения HTTP-соединения

Протокол HTTP построен с использованием стандартной модели "запрос/ответ". Клиент устанавливает соединение с сервером и отправляет запрос. В нем указывается тип запроса, URL (URI, URN), версия протокола HTTP (поскольку формат запроса может меняться от версии к версии) и содержание запроса: информация клиента (параметры) и, возможно, сопровождающая информация или тело сообщения. Сервер HTTP отвечает строкой состояния обработки запроса, которая содержит: версию поддерживаемого протокола, код обработки запроса или код ошибки и возвращаемую по запросу информацию. Сообщения, формируемые клиентом и сервером HTTP, должны быть представлены в формате стандарта MIME.

В простейшем случае соединение представляет собой поток данных между клиентом — инициатором соединения и сервером (рис. 12.1).

Рис. 12.1. Простейшая схема взаимодействия между клиентом и сервером HTTP

В более сложной ситуации в процессе передачи данных принимают участие несколько промежуточных объектов (рис. 12.2). Они могут быть трех видов: Proxy (агент-посредник), Gateway (шлюз) и Tunnel (туннель).

- *Proxy* представляет собой промежуточный агент, который принимает запрос от клиента и, в зависимости от своих настроек, изменяет часть или все сообщение запроса и передает переформатированный запрос далее по цепочке, например, другим серверам или запрашиваемому серверу. В момент принятия запросов Proxy выполняет функции HTTP-сервера, а при передаче запросов выполняет функции HTTP-клиента. Proxy часто используется как ретранслятор протоколов внутренней сети в Интернет и, как правило, представляет собой "главные ворота" *выхода* пользователей внутренней сети во внешние сети.
- *Gateway* (шлюз) представляет собой промежуточный сервер. В отличие от Proxy, шлюз принимает запросы клиента, будто он и есть запрашиваемый сервер, и передает их далее. Работа шлюза совершенно прозрачна для клиента. Шлюз, как правило, используется в качестве ретранслятора запросов внешней сети во внутреннюю, к ресурсам сервера, т. е. это "главные ворота" *входа* пользователей внешней сети.
- *Tunnel* (туннель) представляет собой программу-посредник между двумя соединениями. Туннели используются в тех случаях, когда необходимо организовать поток данных через какой-либо промежуточный объект (например, тот же Proxy), который не может интерпретировать структуру потока данных.

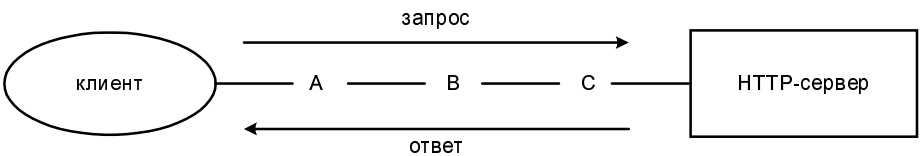


Рис. 12.2. Схема взаимодействия клиента и сервера через промежуточные объекты сети

На схеме, представленной на рис. 12.2, показано соединение между клиентом и сервером с тремя промежуточными объектами ("A", "B" и "C"). Сообщения запроса и ответа должны пройти через четыре отдельных

канала, а каждый из промежуточных объектов может одновременно поддерживать несколько соединений. Например, "B" может получать одновременно запрос как от "A", так и от других клиентов, и передавать сообщения на "C" и на другие серверы.

Каждый из объектов — участников соединения, за исключением туннеля, может поддерживать внутренний кэш запросов и ответов.

- *Кэш* представляет собой локальную базу данных сообщений ответов и систему управления этой базой. Кэш хранит ответы серверов и возвращает их по запросу клиента, не передавая запрос следующему объекту цепочки соединения (рис. 12.3).

  Эффект от применения кэша состоит в том, что при его использовании сокращается количество обращений к удаленным серверам и, тем самым, уменьшается среднее время доступа к информационным ресурсам.

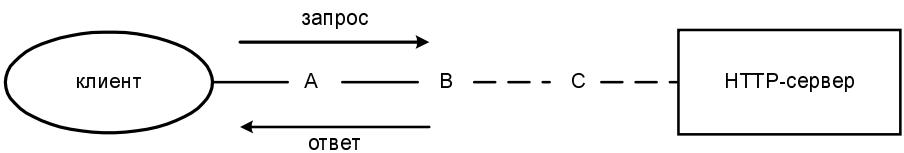


Рис. 12.3. Схема использования кэша

На представленной на рис. 12.3 схеме объект "B" передает ответ на запрос из своего кэша, который был построен на основании данных предыдущих запросов.

Однако необходимо помнить, что далеко не все ответы могут кэшироваться. Некоторые запросы могут содержать параметры, накладывающие ограничения на возможность использования кэша.

HTTP — это протокол прикладного уровня, который, как правило, работает поверх протокола транспортного уровня TCP, хотя, как всякий протокол прикладного уровня, может работать с любым другим протоколом, обеспечивающим надежное и достоверное соединение. При использовании протокола TCP HTTP-сервер применяет, как правило, порт 80, хотя возможно использование и других портов.

HTTP-соединение должно открываться клиентом перед каждым запросом и закрываться сервером после отправки ответа. Как клиент, так и сервер должны учитывать, что соединение может быть преждевременно закрыто либо пользователем, либо по истечении времени соединения, либо из-за сбоя системы.

Описание протокола HTTP

По синтаксису и принципам построения соединения протокол HTTP принадлежит к числу наиболее простых протоколов прикладного уровня, обсуждаемых в данной книге.

Организация запроса протокола HTTP

Итак, после установления TCP-канала между клиентом и HTTP-сервером клиент отправляет на сервер запрос. Общая структура запроса выглядит следующим образом:

```
METHOD <SP> Request-URI <SP> HTTP-Version
[General-Header]
 Date:
 Pragma:
[Request-Header]
 Authorization:
 From:
 If-Modified-Since:
 Referer:
 User-Agent:
[Entity-Header]
 Allow:
 Content-Encoding:
 Content-Length:
 Content-Type:
 Expires:
 Last-Modified:
 extension-header:
/* line */
[Entity-Body]
```

Примечание

Здесь и далее `<SP>` обозначает символ пробела, `/* line */` — обозначение пустой строки, заканчивающейся парой символов `"CR" и "LF"` — возврат каретки, перевод строки.

- Поле `METHOD` содержит идентификатор метода обращения к HTTP-серверу. Метод определяет состав и структуру заголовков запроса, вид передачи и структуру параметров запроса. Наиболее распространенными методами являются `GET` и `POST`. Иногда также используются методы `HEAD`, `DELETE`, `PUT`.
  - Метод `GET` запрашивает информацию о ресурсе, расположенном по заданному значению идентификатора URI. Как правило, запрашиваемая информация представляет собой HTML, TXT или другой текстовый файл. Если URI ассоциирован с исполняемым файлом — CGI-сервисом, процесс интерфейса CGI будет запущен, и клиенту

будут переданы результаты стандартного потока вывода этого процесса. Если ресурсу для работы необходимы параметры, они должны быть указаны в URI этого ресурса.

- Метод `POST`, как правило, используется для передачи клиентом на сервер данных, которые должны быть обработаны ресурсом, указанным в идентификаторе URI. Данный метод чаще всего используется для работы с CGI-сервисом. Метод `POST` передает параметры ресурсу, указанному в идентификаторе URI в теле сообщения, поэтому при его использовании не требуется соблюдать никаких ограничений на длину передаваемой строки параметров. Ответы на запросы методом `POST`, как правило, не кэшируются.
- Метод `HEAD` аналогичен методу `GET`, за исключением того, что клиенту возвращается только заголовок сообщения ответа ("усеченный" `GET`). Этот метод, в основном, используется для тестирования гиперссылок и проверки доступа к ресурсам.
- Метод `DELETE` используется для удаления ресурса, определенного идентификатором URI.
- Метод `PUT` используется в тех случаях, когда клиент желает сохранить передаваемый на сервер ресурс с идентификатором URI.

❒ Поле `Request-URI` содержит идентификатор URI ресурса сервера. Он может быть представлен как в абсолютном, так и в относительном форматах. Абсолютный формат содержит все части идентификатора URI: схему, адрес и путь, а относительный формат — только путь к ресурсу на текущем сервере протокола HTTP.

❒ В поле `HTTP-Version` размещается версия протокола HTTP, который клиент предполагает использовать. Идентификатор версии передается для согласования параметров работы клиента и сервера. Если версия запроса не указана, по умолчанию используется старая версия HTTP/0.9, которую обязаны поддерживать все участники HTTP-соединения. Для идентификации версии протокола HTTP использует схему "major.minor", т. е. версия HTTP/1.0 старше версии HTTP/0.9. Из-за быстрого роста возможностей протокола и, соответственно, его модификаций для стабильной работы HTTP-клиента необходимо следить за тем, чтобы версии протокола HTTP у клиента и сервера совпадали.

Proxy-посредники или шлюзы не должны передавать сообщение, которое имеет идентификатор более поздней версии протокола, чем та, которую они поддерживают. Если получен пакет, имеющий идентификатор более поздней версии протокола, идентификатор версии пакета, по возможности (например, когда пакет не содержит структур, не определенных в текущей версии), должен быть уменьшен, иначе будет отправлено уведомление об ошибке передачи.

Следовательно, первая строчка запроса может выглядеть следующим образом:

```
GET /maindoc.html HTTP/1.0
```

Далее могут следовать заголовки запроса:

- Общий заголовок `General Header`. Этот заголовок может использоваться как в запросе клиента, так и ответе сервера. Заголовок `General Header` появляется только тогда, когда передается тело сообщения (не пуст параметр `Entity-Body`), в том случае, когда клиент передает запрос методом `POST` с параметрами, или когда сервер отвечает передачей сообщения.
  - Поле `Date` содержит время построения сообщения (запроса или ответа).
  - Поле `Pragma` используется для установки специальных директив участникам соединения. Например, опция `no-cache` не позволяет промежуточным объектам соединения использовать кэш при ответе на запрос.

  Заголовок `General Header` может выглядеть, например, так:

```
Date: Tue, 15 Nov 1994 08:12:31 GMT
Pragma: no-cache
```

- Заголовок запроса `Request-Header` появляется только в запросах клиентов. Он позволяет клиентам отправлять на сервер дополнительную информацию о себе.
  - Поле `Authorization` содержит информацию аутентификации пользователя. Ответ на запрос, содержащий поле `Authorization` не кэшируется промежуточными объектами соединения.
  - Поле `From` может содержать интернет-адрес пользователя. Поле может использоваться со стороны сервера для формирования журнала заходов или, например, рассылки праздничных поздравлений наиболее активным посетителям сервера.
  - Поле `If-Modified-Since` используется при работе методом `GET`. Если запрашиваемый ресурс не изменялся с момента, указанного в этом параметре, данный ресурс не возвращается, по запросу возвращается только заголовок сообщения ответа и соответствующий код возврата. Это поле может использоваться при контроле за содержанием кэша.
  - Поле `Referer` содержит URI предшествующего ресурса. Этот параметр может использоваться сервером, например, для оптимизации кэширования, построения путей поиска и т. д.
  - Поле `User-Agent` содержит информацию о программном обеспечении клиента.

Заголовок запроса `Request-Header` может выглядеть, например, так:

```
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
From: webmaster@w3.org
If-Modified-Since: Sat, 29 Oct 1994 19:43:31 GMT
Referer: http://www.w3.org/hypertext/DataSources/Overview.html
User-Agent: CERN-LineMode/2.15 libwww/2.17b3
```

- Заголовок передаваемого сообщения `Entity-Header`. Содержит информацию о структуре и формате тела запроса или, если тело сообщения отсутствует, информацию о запрашиваемом ресурсе. Может появляться как в запросе клиента, так и ответе сервера.
  - Поле `Allow` содержит список методов, поддерживаемых ресурсом. Это поле носит исключительно информационный характер.
  - Поле `Content-Encoding` содержит идентификатор типа дополнительной кодировки ресурса (т. е. для использования этого ресурса он должен быть раскодирован указанным алгоритмом).
  - Поле `Content-Length` определяет длину тела сообщения.
  - Поле `Content-Type` содержит тип ресурса и таблицу кодировки представляемых данных (`us-ascii`, `iso8580-1`, `windows-1251`, `koi8-r` и др., в соответствии со спецификацией стандарта MIME).
  - Поле `Expires` содержит дату окончания срока действия ресурса. Это поле, например, может использоваться в механизмах кэширования как параметр запрета кэширования устаревших ресурсов. Это поле может также информировать пользователя о том, что, например, информация, предоставляемая данным ресурсом устарела.
  - Поле `Last-Modified` содержит дату и время последнего изменения ресурса.

  Заголовок `Entity-Header` может выглядеть, например, так:

```
Allow: GET, HEAD
Content-Encoding: x-gzip
Content-Length: 3495
Content-Type: text/html; charset=windows-1251
Expires: Thu, 01 Dec 1998 16:00:00 GMT
Last-Modified: Tue, 15 Nov 1996 12:45:26 GMT
```

Каждая строка заголовка заканчивается символами "CR" и "LF". Если запрос содержит сообщение, оно располагается после пустой строки — (`/* line */`), которая отделяет заголовки от тела запроса. Формат и длина тела сообщения определяются заголовком `Entity-Header` и должны удовлетворять стандартам MIME.

Помимо описанных заголовков, запрос может содержать поля `Accept`, которые предписывают отвечающему серверу использовать а ответе только указанные форматы данных. Например, если клиент понимает только текстовые данные и картинки в формате gif, запрос должен содержать следующее:

```
Accept: text/*
Accept: image/gif
```

Поля `Accept-Charset`, `Accept-Encoding`, `Accept-Language` определяют, соответственно, допустимые в ответе значения символьного набора, типов кодировок и используемого национального языка.

Кроме того, запрос может содержать поле `Content-Language`, описывающее *основные* (но не все) национальные языки, используемые при формировании тела сообщения. Например, такая строка может содержаться как в ответе сервера, так и в запросе клиента:

```
Content-Language: ru, en
```

Примечание

Поля `Accept` включены в спецификацию заголовка запроса (Request-Header) версии HTTP/1.1. Поле `Content-Language` включено в спецификацию заголовка сообщения (Entity-Header) версии HTTP/1.1. Эти поля добавлены в спецификацию HTTP/1.0 как дополнительные, для обеспечения совместимости с программным обеспечением версии "HTTP/1.1".

Далее приведены некоторые примеры построения запросов.

```
GET /big-document.html HTTP/1.0
Accept: text/html
Accept: image/gif
User-Agent: Lynx/2.2
From: monuli@kanax.edu
/* line */
```

В данном запросе пользователь обращается к текущему HTTP-серверу и запрашивает из корневого каталога документ `big-document.html`. Клиент указал версию, имя своей программы и имя почтового ящика.

```
GET /cgi-bin/exit-banner.exe HTTP/1.0
/* line */
```

В этом случае пользователь указал в идентификаторе URI исполняемый CGI-модуль. Ответом клиенту будет стандартный поток вывода данного модуля. Если вам необходимо передать параметры серверному модулю, это можно сделать двумя способами.

Используя метод GET, вы сможете передать параметры через идентификатор ресурса URI, поскольку он позволяет задавать путь объекта со строкой параметров. Например, так:

```
GET /cgi-bin/post-query?org=Academic%20Computing%20Services
&users=10000&browsers=lynx&browsers=cello&browsers=mosaic
&others=MacMosaic%20WinMosaic
&contact=Michael%20Grob%20grob@kub.uks.edu HTTP/1.0
Accept: text/html
Accept: video/mpeg
Accept: image/gif
User-Agent: Lynx/2.2
From: grob@kix.kan.edu
/* line */
```

Используя метод POST, вы сможете передать параметры в теле сообщения запроса. Например, так:

```
POST /cgi-bin/events-form HTTP/1.0
Accept: text/html
Accept: video/mpeg
Accept: image/jpeg
Accept: image/gif
User-Agent: Lynx/2.2
From: molli@kax.edu
Content-type: application/x-www-form-urlencoded
Content-length: 259
/* line */
start_month=January
&start_day=27
&start_year=1995
&end_month=May
&end_day=1
&end_year=1999
&event_type=Academic%20field
&event_type=Museum%20%26%20gallery
&event_type=Academic%20year
&event_type=Music
&event_type=Athletic
&event_type=Other%20cultural
&event_type=Parties
```

Структура и формат строки параметров те же. Теперь строка параметров передается через тело сообщения. Поля заголовка `Entity-Header` определяют длину тела сообщения и его тип. Разбор строки и обработку параметров выполняет модуль интерфейса CGI, которому они предназначаются.

Примечание

Интерфейс CGI может использовать одно и то же имя параметра с разными значениями. В нашем примере, параметр `event_type` принимает 7 различных значений.

Используя метод `PUT`, клиент может передавать ресурс на сервер. Например:

```
PUT /new-document.html HTTP/1.0
Accept: text/html
Accept: image/gif
User-Agent: Lynx/2.2
From: monuli@kanax.edu
Content-type: text/html; charset=us-ascii
Content-length: 2345
/* line */
<HTML><HEAD><TITLE>...
```

В результате в корневой каталог сервера будет записан HTML-файл с названием `new-document.html` и длиной `2345` байтов.

Структура ответа протокола HTTP

После получения запроса сервер обрабатывает его и отправляет результат обработки клиенту. Ответ состоит из следующих частей:

```
HTTP-Version <SP> Status-Code <SP> Reason-Phrase
[General-Header]
[Responce-Header]
 *Location
 Server
 WWW-Authenticate
[Entity-Header]
/* line */
[Entity-Body]
```

- `HTTP-Version` — идентификатор версии протокола HTTP, которую поддерживает сервер.
- `Status-Code` — трехзначный числовой код статуса обработки запроса. Числовые коды обработки запроса составляют иерархическую структуру.

Например, 200 — код успешной обработки запроса. Описание структуры и спецификации кодов выходит за рамки данной книги. Детальное описание кодов обработки запросов можно найти, например, в документе RFC-1945 (они вам, несомненно, пригодятся, если вы решили программировать свой HTTP-сервер или Web-браузер).

- Reason-Phrase — короткая строка описания статуса, которая, в основном, предназначена для текстового анализа результата обработки запроса.

Первая строка ответа сервера может выглядеть следующим образом:

```
HTTP/1.0 200 OK
```

Далее следуют заголовки. Состав, структура и предназначение заголовков General-Header и Entity-Header такие же, что и в пакете запроса. Ответ содержит специальный заголовок — заголовок ответа Response-Header.

- Заголовок Response-Header позволяет передавать дополнительную информацию обработки запроса, которую нельзя поместить в строку статуса.
  - Поле Location содержит полный идентификатор ресурса URI, который отвечает на отправленный запрос. Это поле — своеобразный переключатель, который перенаправляет запрос на другой сервер. Иными словами, если клиент отправил запрос на WWW-сервер с целью получения какого-либо документа, который был перемещен в другой каталог или на другой сервер, и запрашиваемый сервер отвечает Location: ..., то запрос клиента автоматически перенаправляется с помощью идентификатора URI, указанному сервером.
  - Поле Server содержит спецификацию программного обеспечения WWW-сервера, отвечающего на запрос.
  - Поле WWW-Authenticate содержит параметры схемы и пространства аутентификации.

  Заголовок Response-Header может выглядеть, например, следующим образом:

  ```
  Location: http://www.w3.org/hypertext/WWW/NewLocation.html
  ```

  или так:

  ```
  Server: CERN/3.0 libwww/2.17
  WWW-Authenticate: Basic realm="WallyWorld"
  ```

Далее приведены несколько примеров, содержащих ответы WWW-сервера на запросы клиентов. Например, ответ сервера на запрос на получение HTML-файла может выглядеть так:

```
HTTP/1.0 200 OK
Date: Wednesday, 05-Feb-97 23:04:12 GMT
Server: NCSA/1.1
MIME-version: 1.0
```

```
Last-modified: Monday, 15-Nov-96 23:33:16 GMT
Content-type: text/html
Content-length: 2345
/* line */
<HTML><HEAD><TITLE>...
```

Ответ сервера в результате работы CGI-модуля — построение динамической страницы с текущим значением даты и времени отвечающего сервера — выглядит, например, так:

```
HTTP/1.0 200 OK
Server: MacHTTP
MIME-Version: 1.0
Content-type: text/html
/* line */
<title>Server Script</title>
<h2>The time is:</h2>
September 15, 1997 3:15 pm <p><p>
```

То есть при повторном запросе этого же идентификатора URI клиенту будет возвращена уже другая страница — с другим значением времени. В случае ошибки обработки запроса ответ может выглядеть, например, так:

```
HTTP/1.0 404 Not Found
/* line */
```

Механизмы аутентификации

Протокол HTTP предоставляет возможность использования простого механизма аутентификации пользователя в процессе доступа к ресурсам WWW-сервера. Механизм построен на обмене информацией аутентификации между клиентом и сервером ресурса. Механизм позволяет выбирать схему аутентификации и уровень секретности передаваемых данных.

Далее рассмотрим наиболее распространенную схему, применяемую при аутентификации — `Basic`.

Итак, предположим, что клиент сделал запрос ресурса, который требует аутентификации доступа, например:

```
GET /secret/security.html HTTP/1.0
/* line */
```

В этом случае отвечающий сервер отправляет код ответа `401`, который означает, что требуется аутентификация, и передает в заголовке ответа поле `WWW-Authenticate`, содержащее метод аутентификации — схему или, по крайней мере, одно название области аутентификации. Например, так:

```
HTTP/1.0 401 Unauthorized
```

```
Date: Wednesday, 05-Feb-97 23:04:12 GMT
Server: NCSA/1.1
WWW-Authenticate: Basic realm="Joker"
/* line */
```

Пространство ресурсов WWW-сервера может состоять из нескольких областей аутентификации, и каждая из них может использовать свою схему аутентификации. Параметр `realm` содержит имя данной области аутентификации — символьную строку. В совокупности с идентификатором ресурса URI это имя однозначно определяет ресурс WWW в сети. В данном примере наша область — `Joker` со схемой аутентификации — `Basic`.

Эта схема подразумевает, что клиент должен передать свой идентификатор и пароль для области аутентификации, указанной в параметре `realm`. Если на сервере несколько областей, и каждая требует своих параметров аутентификации — идентификаторов и паролей пользователя, тогда, чтобы воспользоваться ресурсами какой-либо из них, клиент должен зарегистрироваться в каждой отдельно.

После того как клиент получил ответ сервера, предлагающий ему зарегистрироваться, клиент должен отправить на сервер повторный запрос этого же ресурса с полем `Authenticate` в заголовке запроса, содержащем строку с идентификатором и паролем пользователя в кодировке "base64", — это единственный вид шифрования, который использует схема `Basic`.

Например, если идентификатор пользователя "Aladdin", а пароль "open sesame", исходная строка для кодирования "base64", в формате — "user:password", будет выглядеть так: `Aladdin:open sesame`, а отправляемый пакет будет иметь вид:

```
GET /secret/security.html HTTP/1.0
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
/* line */
```

После обработки данного запроса сервер должен разрешить клиенту "Aladdin" пользоваться ресурсом "Joker". Если аутентификация по каким-либо причинам не была успешно выполнена, сервер в своем ответе возвращает код 403 (forbidden).

Схема `Basic` не предусматривает какого-либо шифрования пользовательского идентификатора и пароля при передаче их по сети, поэтому ее можно назвать очень сомнительной схемой защиты, применяемой скорее для тестов, идентификации или журнализации входов пользователей, чем для серьезной защиты ресурсов WWW-сервера.

Для обеспечения более надежного способа аутентификации в системах, требующих высокого уровня защищенности, применяется шифрование данных на уровне транспортного протокола TCP или UDP.

В не столь "сверхсекретных" WWW-серверах применяется механизм SSL (Secure Sockets Layer). Механизм SSL обеспечивает дополнительное шифрование при передаче пароля, идентификатора пользователя и данных, передаваемых по соединению. Протокол SSL представляет собой самостоятельный протокол шифрования, поверх которого может работать как протокол HTTP, так и, например, протоколы FTP и SMTP. Протокол SSL обеспечивает шифрование данных на уровне протоколов DES (Data Encryption Standard) и MD5 (Message-Digest Algorithm), в то же время он достаточно гибок для использования других методов шифрования. Более подробное обсуждение механизмов аутентификации протокола HTTP выходит за рамки данной книги.

Взаимодействие протокола HTTP и CGI-сервиса

Интерфейс CGI (Common Gateway Interface) представляет собой простейший интерфейс запуска внешних программ. Как правило, мало кто представляет себе интерфейс CGI вне работы с WWW-сервисом, хотя необходимо заметить, что CGI-спецификация существует уже давно, и только с 1993 года ее стали поддерживать HTTP-серверы, благодаря чему она и получила широкую известность. Вместе с HTTP-сервером, интерфейс-CGI отвечает за обработку клиентских запросов к исполняемым модулям и отправку результатов их работы клиенту. Если вы не собираетесь разрабатывать HTTP-сервер, то интерфейс CGI — это единственная возможность поработать с параметрами структур заголовков запросов и ответов протокола HTTP.

Итак, клиент отправил запрос с использованием идентификатора URI, ассоциированный с CGI-модулем. По получению запроса HTTP-сервер создает среду (или виртуальную машину), где будет запущен этот CGI-модуль, сохраняет часть параметров заголовков протокола HTTP в переменные окружения этой среды и запускает CGI-модуль.

Ниже представлен список наиболее употребительных и часто используемых для построения CGI-модулей параметров окружения, в которых сохраняются значения HTTP-заголовка.

- `AUTH_TYPE` — схема механизма аутентификации (если такой механизм присутствует). Копируется часть поля `Authorization` заголовка `Request-Header`.
- `CONTENT_LENGTH` — длина тела сообщения. Копируется поле `Content-Length` заголовка `Entity-Header`.
- `CONTENT_TYPE` — тип данных тела сообщения. Копируется поле `Content-Type` заголовка `Entity-Header`.
- `QUERY_STRING` — строка параметров в идентификаторе ресурса URI. Напомним, что полный идентификатор ресурса URI с параметрами имеет

вид: "scheme://host:port/ path query", где "scheme" — схема доступа, "host" — адрес хоста, "port" — номер порта, "path" — путь к ресурсу внутри хоста, "query" — строка параметров ресурса, которую и содержит переменная QUERY_STRING.

- REMOTE_ADDR — IP-адрес агента, выполнившего запрос. Это может быть как адресом клиента, так и адресом Proxy или любого другого промежуточного объекта соединения.
- REMOTE_USER — если ресурс требует аутентификации, т. е. параметр AUTH_TYPE не пуст, переменная содержит идентификатор пользователя, иначе — NULL.
- REQUEST_METHOD — метод запроса ресурса (GET, POST или др.).
- SERVER_NAME — имя запрашиваемого сервера протокола HTTP.
- SERVER_PORT — порт соединения сервера протокола HTTP.
- SERVER_PROTOCOL — версия протокола HTTP, поддерживаемого сервером.
- HTTP_* (вместо символа "*" следует указать название поля заголовка) — в зависимости от версии протокола, поддерживаемого данным сервером, может содержать значения HTTP-заголовков в том виде, в котором они были получены сервером. Например, параметр HTTP_REFERER содержит значение поля Referer заголовка Request-Header. Однако HTTP-сервер не обязан формировать такие переменные для всех передаваемых полей, например, при использовании механизма аутентификации поле Authorization не копируется. Также не копируются поля Content-Length и Content-Type, поскольку их значения уже хранятся в переменных CONTENT_LENGTH и CONTENT_TYPE соответственно.

Если вы разрабатываете интерфейс CGI, например, на языке C/C++, CGI-модуль может получить значения этих переменных через вызов функции getenv, например, так:

```
char* rm;
...
if((rm = getenv("REQUEST_METHOD")) == NULL){ /* error! */ }
```

где rm будет указывать на используемый клиентом метод обращения к ресурсу. Таким же образом вы можете получить в своей программе все доступные серверу HTTP параметры клиента.

Если клиент для запроса использует метод GET, то строка параметров управления работой CGI-модуля содержится в переменной QUERY_STRING, если же клиент использует метод POST, то переменные содержатся в теле сообщения запроса. Тело сообщения помещается HTTP-сервером в стандартный поток вывода среды работы CGI-модуля. Поэтому, чтобы полу-

чить его, достаточно знать длину потока, т. е. считать значение переменной `number = "CONTENT_LENGTH"`, и после этого вызвать стандартную функцию чтения входного потока, например, `fgets(input, number, stdin)`.

После анализа передаваемых параметров интерфейса CGI формирует ответ. Ответное сообщение CGI-модуль должен отправить в поток стандартного вывода (stdout). Как правило, интерфейс CGI использует для ответа следующий формат:

```
[CGI-Header]
 Content-Type
 Location
 Status
/* line */
[Entity-Body]
```

- Заголовок `CGI-Header` содержит параметры работы CGI-модуля.
  - Поле `Content-Type` должно обязательно присутствовать, если передается тело сообщения. Поле содержит идентификатор типа данных сообщения. Это поле аналогично одноименному полю заголовка `Entity-Header`.
  - Поле `Location` содержит идентификатор URI нового ресурса, на который интерфейс CGI перенаправляет запрос. Как правило, если присутствует это поле, другие поля заголовка не указываются. Это поле аналогично одноименному полю заголовка `Response-Header`.
  - Поле `Status` содержит код возврата обработки запроса. Интерфейс CGI может установить свой код возврата обработки запроса. Если поле не задано, возвращается код `200 OK`. Это поле аналогично полям `Status-Code` и `Reason-Phrase` строки статуса заголовка ответа.

  CGI-заголовок может выглядеть следующим образом:

  ```
  Content-Type: text/html
  Status: 777 Very Good
  ```

  или так:

  ```
  Location: http://www.server.com:443/redirector/file.html
  ```

Каждая строчка заголовка заканчивается символами "CR" и "LF". После формирования заголовка, даже если тело сообщения отсутствует, не забывайте выводить пустую строку — она определяет конец заголовка и служит разделителем между заголовком и телом сообщения. Далее приведен пример выходного потока CGI-модуля:

```
Content-Type: text/html
/* line */
```

```
<HTML>
<HEAD>Hello message</HEAD>
<BODY>
You visited our CGI module!
</BODY></HTML>
```

Расширение возможностей протокола HTTP

В 1996 г. была стандартизована и с января 1997 г. стала быстро распространяться новая версия протокола HTTP — HTTP/1.1. В этой версии значительно расширены возможности протокола HTTP при управлении соединением и участвующих в нем агентов. Опишем наиболее интересные новшества версии HTTP/1.1.

В версии HTTP/1.1 появились два новых метода — `OPTIONS` и `TRACE`, а именно:

1. Метод `OPTIONS` запрашивает характеристики соединения между клиентом и сервером, характеристики сервера, требования для запроса данного ресурса и др. При этом указанный ресурс не запрашивается и, если этим ресурсом является CGI-модуль, он не запускается. Если вместо идентификатора URI стоит символ-джокер ("`*`"), метод запрашивает всю доступную информацию о данном сервере.
2. Метод `TRACE` применяется для организации "петель". При получении такого запроса сервер должен немедленно отправить поступившую информацию обратно. Этот метод позволяет трассировать возможные соединения и получать информацию о временах прохождения и обработки запросов.

В основной заголовок `General-Header` сообщения в новой версии протокола добавлены следующие поля:

1. Поле `Cache-Control` позволяет более гибко управлять механизмами кэширования информации, устанавливать ограничения на время и объекты кэширования, управлять ограничениями на запросы из кэша определенных клиентов и др.
2. Поле `Connection` позволяет клиенту устанавливать параметры соединения с сервером. Как правило, HTTP-соединение закрывается после того, как сервер ответил на запрос клиента. Эта опция позволяет не закрывать соединение после ответа сервера или закрыть его по указанию клиента.
3. Поле `Upgrade` содержит спецификации других протоколов, по которым может быть установлено соединение с данным сервером (или клиентом), помимо протокола HTTP.

   Например: `Upgrade: HTTP/2.0, SHTTP/1.3, IRC/6.9, RTA/x11`.

4. Поле `Via` используется для журнализации промежуточных Proxy-агентов или шлюзов, через которые прошел запрос (или ответ). Это поле подобно полю `Received` в формате почтового сообщения, например: `Via: 1.0 fred, 1.1 nowhere.com (Apache/1.1)`.

Заголовок запроса `Request-Header` также был расширен новыми полями. Особый интерес могут вызвать следующие:

1. Поля семейства `Accept*` (`Accept`, `Accept-Charset`, `Accept-Language`), как было отмечено, уже фигурировали в версии HTTP/1.0 как дополнительные поля. В версии протокола HTTP/1.1 они переместились в заголовок `Request-Header` и приобрели новый параметр `q`, который принимает значения от 0 до 1 (по умолчанию `q` = 1). Этот параметр означает приоритетность (качество обработки) запрашиваемых данных указанного типа. Например, строка `Accept: text/*; q=0.8, text/html` означает, что клиент в ответ на запрос предпочитает получить html-текст, хотя может обработать и любой другой, но со снижением качества обработки на 20%.
2. Поле `Host` должно содержать место нахождения ресурса URL запрашиваемого хоста. То есть, если запрос производится по относительному пути, например, `GET /readme.html HTTP/1.1`, идентификатор URI сервера можно найти в поле `Host`.
3. Поля `If-Match`, `If-None-Match`, `If-Range`, `If-Unmodofied-Since` устанавливают различные флаги на предыдущие запросы указанного ресурса для более эффективного управления кэшем сервера и/или промежуточных агентов.
4. Поле `Proxy-Authorization` позволяет клиенту регистрироваться на хостах промежуточных агентов соединения. Это поле может служить для построения аутентификации клиента на всем пути к запрашиваемому ресурсу.
5. Поле `Range` позволяет контролировать размер передаваемого сообщения.

Заголовок ответа сервера `Response-Header` расширен такими полями, как:

1. Поле `Age`, которое содержит количество секунд, затраченное сервером на отправку ответа по данному запросу. Поле позволяет контролировать производительность различных серверов и промежуточных объектов сети.
2. Поле `Warning` содержит дополнительную информацию о статусе обработки запроса — своеобразная система подчиненных кодов коду статуса ответа.

Заголовок текста сообщения `Entity-Header` расширен полями:

1. Поля семейства `Content*`, а именно: `Content-Base`, `Content-Language`, `Content-Location`, `Content-MD5`, `Content-Range` — описывают, соответственно, формат, национальный алфавит, расположение частей ресурса

(если таковые есть), алгоритм шифрования, размер частей и всего сообщения.

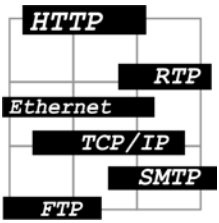
2. Поле `ETag` содержит ключевые слова данного ресурса, по которым ресурс идентифицируется как на сервере, так и в кэше промежуточных агентов.

Безусловно, представленные здесь усовершенствования протокола HTTP/1.1, по сравнению с версией протокола HTP/1.0, далеко не все и описаны очень сжато. Основное внимание следует обратить на тенденции направления развития протокола и расширяемую функциональность.

Сравнивая функции и структуру версий протокола HTTP "1.0" и "1.1", следует отметить следующее:

1. Развивается тенденция увеличения производительности протокола за счет более эффективной работы с кэшем, промежуточными агентами передачи данных и каналами соединения.
2. Намечается более тесная интеграция протокола HTTP с промежуточными агентами соединения — вплоть до аутентификации на proxy-агентах и шлюзах соединения.
3. Происходит расширение возможностей передачи распределенных ресурсов.
4. Развиваются дополнительные механизмы обеспечения защищенности передаваемых данных.

Глава 13



Протоколы передачи файлов

Протокол FTP (File Transfer Protocol, Протокол передачи файлов) — это один из первых и все еще широко используемых сервисов сети Интернет. Первые спецификации протокола FTP относятся к 1971 году. С тех пор он претерпел множество модификаций и значительно расширил свои возможности. Протокол FTP может использоваться как в программах пользователей, так и в виде специальной утилиты операционной системы.

Протокол FTP предназначен для решения следующих задач:

- доступа к файлам на удаленных хостах;
- использования ресурсов удаленных компьютеров;
- обеспечения независимости клиента от файловых систем удаленных хостов;
- эффективной и надежной передачи данных.

Протокол FTP — это протокол прикладного уровня, который использует TCP в качестве протокола транспортного уровня. Протокол FTP не может использоваться для передачи конфиденциальных данных, поскольку не обеспечивает защиты передаваемой информации и передает между сервером и клиентом открытый текст. FTP-сервер может потребовать от FTP-клиента аутентификации (т. е. при подсоединении к серверу FTP-пользователь должен будет ввести свой идентификатор и пароль). Однако и пароль, и идентификатор пользователя будут переданы от клиента на сервер открытым текстом.

Основные принципы протокола FTP

Схема взаимодействия компонентов с использованием протокола FTP представлена на рис. 13.1, где используются следующие обозначения.

- User Interface — пользовательский интерфейс работы с протоколом FTP.
- User-PI — (User — Protocol Interpretator) интерпретатор команд пользователя. Этот объект взаимодействует с модулем Server-PI для обмена командами управления передачей данных по каналу "Передача команд" и с User-DTP-модулем, который осуществляет непосредственную передачу данных по каналу "Передача данных".

- User-DTP — (User — Data Transfer Process) модуль, осуществляющий обмен данными между клиентом и FTP-сервером по каналу "Передача данных" на основании команд модуля User-PI. Этот объект взаимодействуют с файловой системой пользователя и объектом Server-DTP.
- Server-PI — (Server — Protocol Interpretator) модуль управления обменом данных со стороны сервера по каналу "Передача команд".
- Server-DTP — (Server — Data Transfer Process) модуль, осуществляющий обмен данными со стороны сервера по каналу "Передача данных".
- Сервер FTP — модуль, осуществляющий работу FTP-сервера. Он состоит из модуля управления передачей — Server-PI и модуля, осуществляющего передачу — Server-DTP.
- Пользователь FTP — модуль клиента протокола FTP. Он состоит из модуля управления передачей — User-PI и модуля, осуществляющего передачу — User-DTP.

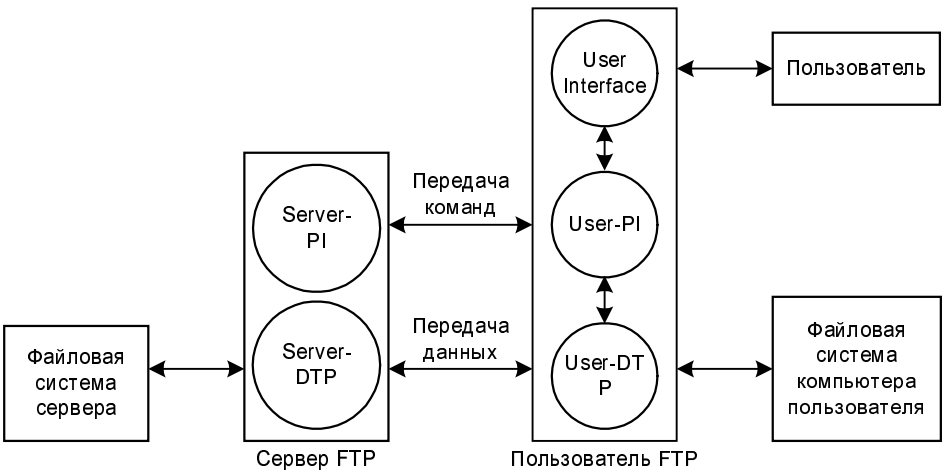


Рис. 13.1. Схема взаимодействия компонентов с использованием протокола FTP

Как видно из рисунка, протокол FTP поддерживает сразу два канала соединения — канал передачи команд и результатов их выполнения, а также канал передачи данных. Канал передачи данных может использоваться для передачи как в одном, так и в другом направлениях, кроме того, он может закрываться и открываться по командам управляющих модулей в процессе работы. Канал передачи команд открывается с установлением соединения и используется только для передачи команд и ответов их обработки.

Алгоритм протокола FTP (рис. 13.1) состоит в следующем:

1. Сервер FTP использует порт 21 протокола TCP, который всегда находится в состоянии ожидания управляющего соединения со стороны FTP-клиента.

2. После того как устанавливается управляющее соединение модуля User-PI с модулем сервера — Server-PI, клиент может отправлять на сервер команды. FTP-команды определяют параметры соединения передачи данных: роль участников соединения (активный или пассивный), порт соединения (как для User-DTP, так и для Server-DTP), тип передачи, тип передаваемых данных, структуру данных и управляющие директивы, обозначающие действия, которые пользователь хочет совершить, например, сохранить, считать, добавить или удалить данные или файл и др.
3. После того как согласованы все параметры канала передачи данных, один из участников соединения, который является пассивным (например, клиентский модуль User-DTP), становится в режим ожидания открытия соединения с использованием заданного для передачи данных порта. После этого активный модуль (например, Server-DTP) открывает соединение и начинает передачу данных.
4. После окончания передачи данных соединение между Server-DTP и User-DTP закрывается, но управляющее соединение Server-PI — User-PI остается открытым. Пользователь, не закрывая FTP-сессии, может еще раз открыть канал передачи данных, передать необходимую информацию и т. д.

Протокол FTP может использоваться не только для передачи файлов между клиентом и сервером, но и между двумя FTP-серверами, ни один из которых не расположен на локальном хосте пользователя (рис. 13.2).

Для этого пользователь сначала устанавливает управляющие соединения с двумя FTP-серверами, а затем устанавливает между ними канал передачи данных. В этом случае управляющая информация передается через модуль User-PI, но данные транслируются через канал Server1-DTP_Server2-DTP.

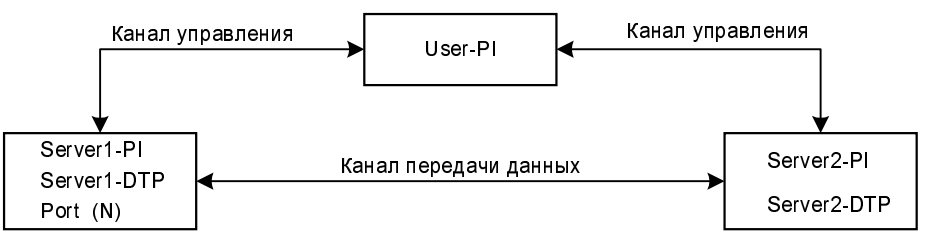


Рис. 13.2. Схема управления FTP-соединением между двумя удаленными серверами протокола FTP

Алгоритм работы схемы управления, изображенной на рис. 13.2, выглядит следующим образом:

1. Пользователь с помощью модуля User-PI указал первому серверу (Server1-PI) работать в пассивном режиме, после чего этот сервер

(Server1-PI) отправил пользователю FTP в модуль User-PI адрес и номер порта (N), который он будет слушать.

2. Пользователь с помощью модуля User-PI назначил второй сервер (Server2-PI) в качестве активного участника соединения и указал ему передавать данные на хост, то есть первый сервер (Server1-PI) в ранее указанный порт (N).
3. Модуль User-PI подал первому серверу (Server1-PI) команду "сохранить поступившие данные в некотором файле", а второму серверу (Server2-PI) — "передать содержание этого файла".
4. Между первым (Server1-DTP) и вторым (Server2-DTP) серверами образуется поток данных, который управляется клиентским хостом.

Основу передачи данных протокола FTP составляет механизм установления соединения между соответствующими портами и выбора параметров передачи. Каждый участник FTP-соединения должен поддерживать порт передачи данных по умолчанию. По умолчанию модуль User-DTP использует тот же порт, что и для передачи команд (обозначим его "U"), а модуль Server-DTP использует порт, номер которого (L – 1), где L — номер управляющего порта. Однако, как правило, участниками соединения используются порты передачи данных, выбранные для них модулем User-IP, поскольку из управляющих процессов участвующих в соединении, только модуль User-PI может изменить порты, передачи данных как у модуля User-DTP, так и у Server-DTP.

Пассивная сторона соединения до того момента, когда будет подана команда "начать передачу", должна "слушать" свой порт передачи данных. Активная сторона, подающая команду к началу передачи, определяет направление перемещения данных.

После того как соединение установлено, между модулями Server-DTP и User-DTP начинается передача. Одновременно по каналу Server-PI" — User-PI передаются уведомления о получении данных. Протокол FTP требует, чтобы управляющее соединение находилось в открытом состоянии, пока по каналу обмена данными идет передача. Сессия протокола FTP считается закрытой только после закрытия управляющего соединения.

Примечание

При передаче данных полагается считать байт состоящим из 8 бит, независимо от типов файловых и операционных систем, участвующих в процессе обмена данными.

Как правило, сервер протокола FTP отвечает за открывание и закрывание канала передачи данных. FTP-сервер должен самостоятельно закрыть канал передачи данных в следующих случаях:

1. Сервер закончил передачу данных в формате, который требует закрытия соединения.

2. Сервер получил от пользователя команду "прервать соединение".
3. Пользователь изменил параметры порта передачи данных.
4. Было закрыто управляющее соединение.
5. Возникли ошибки, которые делают дальнейшую передачу данных невозможной.

Описание команд протокола

Команды управления контролем передачи данных, которыми обмениваются модули User-PI и Server-PI, можно разделить на три большие группы.

1. Команды управления доступом к системе.
2. Команды управления потоком данных.
3. Команды FTP-сервиса.

Примечание

Не следует путать команды FTP-приложения, которое входит в комплект сетевых утилит вашей операционной системы, с командами протокола FTP.

Рассмотрим несколько наиболее популярных и характерных команд из каждой группы. Среди команд управления доступом к системе следует отметить следующие.

- `USER`. Как правило, эта команда открывает FTP-сессию между клиентом и сервером. Аргументом команды является имя (идентификатор) пользователя для работы с файловой системой. Эта команда может подаваться не только в начале, но и в середине сессии, если, например, пользователь желает изменить идентификатор, от имени которого будут проводиться действия. При этом все переменные, относящиеся к старому идентификатору, освобождаются. Если во время изменения идентификатора происходит обмен данными, обмен завершается со старым идентификатором пользователя.
- `PASS`. Данная команда подается после ввода идентификатора пользователя и, в качестве аргумента, содержит пароль пользователя. Напомним, что данные аутентификации протокола FTP передаются по сети открытым текстом, поэтому для обеспечения защищенности канала пользователю необходимо предпринимать дополнительные меры.
- `CWD`. Команда позволяет пользователю работать с различными каталогами удаленной файловой системы. Аргументом команды является строка, указывающая путь каталога удаленной файловой системы, в котором желает работать пользователь.
- `REIN`. Команда повторной инициализации. Эта команда очищает все переменные текущего пользователя и сбрасывает параметры соединения.

Если в момент подачи команды происходит передача данных, передача продолжается и завершается с прежними параметрами.

- QUIT. Команда закрывает управляющий канал. Если в момент подачи команды происходит передача данных, канал закрывается после окончания передачи.

Команды управления потоком устанавливают параметры передачи данных. Команды управления потоком могут подаваться в любом порядке, но все они должны предшествовать командам FTP-сервиса. Из команд управления потоком данных следует выделить следующие.

- PORT. Команда назначает адрес и порт хоста, который будет использоваться как активный участник соединения по каналу передачи данных. Аргументами команды являются 32-битный IP-адрес и 16-битный номер порта соединения. Эти значения разбиты на шесть 8-битных полей и представлены в десятеричном виде: h1, h2, h3, h4, p1, p2, где hN — байты адреса (от старшего к младшему), а pN — байты порта (от старшего к младшему). Например, если мы хотим использовать хост 10.145.56.1:1997, команда PORT будет выглядеть следующим образом:

```
PORT 10, 145, 56, 1, 7, 205
```

- PASV. Эта команда отправляется модулю, который будет играть пассивную роль в передаче данных, т. е. "слушать" соединение. Ответом на данную команду должна быть строка, содержащая адрес и порт хоста, находящиеся в режиме ожидания соединения в формате команды PORT — "h1, h2, h3, h4, p1, p2".

- Команды TYPE, STRU, MODE определяют, соответственно, тип передаваемых данных (ASCII, Image и др.), структуру или формат передачи данных (File, Record, Page), способ передачи (Stream, Block и др.). Использование этих команд целесообразно при организации взаимодействия в различных операционных и файловых систем взаимодействующих хостов.

Команды FTP-сервиса определяют действия, которые необходимо произвести с указанными файлами. Как правило, аргументом команд этой группы является путь к файлу. Синтаксис указанного пути должен удовлетворять требованиям формата представления пути со стороны файловой системы обработчика команды. Из команд FTP-сервиса можно выделить следующие:

- RETR. Эта команда указывает модулю Server-DTP передать копию файла, заданного параметром этой команды, модулю передачи данных на другом конце соединения.

- STOR. Команда указывает модулю Server-DTP принять данные по каналу передачи данных и сохранить их в виде файла, имя которого задано параметром этой команды. Если такой файл уже существует, он будет замещен новым, если нет, будет создан новый файл с указанным в параметре команды именем.

- Команды RNFR и RNTO должны следовать одна за другой. Первая команда содержит в качестве аргумента старое имя файла, вторая — новое. Последовательное применение этих команд переименовывает файл.
- ABOR. Команда предписывает серверу прервать выполнение предшествующей сервисной команды (например, передачу файла) и закрыть канал передачи данных.
- Команда DELE удаляет указанный файл.
- Команды MKD и RMD выполняют команды управления каталогами, соответственно, создать и удалить указанный в аргументе каталог.
- Список файлов каталога можно получить командами LIST и NLST.

Все команды протокола FTP отправляются модулем User-PI в текстовом виде — по одной команде на строке. Каждая строка команды — идентификатор и аргументы — заканчивается символами возврата каретки и перевода строки. Имя команды отделяется от аргумента символом пробела — <SP>.

Обработчик команд возвращает код обработки каждой команды, состоящий из трех цифр. Коды обработки составляют определенную иерархическую структуру и, как правило, определенная команда может возвратить только определенный набор кодов. За кодом обработки команды следует символ пробела — <SP>, затем следует текст пояснения. Например, строка успешного завершения операции выглядит следующим образом: `200 Command okay`.

Сценарий работы протокола FTP

Приведем несколько простейших примеров работы с командами протокола FTP. Обозначения: `S:` — сервер, `U:` — пользователь.

```
S: 220 Service ready for new user
U: USER Gluk
S: 331 User name okay, need password
U: PASS murmur
S: 230 User logged in, proceed
U: RETR test.txt
S: 150 File status okay; about to open data connection
```

< Идет передача файла ...>

```
S: 226 Closing data connection, file transfer successful
U: TYPE I
S: 200 Command okay
U: STOR /home/images/first.my
S: 550 Access denied
U: QUIT
```

Алгоритм передачи данных между двумя серверами протокола FTP (в соответствии со схемой, изображенной на рис. 13.2) представлен на рис. 13.3.

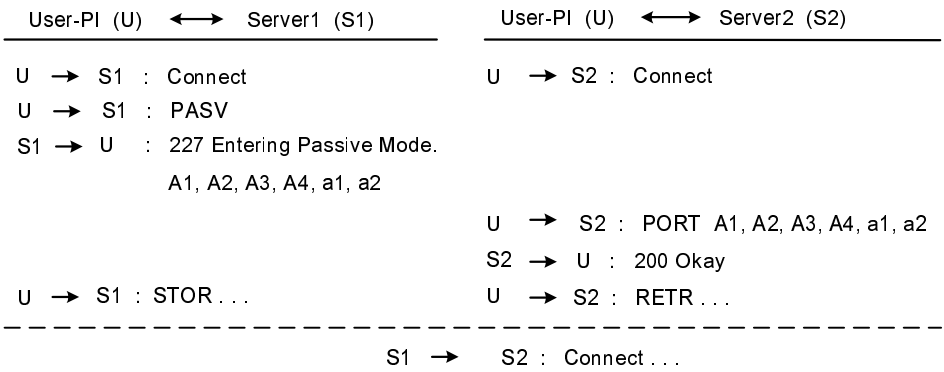


Рис. 13.3. Алгоритм передачи данных между двумя серверами протокола FTP

Протоколы TFTP и SFTP

У протокола FTP есть два "младших брата", которыми являются упрощенные протоколы передачи файлов:

- SFTP — (Simple FTP) — простой протокол передачи файлов;
- TFTP — (Trivial FTP) — простейший протокол передачи файлов.

Протокол TFTP работает поверх транспортного протокола UDP и обеспечивает выполнение только самых элементарных операций передачи файлов, а именно, записи и чтения файлов. Этот протокол не позволяет вызывать список каталога и не имеет никаких средств аутентификации, но может передавать 8-битную информацию в соответствии со всеми стандартами сети Интернет.

Поскольку передача данных осуществляется поверх протокола UDP, протокол TFTP реализует собственные методы надежной доставки данных — пакеты подтверждения, нумерация блоков данных и пакетов подтверждения и т. п. Все это выглядит очень похоже на упрощенный вариант эмуляции функций протокола TCP на прикладном уровне.

Протокол TFTP работает лишь с пятью командами:

1. `RRQ` (Read Request) — запрос на чтение.
2. `WRQ` (Write Request) — запрос на запись.
3. `DATA` (Data) — пакет данных.

4. ACK (Acknowledgment) — подтверждение.
5. ERROR (Error) — ошибка.

Процесс передачи данных начинается с поступления от клиента протокола TFTP на сервер запроса на чтение или запись файла. Соединение устанавливается после получения подтверждения готовности на один из запросов, либо на запись — WRQ, либо на чтение — RRQ.

Примечание

При открытии соединения каждая из сторон выбирает (случайным образом) уникальный идентификатор — TID, который используется в протоколе UDP как порт соединения. Каждый пересылаемый пакет ассоциирован с двумя идентификаторами TID, соответствующими каждой стороне соединения. Первоначальный запрос отправляется инициатором TFTP-соединения в UDP-порт 69 (порт инициализации), в котором указывается порт соединения. Дальнейший обмен уже происходит через порты, выбранные участниками передачи данных.

Если сервер разрешает запрос, соединение открывается, и указанный файл передается блоками по 512 байтов. Каждый пакет передаваемых данных содержит один блок (512 байтов) передаваемых данных и порядковый номер блока в передаваемом потоке. Поступление каждого блока на хост назначения должно быть подтверждено пакетом с командой ACK (подтверждение), с номером поступившего блока. Только после получения пакета подтверждения будет отправлен следующий пакет данных.

В том случае, если длина пакета составляет менее 512 байтов, это служит сигналом для закрытия соединения. Если пакет был потерян в процессе передачи, через некоторый промежуток времени сервер должен будет отправить этот пакет данных повторно.

Три типа ситуаций порождают отправку протоколом TFTP сообщений об ошибках:

1. Не подтвержден запрос, например, не был найден файл, нет прав доступа и др.
2. Неправильный формат пакета, например, произошла ошибка коммутации и др.
3. Потеря доступа к требуемому ресурсу.

При большом количестве сообщений об ошибках соединение может быть закрыто по инициативе любой из сторон.

Ниже приведен порядок действий при обработке TFTP-транзакции:

1. Хост A с помощью команды WRQ отправляет запрос хосту B. Порт источника — TIDA, порт назначения — 69. Пакет содержит имя файла, тип передачи (ASCII, 8 бит или др.).

2. Хост B отправляет команду ACK (номер блока — 0) хосту A. Порт источника — TIDB, порт назначения — TIDA.
3. Хост A отправляет (по соединению TIDA — TIDB) команду DATA и блок данных. Пакет также содержит номер блока.

Протокол передачи файлов SFTP целесообразно использовать в тех случаях, когда требуется чуть более гибкий и надежный протокол, чем TFTP, но не такой сложный и громоздкий, как протокол FTP.

Протокол SFTP обеспечивает следующие режимы и функции:

- идентификация пользователя;
- передача файлов;
- просмотр каталогов;
- изменение текущего каталога;
- переименование и удаление файлов.

В большинстве случаев приведенных выше режимов и функций оказывается вполне достаточно для организации полнофункционального обмена с удаленным FTP-сервером. Протокол SFTP может передавать 8-битный поток данных и использует, как и протокол TFTP, только одно соединение для передачи управляющих команд и данных. В отличие от TFTP, протокол SFTP работает поверх протокола TCP, порт 115.

Команды протокола SFTP отправляются поочередно, после получения ответа обработки предшествующей команды. Все команды состоят из четырех ASCII-символов и символа пробела, который отделяет команду от аргументов. Ответ сервера состоит из кода ответа и текстового сообщения. Каждая команда и ответ должны заканчиваться символом "NULL" ('\0' — то же, что и завершение строки в языке программирования C). Протокол SFTP для управления передачей данных использует всего 11 основных команд и только 4 кода ответа (символы: "+" — успешно, "-" — ошибки, "!" — внимание, " " — другое).

Команды протокола SFTP имеют почти тот же синтаксис и предназначение, что и аналогичные команды протокола FTP.

В представленном далее сценарии работы протокола SFTP SFTP-сервер с именем MIT-XX находится в состоянии ожидания соединения. Пользователь "MKL" (пароль "foo") подсоединился к SFTP-серверу, запросил список файлов стандартного формата из каталога PS — командой LIST F PS:, затем запросил список файлов в расширенном формате из того же каталога командой LIST V PS:. Далее пользователь построил запрос на файл с именем Small.File — команда RETR SMALL.FILE. В ответ сервер вернул его размер — 69 и подготовил к отправке. После этого командой SEND пользователь считал файл. По команде DONE соединение закрылось.

Введенные обозначения: `S:` — SFTP-сервер, `U:` — пользователь протокола SFTP.

```
S: +MIT-XX SFTP Service
U: USER MKL
S: +MKL ok, send password
U: PASS foo
S: ! MKL logged in
U: LIST F PS:
S: +PS:
    Small.File
U: LIST V
S: +PS:
    Small.File  1       69(7)  P775240  2-Aug-94 20:08  MKL
U: RETR SMALL.FILE
S:  69
U: SEND
S: This is a small file, the file is sent without a terminating null.
U: DONE
S: +MIT-XX closing connection
```

Как видно из представленного сценария, протокол SFTP является простым протоколом, построенным по тривиальной схеме "запрос-ответ" с минимальным набором команд, который позволяет выполнять все необходимые операции управления передачей и размещения данных на диске.

Протокол SFTP вполне удовлетворяет работе с одним пользователем и одним сервером, кроме того, его легко использовать и программировать. Более подробное описание анатомии и работы протокола SFTP вы сможете найти, например, в документ RFC-913.

Утилита FTP

Неискушенному пользователю сочетание букв FTP больше известно как название сервисной утилиты, которая позволяет подсоединяться к серверам протокола FTP и обмениваться различными файлами между FTP-серверами и компьютерами клиентов.

Некоторые из этих утилит используют сервис командной строки, некоторые выполнены с графическим интерфейсом, но, как правило, все они поддерживают стандарт протокола FTP и определяют основной набор команд, которые должны поддерживаться всеми вариантами реализации

FTP-сервиса. Возможности и функции утилиты, которой вы пользуетесь, зависят от предназначения и сетевого оснащения вашей системы.

Обычно утилита FTP предоставляет пользователю оболочку с интерфейсом командной строки. В зависимости от реализации и версии вашей программы FTP, она может поддерживать те или иные функции работы с удаленной файловой системой. Команды утилиты FTP, как правило, построены по принципу работы с локальной файловой системой и весьма отличаются от команд FTP-протокола.

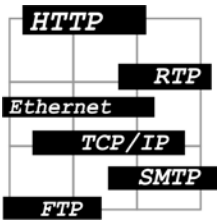
По сути, FTP-утилита представляет собой интерпретатор команд оболочки в команды (или последовательности команд) FTP-протокола. Отправка команд FTP-соединения, анализ ответов сервера протокола FTP, открытие канала передачи данных, процесс передачи данных и закрытие канала передачи данных — все это происходит внутри FTP-модуля. Обо всех событиях передачи команд и данных пользователь информируется в соответствии с интерпретацией этих событий FTP-утилитой.

В качестве примера представлен простейший сценарий передачи текстового файла с FTP-сервера на локальный хост:

```
C:\>ftp
ftp> open ftp.sun.com
Connected to ftp.sun.com.
220-Welcome to Sun Microsystems Corporate FTP Server.
220-
220 ftp FTP server (ftpd Wed Oct 30 23:31:06 PST 1996) ready.
User (ftp.sun.com:(none)): ftp
331 Guest login ok, send your complete e-mail address as password.
Password:
230 Guest login ok, access restrictions apply.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 34
dr-xr-xr-x   8 root     0            512 Feb 14  1997.
d--x--x--x   3 root     users        512 Oct 31  1996 etc
drwxrwx-wt   3 root     42          5120 Nov 21 10:37 incoming
drwxrwxr-x  12 root     42          4096 Nov 19 23:26 pub
dr-xr-xr-x   4 root     users        512 Jun  7  1996 usr
-rw-r--r--   1 root     users         49 Sep 27  1996 welcome.msg
226 Transfer complete.
```

```
624 bytes received in 19,66 seconds (0,03 Kbytes/sec)
ftp> get welcome.msg primer.txt
200 PORT command successful.
150 Opening ASCII mode data connection for welcome.msg (49 bytes).
226 Transfer complete.
50 bytes received in 2,73 seconds (0,02 Kbytes/sec)
ftp> quit
221 Goodbye.
```

Глава 14



Сетевая файловая система — NFS

Протокол NFS (Network File System, Сетевая файловая система) позволяет монтировать в единое целое файловые системы нескольких, возможно удаленных друг от друга, машин и предоставлять удаленный доступ к файлам каждой из них.

Протокол NFS, как правило, реализуется в ядре операционной системы компьютера, поэтому все его приложения получают возможность работать с удаленными файлами так же, как они работают с локальными.

Функционирование NFS-системы базируется на протоколе NFS, который предназначен для предоставления универсального интерфейса работы с файлами для различных типов компьютеров, операционных систем, сетевых архитектур и транспортных протоколов.

Универсальность и переносимость NFS достигается за счет использования двух протоколов.

- RPC (Remote Procedure Call) — протокол удаленных процедур, который служит основой построения сетевой файловой системы.
- XDR (eXternal Data Representation) — протокол универсального представления данных

NFS-сервер использует набор процедур протокола RPC, который предоставляет процедурно-ориентированный интерфейс для обращения к удаленным сервисам сети. Универсальность системы NFS состоит в том, что она не требует дополнительного сервиса от протоколов транспортного уровня и может взаимодействовать с любой реализацией протокола RPC. Иными словами — протокол RPC является для протокола NFS "функциональным" протоколом, который обеспечивает работу протокола NFS над транспортным уровнем, т. е. передает вызовы функций между объектами NFS-сети.

Протокол XDR обеспечивает механизмы стандартного представления данных для прикладного протокола. NFS-протокол использует механизмы форматирования протокола XDR для обращения к RPC-сервису и обмена данными между NFS-серверами и клиентами сетей различных топологий, находящихся под управлением разных операционных систем.

Особенностью протокола NFS является то, что он проектировался как протокол с "нефиксированным состоянием". Это означает, что серверу обслуживания протокола не нужно определять состояние его клиентов для выполнения функций передачи данных. Например, для того чтобы обратиться к серверу, клиенту достаточно отправить запрос на отправку данных и дождаться ответа или завершить работу по тайм-ауту. В случае возникновения ошибки клиенту не важно, по какой причине это произошло — из-за сбоя работы сети, аппаратуры или программного обеспечения.

Кроме того, в основу архитектуры протокола NFS положен принцип максимального "единообразия" всех операций, т. е. идентичность способов и методов работы с функциями и структурами данных протокола.

Принципы построения протокола

Протокол NFS представляет собой набор процедур протокола RPC. Аргументы и результаты работы этих процедур представляются в формате протокола XDR. Поскольку протокол NFS разрабатывался для UNIX-систем, большинство операций имеют семантику файловых систем UNIX.

Файловая система

Протокол NFS предполагает, что файловая система, с которой он работает, построена по иерархическому принципу. Каждый объект каталога, будь то файл, подкаталог или устройство, имеет строковое имя. Разнообразные операционные системы могут иметь различные ограничения на глубину вложенности дерева каталогов или длину имени, так же как и различное представление пути файла и составляющих его компонент (каталогов, подкаталогов и имени) — для работы протокола NFS это не имеет значения.

Файловая система хоста представляется для протокола NFS в виде "дерева" каталога, которое размещено на сервере (на одном диске или на его части) с заданным "корнем". Некоторые операционные системы представляют каталоги сервера в виде единого "дерева", другие же предпочитают работать с "лесом" таких "деревьев". Протокол NFS может работать с любой организацией файловой системы. Отдельный файл для NFS-системы представляет собой именованную область потока байтов.

Протокол NFS обрабатывает только по одному компоненту пути файла. Это делается, во-первых, потому, что в различных системах в качестве знака разделения компонент пути используются разные символы и обрабатывать весь путь целиком все равно невозможно, во-вторых, хотя файлы и каталоги — подобные объекты, для чтения файлов и каталогов используются разные функции — это необходимое требование сетевого стандарта представления каталогов.

Протокол NFS проектировался с таким расчетом, чтобы его серверы работали настолько быстро, насколько им позволяет их файловая система. Однако серверы и клиенты протокола NFS сильно взаимосвязаны, и если клиентская файловая система чересчур усложнена, это может вызвать проблемы в работе NFS-сервера. Например, некоторые операционные системы позволяют удалять открытые файлы, т. е. процесс может открыть файл и, пока он находится в открытом состоянии, удалить его из каталога. В данном случае этот файл будет доступен для чтения и записи, пока процесс держит его открытым, даже если у него в файловой системе уже нет имени. Протокол NFS не поддерживает такие способы работы для обеспечения выполнения правила синхронности и согласованности операций, которому подчиняются все процедуры протокола NFS.

Каждый NFS-клиент потенциально может работать и как NFS-сервер. Удаленная и локальная файловые системы могут свободно перемешиваться на компьютере клиента. Это может иногда приводить к некоторым интересным проблемам. Например, сервер имеет каталог `/usr` и он смонтировал файловую систему другого сервера в подкаталог `/usr/src`. Теперь допустим, что пользователь смонтировал к себе каталог этого сервера `/usr`, двигается по "дереву" каталога файловой системы данного сервера и достигает точки, где смонтирована файловая система другого сервера. Сможет ли он в этой точке перебраться в файловую систему другого сервера?

Здесь срабатывает так называемое правило "достаточности", которое предназначено упростить и ускорить работу протокола NFS. Это правило предписывает каждому участнику распределенной файловой системы следить только за состоянием собственных связей и не позволяет использовать чужие связи. В противном случае перемещение потребовало бы целой серии сетевых вызовов, регистрационных сообщений и др. Таким образом, клиент не увидит каталогов смонтированной системы `/usr/src` второго сервера до тех пор, пока не смонтирует эти каталоги самостоятельно.

Механизм аутентификации

Строго говоря, протокол NFS не предусматривает проверку прав доступа к файловой системе. Протокол NFS предполагает, что проверку прав доступа выполняет операционная система сервера тиражируемой файловой системы. Протокол NFS обеспечивает только передачу необходимых параметров аутентификации между участниками соединения.

Примечание

RPC-протокол, который используется протоколом NFS, позволяет применять различные типы аутентификации и предоставляет протоколу NFS для передачи параметров аутентификации целый блок переменных.

Необходимо заметить, что сервисы с "фиксированным" состоянием (например, разнообразные NFS-клиенты) и сервисы с "нефиксированным" состоянием (серверы протокола NFS) по-разному взаимодействуют с процедурами доступа. Большинство сервисов "фиксированного" состояния проверяют права доступа, например, для операций чтения или записи только при открытии файла. Сервисы "нефиксированного" состояния проверяют права при каждой операции чтения/записи. Поэтому в локальной системе пользователь может открыть файл и изменить флаг доступа, после чего будет продолжать операции с этим файлом в соответствии с параметрами его открытия. При работе с удаленной файловой системой, чтобы устранить такое несимметричное поведение, серверы протокола NFS применяют специальный алгоритм разграничения прав доступа в соответствии с правилами работы файловой системы NFS-сервера.

Описание протокола

Протокол NFS использует транспортные протоколы UDP, порт 2049 (NFSv2), или TCP (NFSv3) — порт 2049. Конкретная реализация NFS-сервиса, однако, во многом зависит от спецификации протокола RPC используемой операционной системы, в частности, функции протокола RPC по обеспечению транспорта — RPC_portmapping, назначающей тот или иной порт для использования передачи вызовов NFS-функций.

Все процедуры протокола NFS должны выполняться в синхронном режиме. Поэтому, если клиент получил ответ о выполнении какого-либо действия, это значит, что операция полностью завершена и все данные, перемещаемые в этой операции, записаны на жесткий носитель и находятся в устойчивом состоянии. Например, после того, как клиент получил ответ о завершении команды `WRITE`, блоки данных файловой системы, блоки информации структуры файловой системы, связи этого файла и другие параметры, записаны на жесткий диск и, если в этот момент, скажем, отключится питание, никакая информация не пропадет. Эта особенность протокола NFS очень важна для обеспечения устойчивости работы сервера и системы в целом.

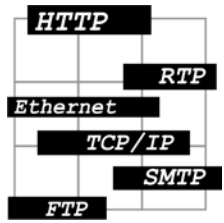
Как уже было отмечено, протокол NFS — это набор вызовов для выполнения операций с файлами. Например, клиент хочет прочитать содержимое файла, расположенного на сервере в определенном каталоге. Для этого ему нужно организовать следующую последовательность вызовов функций NFS:

1. Вызвать процедуру `LOOKUP` с параметрами имени файла и дескриптора каталога. Эта функция возвратит дескриптор файла, атрибуты файла и атрибуты каталога.

2. Для проверки доступа на чтение к файлу можно вызвать функцию `ACCESS`. Эта функция в качестве параметров использует дескриптор объекта и тип доступа, и возвращает логическую переменную (да/нет).
3. Вызвать функцию `READ`, которая считает блок данных файла. Аргументами функции являются дескриптор файла, смещение и размер считываемых данных. Функция возвращает атрибуты файла по завершению операции, количество прочитанных байтов, идентификатор завершения чтения и поток считанных данных.

Как видно из представленного примера, алгоритмы работы с удаленной файловой системой почти идентичны алгоритмам работы с локальной файловой системой.

Глава 15



Направления дальнейшего развития протокола IP

Давайте представим большой растущий город с множеством домов и роскошных апартаментов. А теперь представьте себе, что ввиду отсутствия новых телефонных номеров обитатели этих домов не смогут пользоваться телефонными услугами из-за нехватки индексных цифр. Иными словами, поселиться здесь может любой, но на полноценное общение с внешним миром рассчитывать не придется.

Несмотря на неправдоподобность этого сценария, возникновение подобной ситуации в сети Интернет вполне возможно уже в самом ближайшем будущем. Согласно некоторым расчетам, последний доступный IP-адрес будет занят где-то между 2005 и 2010 годами. К счастью, организация IETF (Internet Engineering Task Force) успешно решает эту и другие проблемы текущей четвертой версии протокола сети Интернет (IPv4) с помощью межсетевого протокола следующего поколения IPng (Internet Protocol next generation), так же известного, как Internet Protocol version 6 (IPv6).

В июне 1992 г. сообщество сети Интернет выдвинуло три предложения относительно нового протокола IP. К декабрю того же года организация IESG (Internet Engineering Steering Group), подкомитет IETF, организовал директорат, так и названный IPng Directorate, для рассмотрения новых предложений относительно протокола IP и подготовки рекомендаций. "Рекомендации по протоколу IP следующего поколения", содержащиеся в документе RFC-1752, были опубликованы в январе 1995 года. Определенный организацией IESG новый протокол сетевого уровня сети Интернет стал в конце концов упоминаться в качестве версии номер 6 протокола IP.

Протокол IP версии 6 (IPv6) — это новая версия сетевого протокола Интернета, предназначенная для постепенной замены предыдущей, четвертой версии (IPv4).

Предпосылки и направления модернизации TCP/IP

Шестая версия предполагает развитие протокола сети Интернет по следующим направлениям:

- расширение адресного пространства;

- модернизация формата заголовка пакета;
- обеспечение возможности дальнейшего расширения;
- изменение концепции обеспечения качества обслуживания.

Одной из основных проблем сегодняшней сети Интернет, несмотря на непрекращающиеся попытки ее преодоления, была и остается проблема нехватки сетевых адресов. Как уже было отмечено, возникновение этой проблемы вызвано неудачным выбором формата сетевого адреса Интернет, который принципиально не позволяет обеспечить однозначную адресацию всех узлов глобальной сети.

Сегодня эта проблема усугубляется в связи с лавинообразным ростом числа пользователей сети и недостаточно продуманной, особенно на начальной стадии развития сети Интернет, процедурой назначения групп адресов различным организациям. Например, в настоящее время существует неоправданно много сетей класса A и B, а ведь на их долю приходится значительная часть адресного пространства.

Протокол IPv6 решает потенциальную проблему нехватки IP-адресов путем использования 128-разрядных адресов вместо 32-разрядных адресов, используемых в протоколе IPv4, благодаря чему адресное пространство сети расширяется в 2^{96} раз. Результатом этого изменения будет то, что любой житель Земли может получить в свое распоряжение несколько IP-адресов.

Новый формат сетевого адреса протокола IPv6 обеспечивает возможность построения адресной иерархии с большим количеством уровней. В новой версии протокола слиты воедино такие функционально близкие режимы адресации, как Multicast и Broadcast. В протоколе IPv6 режим адресации Broadcast как таковой отсутствует, при необходимости вместо него может быть использован режим Multicast для всех узлов конкретной сети.

Существенной переработке подверглась структура IP-пакета, причем изменения затронули не только форматы сетевых адресов. В протоколе IPv6 используется принципиально новая концепция заголовка пакета. В противоположность фиксированному заголовку в протоколе IPv4, заголовок пакета протокола IPv6 имеет переменную и расширяемую структуру. Реализация концепции "вложенных заголовков" позволила существенно упростить процесс обработки пакетов на промежуточных узлах сети и одновременно придать протоколу IP новые функции и возможность их дальнейшего расширения. Некоторые поля заголовка протокола IPv4 были удалены в силу ненужности или из-за невозможности их эффективного использования. Вместо этих полей комитет организации IETF ввел в состав заголовка протокола IPv6 несколько новых полей. К этим новинкам можно отнести следующие поля и заголовки: поле идентификации пакетов, требующих специальной потоковой обработки, заголовки расширения для выполнения операций шифрования и фрагментации, а также заголовок, обеспечивающий выполнение маршрутизации от источника. Новый формат заголовка

пакета протокола IPv6 позволяет значительно более эффективно использовать опции пересылки дейтаграмм по указанному маршруту и предоставляет существенно больше возможностей для внесения изменений в опции и добавления новых.

Технология вложенных заголовков

Заголовок пакета протокола IPv6 состоит из основного постоянного заголовка, к которому при необходимости по принципу цепочки добавляются дополнительные заголовки расширения. Каждый дополнительный заголовок протокола IPv6 содержит тип следующего за ним заголовка, благодаря чему и создается цепочка заголовков. В заголовках расширения указываются дополнительные функции, которые должны быть выполнены в процессе или по завершении передачи данного пакета по сети. Таким образом, информация, необходимая для настройки протоколов сетевого уровня, которая носит вспомогательный характер, размещается в дополнительных заголовках между основным заголовком IPv6 и заголовком протокола транспортного уровня (например, протокола TCP). Эти заголовки представляют собой блоки данных, каждый из которых отвечает за выполнение конкретных независимых функций. В каждом из этих заголовков содержится поле идентификатора следующего заголовка.

На рис. 15.1 представлена структура IPv6-пакета, не имеющего дополнительных заголовков. За основным заголовком протокола IPv6 в данном примере непосредственно следует заголовок протокола TCP.

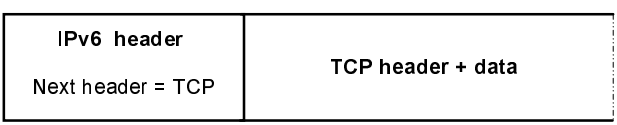


Рис. 15.1. Структура пакета IPv6, не имеющего дополнительных заголовков.

На рис. 15.2 представлена схема пакета протокола IPv6 с одним дополнительным заголовком пакета, определяющим маршрутизацию пакета от источника.

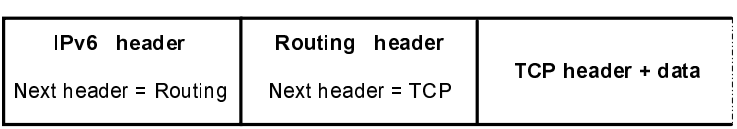


Рис. 15.2. Схема пакета протокола IPv6 с одним дополнительным заголовком пакета

На рис. 15.3 представлена схема пакета протокола IPv6 с двумя дополнительными заголовками, за которыми идет структура TCP.

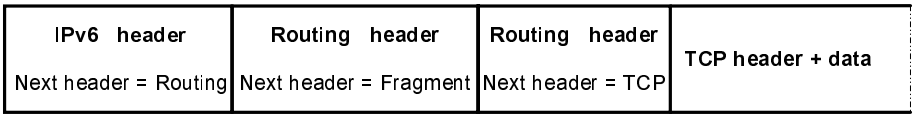


Рис. 15.3. Схема пакета протокола IPv6 с двумя дополнительными заголовками

Механизм вложенных заголовков функционирует следующим образом: сначала обрабатывается основной заголовок протокола IPv6, в котором может быть помещено указание на наличие последующего вложенного заголовка (Next Header). В свою очередь, этот следующий заголовок должен указывать на следующий за ним заголовок — и так далее по всей цепочке. Вложенные заголовки, как правило, обрабатываются строго в соответствии с их порядком размещения в цепочке, не анализируются и не обрабатываются по пути следования пакета до получателя (или группы получателей), адрес которого указан в поле адреса назначения основного заголовка.

Из этого правила обработки дополнительных заголовков протокола IPv6 есть, однако, исключение, касающееся заголовка Hop-by-Hop Options. Этот заголовок расширения содержит информацию, которая должна быть обработана на каждом узле сети, через который проходит пакет. Если заголовок расширения Hop-by-Hop Options присутствует в пакете протокола IPv6, он должен следовать непосредственно за основным заголовком.

В табл. 15.1 перечислены типы и назначение используемых в настоящее время заголовков расширения протокола IPv6.

Таблица 15.1. *Типы и назначение заголовков расширения протокола IPv6*

Значение поля Next Header	Тип заголовка расширения	Назначение
0	Hop-by-Hop Options Header	Заголовок опций промежуточных узлов
43	Routing Header	Заголовок маршрутизации
44	Fragment Header	Заголовок фрагментации
60	Destination Options Header	Заголовок опций получателя
50	Authentication Header	Заголовок аутентификации
51	Encapsulating Security Payload Header	Заголовок шифрования полезной нагрузки
59	No Next Header	Отсутствие заголовка расширения

Для повышения эффективности использования вычислительных ресурсов маршрутизаторов, обрабатывающих пакеты протокола IPv6, спецификация этого протокола определяет рекомендуемый порядок следования заголовков в пакете:

1. IPv6 Header — основной заголовок пакета.
2. Hop-by-Hop Options Header — заголовок опций промежуточных узлов.
3. Destination Options Header — заголовок опций получателя (1).
4. Routing Header — заголовок маршрутизации.
5. Fragment Header — заголовок фрагментации.
6. Authentication Header — заголовок аутентификации.
7. Encapsulating Security Payload Header — заголовок шифрования полезной нагрузки.
8. Destination Options Header — заголовок опций получателя (2).
9. Upper Layer Header — заголовок следующего протокола.

Заголовок опций получателя (1) обрабатывается получателем, адрес которого указан в поле адреса назначения основного заголовка, и получателями, перечисленными в заголовке маршрутизации. Заголовок опций получателя (2) обрабатывается только на узле назначения (в скобках указан номер заголовка опций получателя).

Примечание

Может случиться так, что заголовком следующего протокола снова окажется заголовок протокола IPv6 (например, так будет в том случае, если один IPv6-пакет инкапсулируется в другой при организации тоннеля). В этом случае на последующем уровне может быть организована своя последовательность заголовков, следующих друг за другом в аналогичном порядке. Сетевые модули протокола IPv6 должны уметь обрабатывать подобные иерархические цепи заголовков.

Основной заголовок протокола IPv6

Структура основного заголовка протокола IPv6 приведена на рис. 15.4.

В поле **Version**, занимающем 4 бита в основном заголовке протокола IPv6, помещается номер версии протокола. В рассматриваемом случае значение этого поля равно 6.

Поле **Traffic Class** занимает 8 бит в основном заголовке протокола IPv6. Значение этого поля определяет относительный уровень приоритета, установленный для содержимого поля полезной нагрузки данного пакета. Поле **Traffic Class** позволяет отправителю назначать пакету определенный уровень приоритета по отношению к пакетам, формируемым другими приложения-

ми, что, в свою очередь, обеспечивает возможность реализации концепции раздельного обслуживания (DiffServ). Поскольку значение, помещенное в это поле, может быть изменено в процессе перемещения пакета по сети, нельзя ожидать, что получатель примет пакет, имеющий точно такое же значение поля **Traffic Class**, какое было сформировано при передаче. По умолчанию значение поля **Traffic Class** устанавливается равным 0.

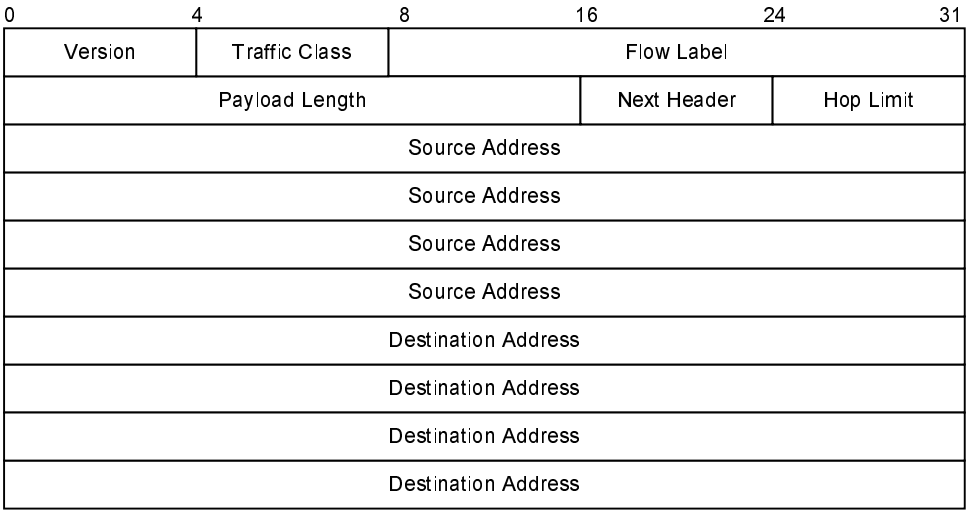


Рис. 15.4. Структура основного заголовка протокола IPv6

Поле **Flow Label** занимает 20 бит в основном заголовке протокола IPv6 и содержит значение метки потока. Это поле может использоваться отправителем для того, чтобы помечать пакеты, требующие специальной единообразной обработки сетевыми модулями протокола IPv6.

Поток представляет собой последовательность пакетов, отправляемых определенному получателю (или группе получателей), на пути к которым пакеты должны пройти специальную обработку. Поскольку между одними и теми же узлами может быть организовано несколько потоков, то и значение поля **Flow Label** позволяет идентифицировать конкретный поток в группе потоков, соединяющих два узла сети. Если значение этого поля установлено в 0, то считается, что дейтаграмма не принадлежит ни к какому потоку. Все пакеты, принадлежащие к одному потоку, должны отправляться по одному и тому же адресу назначения и с одним и тем же приоритетом.

Меткой потока служит выбираемое случайным образом число в диапазоне 1—FFFFFF. Хосты или шлюзы, не поддерживающие функции потоковой обработки, должны при формировании очередного пакета устанавливать значение этого поля равным нулю и игнорировать содержание этого поля при обработке получаемого пакета.

Поле **Payload Length** занимает 16 бит и содержит выраженное в байтах значение длины поля данных, следующего за основным заголовком этого пакета.

Поле **Payload Length** в заголовке протокола IPv6 имеет назначение, аналогичное тому, что имеет поле **Total Length** в заголовке протокола IPv4. Эти два поля сходны между собой, но не тождественны, поскольку поле **Payload Length** содержит только длину поля данных, размещенного после основного заголовка, в то время как поле **Total Length** учитывает длину заголовка.

В поле **Next Header**, которое занимает 8 бит в основном заголовке протокола IPv6, содержится информация о типе заголовка расширения, который следует за основным заголовком IPv6.

Поле **Next Header** в заголовке протокола IPv6 имеет назначение, аналогичное тому, что имеет поле **Protocol** в заголовке протокола IPv4. Однако в заголовке протокола IPv4 значение поля **Protocol** определяет только, данные какого транспортного протокола следуют за IP-заголовком. Ниже будет показано, каким образом в протоколе IPv6 поле **Next Header** позволяет использовать дополнительные заголовки расширения протоколов сетевого и транспортного уровней. Поскольку поле **Next Header** предоставляет информацию о наличии дополнительных заголовков, следующих за основным, в протоколе IPv6 отсутствует необходимость формирования поля, аналогичного по назначению полю **Internet Header Length** протокола IPv4.

Поле **Hop Limit** занимает 8 бит в заголовке протокола IPv6. Это поле в заголовке протокола IPv6 имеет назначение, аналогичное тому, что имеет поле **Time to Live** в заголовке протокола IPv4. Значение, размещенное в поле **Hop Limit**, уменьшается на 1 при прохождении пакетом очередного узла сети. Если значение этого поля становится равным 0, соответствующий пакет IPv6 подлежит уничтожению.

Можно вспомнить, что в протоколе IPv4 для определения периода существования пакета в сети использовались единицы времени — секунды. Однако это решение оказалось неудачным, поскольку при прохождении пакета через маршрутизатор (ввиду трудности определения времени ожидания в очереди) значение поля **Time to Live** всегда уменьшалось на 1 секунду. Признав эффективность использования числа пройденных узлов сети в качестве способа определения срока жизни пакета, разработчики новой версии протокола заменили в IPv6 поле **Time to Live** на поле **Hop Limit**.

Поле **Source Address** занимает 128 бит в основном заголовке протокола IPv6 и содержит адрес отправителя данного пакета.

Поле **Destination Address** также занимает 128 бит в основном заголовке протокола IPv6 и может содержать адрес получателя данного пакета.

Примечание

Поле **Destination Address** может и не содержать конечный адрес назначения. Если в заголовке присутствует вложенный заголовок маршрутизации (Routing

Header), в этом поле размещается адрес ближайшего промежуточного узла маршрута.

Сетевые адреса протокола IPv6

Как упоминалось выше, сетевые адреса протокола IPv6 имеют длину 128 бит. По словам председателя совета по архитектуре сети Интернет, такое количество адресов позволит подключить к сети Интернет свыше 1 квадрильона компьютеров в 1 триллионе сетей, т. е. практически любое устройство — от холодильников, микроволновых печей и кофеварок до видеокамер, сотовых телефонов и т. д.

В протоколе IPv6 использован тот же принцип построения сетевого адреса, что и в предыдущей, четвертой версии протокола: сетевой адрес состоит из сетевого префикса и адреса узла. Существенное отличие состоит в том, что в протоколе IPv6, благодаря большему размеру сетевого адреса, в качестве адреса узла сети может быть использован MAC-адрес интерфейса, что делает ненужным выполнение процедуры протокола ARP (Address Resolution Protocol).

В последующих разделах описаны типы и назначение сетевых адресов протокола IPv6.

Категории сетевых адресов

Сетевые адреса протокола IPv6 присваиваются интерфейсам узлов сети Интернет и подразделяются на три основных категории:

- Unicast — индивидуальные адреса;
- Multicast — групповые адреса;
- Anycast — кластерные адреса.

Индивидуальный адрес протокола IPv6 определяет один интерфейс подключения к сети этого протокола. Пакет, адресованный на Unicast-адрес протокола IPv6, должен быть доставлен только на указанный интерфейс.

Групповой адрес протокола IPv6 определяет группу интерфейсов. Пакет, адресованный на Multicast-адрес протокола IPv6, должен быть доставлен на сетевые интерфейсы всех членов указанной группы.

Кластерный адрес определяет группу интерфейсов, принадлежащих, как правило, разным узлам. Пакет, адресованный на Anycast-адрес протокола IPv6, должен быть доставлен только на один интерфейс, принадлежащий активному члену данного кластера.

Примечание

В качестве кластерного может быть использован любой из индивидуальных адресов протокола IPv6. Адреса такого типа могут быть присвоены, например,

кластеру, состоящему из двух пограничных маршрутизаторов, который построен по принципу горячего резервирования. При этом узлы подключенной сети должны использовать для обмена с внешними сетями только один адрес, который не будет изменен даже в случае выхода из строя активного маршрутизатора кластера.

Использование кластерных адресов протокола IPv6, при наличии соответствующих программ, обеспечивающих организацию кластеров, позволит свести к минимуму проблемы, возникающие при резервировании сетевых компонентов или реконфигурации сети.

Формы представления сетевых адресов протокола IPv6

Сетевые 128-разрядные адреса протокола IPv6 принято представлять в виде восьми 16-разрядных целых чисел, разделенных двоеточием. Для обеспечения компактности записи каждое из этих чисел представляется четырьмя шестнадцатеричными цифрами. Таким образом, полный сетевой адрес протокола IPv6 представляет собой последовательность, состоящую из 32 шестнадцатеричных цифр.

Спецификация протокола IPv6 для обеспечения большей компактности и читаемости сетевого адреса, разрешает использование при его написании специального метасимвола, состоящего из пары последовательных двоеточий (::). Применение этого метасимвола позволяет не указывать количество лидирующих и завершающих нулей в сетевых адресах IPv6. Символ "::" также может быть использован для замены непрерывной последовательности нулей, находящейся внутри сетевого адреса. Следует, однако, отметить, что при написании одного сетевого адреса протокола IPv6 может быть использован только один метасимвол "::". Ожидается, что на начальной стадии развертывания протокола IPv6 только часть из 128 бит найдет реальное применение, поэтому сетевые адреса будут содержать множество нулей. Таким образом, использование метасимвола "::" вместо последовательности нулевых чисел должно упростить процесс конфигурации сетевых устройств протокола IPv6.

В качестве иллюстрации рассмотрим следующий полный 128-разрядный адрес протокола IPv6: 501A:0000:0000:0000:00FC:ABCD:3F1F:3D5A. Поскольку спецификация протокола IPv6 позволяет пропустить начальные нули в каждом блоке из 4 шестнадцатеричных цифр, указанный адрес можно записать в сокращенном виде следующим образом: 501A:0:0:0:FC:ABCD:3F1F:3D5A. Если, к тому же, воспользоваться принятым соглашением об исключении последовательных блоков нулей, адрес можно записать еще короче — 501A::FC:ABCD:3F1F:3D5A. Теперь адрес состоит всего из пяти шестнадцатеричных чисел вместо восьми.

Специальные сетевые адреса протокола IPv6

В протоколе IPv6 не существует понятия класса, однако, по традиции, старшие разряды адреса этого протокола используются в качестве признака принадлежности данного адреса к определенному типу.

В табл. 15.2 перечислены некоторые типы сетевых адресов протокола IPv6.

Таблица 15.2. *Типы и назначение сетевых адресов протокола IPv6*

Значение старшего байта	Тип сетевого адреса	Назначение
0000 0000	Reserved	Зарезервирован для адресов протокола IPv4
0000 001*	Reserved for IPX	Зарезервирован для адресов протокола IPX
001* ****	Global Unicast Addresses	Глобальные индивидуальные адреса протокола IPv6
1111 1110	Local Unicast Addresses	Локальные индивидуальные адреса протокола IPv6
1111 1111	Multicast Addresses	Групповые адреса протокола IPv6

Некоторые адреса из первого резервного диапазона протокола IPv6 предназначены для выполнения специальных установленных функций. Например, значение "все нули" (в компактной нотации — "::") обозначает ситуацию, когда сетевой адрес не был назначен и поэтому соответствует неопределенному адресу протокола IPv6. Этот адрес не может быть использован в качестве адреса назначения.

Сетевой адрес — 0:0:0:0:0:0:0:1 (::1) присвоен постоянно активному виртуальному интерфейсу каждого узла сети протокола IPv6 и может быть использован для проверки функционирования стека протоколов данного узла при помощи стандартного набора диагностических команд (например, команды `ping`).

В этом же диапазоне сетевых адресов предполагается разместить гибридные адреса протоколов IPv6 или IPv4, предназначенные для обеспечения взаимной совместимости этих сетей.

Например, IPv4-совместимые адреса протокола IPv6 будут состоять из 32-разрядных адресов протокола IPv4, записанных в младших 32 разрядах адресного пространства протокола IPv6. Иначе говоря, префикс такого адреса будет состоять из 96 нулей. Для таких случаев спецификация протокола

IPv6 предусматривает возможность использования специальной нотации, в которой последние 32 бита сетевого адреса представляются в принятом для сети протокола IPv4 десятичном виде. Таким образом, сетевой адрес протокола IPv4 — 194.195.196.197 соответствует сетевому адресу протокола IPv6 — 0:0:0:0:0:0:194.195.196.197, который благодаря использованию компактной нотации может быть представлен в почти привычном для пользователей современной сети Интернет виде — ::194.195.196.197 .

Заголовки расширения протокола IPv6

Как было отмечено в предыдущих разделах данной главы, заголовки расширения обеспечивают возможность определения дополнительных функций, выполняемых на сетевом уровне протокола IPv6.

Заголовок Hop-by-Hop Options

Заголовок опций промежуточных узлов предназначен для определения дополнительных функций по обработке блока данных сетевого уровня протокола IPv6, которые должны выполнятся на каждом из промежуточных узлов сети по пути следования этого блока данных по сети к узлу назначения. На наличие заголовка расширения Hop-by-Hop Options указывает значение 0 идентификатора **Next Header** предыдущего заголовка.

Структура заголовка расширения протокола IPv6 Hop-by-Hop Options представлена на рис. 15.5.

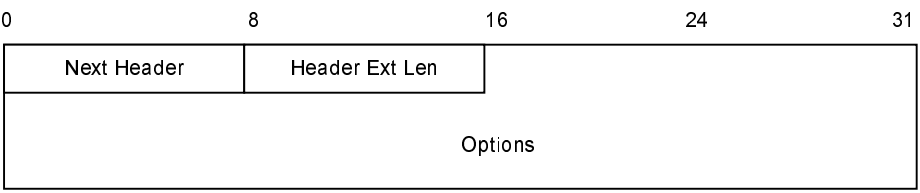


Рис. 15.5. Структура заголовка расширения протокола IPv6 Hop-by-Hop Options

Поле **Next Header** занимает 8 бит в заголовке расширения Hop-by-Hop Options и предназначено для указания типа следующего заголовка расширения данного пакета протокола IPv6.

Поле **Header Ext Len** занимает 8 бит в заголовке расширения Hop-by-Hop Options и предназначено для размещения длины поля полезной нагрузки данного заголовка расширения, выраженной в 64-битных словах.

Поле **Options** заголовка расширения Hop-by-Hop Options имеет переменную длину и предназначено для размещения параметров дополнительных процедур обработки данного информационного пакета.

Следует отметить, что в протоколе IPv6 используется стандартная схема формирования полей подобного рода, которая имеет название TLV (Type-Length-Value). При использовании данной схемы дополнительные параметры оформляются в виде информационных блоков, структура которых представлена на рис. 15.6.

Рис. 15.6. Универсальная структура представления данных поля Option

Блок **Option Type** занимает первые 8 бит в поле **Option** и предназначен для указания типа данной опции. Значения двух старших битов **Option Type** определяют выбор одного из четырех возможных режимов обработки данного блока в том случае, если значение, размещенное в последующих битах, не было распознано узлом сети.

В табл. 15.3 перечислены возможные варианты обработки нераспознанных опций пакетов протокола IPv6 в зависимости от значения старших разрядов поля **Option Type**.

Таблица 15.3. *Варианты обработки нераспознанных опций пакетов протокола IPv6*

Старшие биты поля Option Type	Обработка пакета
00	Пропустить опцию и продолжить обработку пакета
01	Уничтожить пакет
10	Уничтожить пакет и сформировать сообщение протокола ICMP "Parameter Problem" в адрес его источника
11	Уничтожить пакет и, если это был не MULTICAST–пакет, сформировать сообщение протокола ICMP "Parameter Problem" в адрес его источника

Блок **Opt Data Len** занимает 8 бит в поле **Option** и предназначен для размещения длины последующего блока **Option Data**, выраженной в байтах.

Блок **Option Data** имеет переменную длину и предназначен для непосредственного размещения управляющей информации данной опции.

Заголовок маршрутизации Routing Header

Заголовок маршрутизации предназначен для явного указания источником узловых точек маршрута прохождения данного пакета протокола IPv6 по сети. В заголовке маршрутизации помещаются адреса промежуточных узлов, через которые должен обязательно пройти пакет в процессе его доставки на узел назначения. На наличие заголовка маршрутизации Routing Header в пакете протокола IPv6 указывает значение 43 идентификатора **Next Header** предыдущего заголовка.

Структура заголовка расширения протокола IPv6 Routing Header представлена на рис. 15.7.

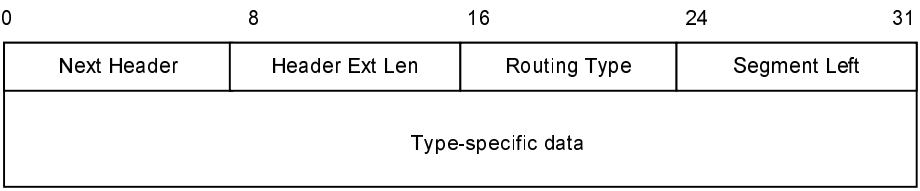


Рис. 15.7. Структура заголовка расширения протокола IPv6 Routing Header

Поле **Next Header** занимает 8 бит в заголовке маршрутизации Routing Header и предназначено для указания типа следующего заголовка расширения данного пакета протокола IPv6.

Поле **Header Ext Len** занимает 8 бит в заголовке маршрутизации Routing Header и предназначено для размещения длины поля полезной нагрузки данного заголовка расширения, выраженной в 64-битных словах.

Поле **Routing Type** занимает 8 бит в заголовке маршрутизации Routing Header и предназначено для размещения номера используемого варианта заголовка маршрутизации.

Поле **Segment Left** также занимает 8 бит в заголовке маршрутизации Routing Header и содержит количество промежуточных узлов указанного маршрута, которые осталось пройти данному пакету.

Поле **Type-specific data** в заголовке маршрутизации Routing Header имеет переменную длину. Структура этого поля определяется используемым вариантом формирования маршрута.

На рис. 15.8 представлена в качестве примера структура заголовка маршрутизации, определенная в спецификации протокола IPv6 для варианта **Routing Type** = 0.

Значение стандартных полей заголовка маршрутизации определено выше по тексту. Поле **Reserved** заголовка маршрутизации имеет длину 32 бит и зарезервировано для дальнейшего использования.

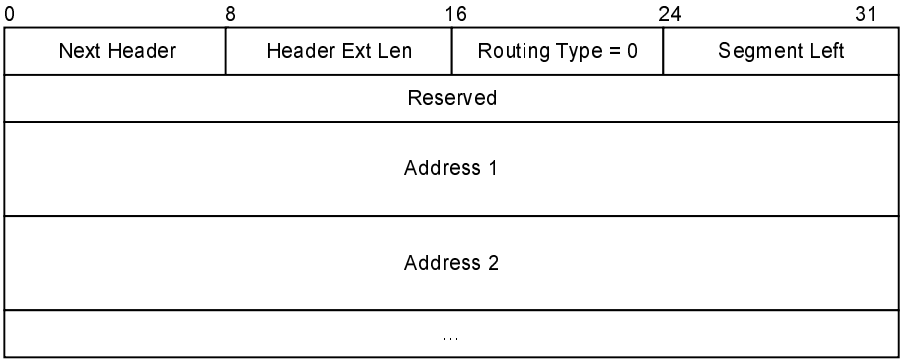


Рис. 15.8. Структура заголовка маршрутизации протокола IPv6 для варианта **Routing Type** = 0

Последующие поля **Address 1**, **Address 2** и т. д. имеют длину 128 бит и предназначены для непосредственного размещения сетевых адресов промежуточных узлов маршрута.

Заголовок фрагментации Fragment Header

Заголовок фрагментации предназначен для обеспечения передачи по сети протокола IPv6 блоков данных, размер которых превышает значение Maximum Transfer Unit (MTU), установленное на каналах, соединяющих промежуточные узлы данного маршрута. В отличие от принципа фрагментации, использованного в протоколе IPv4, промежуточные IPv6-узлы не имеют возможности выполнять фрагментацию пакета по собственной инициативе. Инициатором процедуры фрагментации в сетях протокола IPv6, как правило, является источник пакета. Для того чтобы избежать фрагментации пакета, его источник может использовать специальные процедуры определения маршрута с одновременным фиксированием значения MTU. На наличие заголовка фрагментации Fragment Header в пакете протокола IPv6 указывает значение 44 идентификатора **Next Header** предыдущего заголовка.

Примечание

В некоторых случаях промежуточные узлы протокола IPv6 все-таки могут выполнять фрагментацию исходного пакета, однако для этого они должны использовать стандартную процедуру построения туннеля. В этом случае общий принцип фрагментации пакетов не нарушается, поскольку граничный узел тоннеля становится вторичным источником пакета.

Структура заголовка расширения протокола IPv6 Fragment Header представлена на рис. 15.9.

Поле **Next Header** занимает 8 бит в заголовке фрагментации Fragment Header и предназначено для указания типа следующего заголовка расширения данного пакета протокола IPv6.

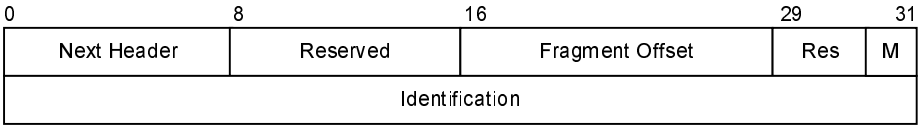


Рис. 15.9. Структура заголовка расширения протокола IPv6 Fragment Header

Поля **Reserved** и **Res** занимают 8 и 2 бита соответственно в заголовке фрагментации Fragment Header и зарезервированы для дальнейшего использования.

Поле **Fragment Offset** занимает 13 бит в заголовке фрагментации Fragment Header. Назначением этого поля является указание положения фрагмента данных, расположенного в поле полезной нагрузки данного пакета относительно начала исходного (нефрагментированного) блока данных.

Поле **M** занимает 1 бит заголовке фрагментации Fragment Header и предназначено для размещения признака наличия последующих фрагментов. В это поле помещается значение 0, если фрагмент, переносимый данным пакетом, является завершающим фрагментом последовательности. В противном случае значение этого поля устанавливается равным 1.

Поле **Identification** занимает 32 бита в заголовке фрагментации Fragment Header и предназначено для размещения индивидуального идентификатора — признака фрагментированного пакета. Наличие данного признака позволяет приемнику одновременно восстанавливать несколько фрагментированных пакетов, получаемых от одного источника.

Заголовок Destination Options Header

Заголовок опций промежуточных узлов предназначен для определения дополнительных функций по обработке блока данных сетевого уровня протокола IPv6, которые должны быть выполнены на узле назначения при получении данного пакета. На наличие заголовка расширения Destination Options Header указывает значение 60 идентификатора **Next Header** предыдущего заголовка.

Структура заголовка расширения протокола Destination Options Header представлена на рис. 15.10.

Поле **Next Header** занимает 8 бит в заголовке расширения Destination Options Header и предназначено для указания типа следующего заголовка расширения данного пакета протокола IPv6.

Поле **Header Ext Len** занимает 8 бит в заголовке расширения Destination Options Header и предназначено для размещения длины поля полезной нагрузки данного заголовка расширения, выраженной в 64-битных словах.

Поле **Options** заголовка расширения Destination Options Header имеет переменную длину и предназначено для размещения параметров дополнительных

процедур обработки данного информационного пакета. При формировании информационных блоков, размещаемых в данном поле, применяется схема TLV (Type-Length-Value), описанная в предыдущих разделах данной главы.

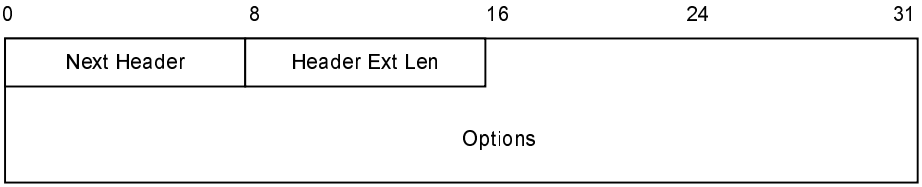


Рис. 15.10. Структура заголовка расширения протокола IPv6 Destination Options Header

Заголовки информационной безопасности протокола IPv6

Для обеспечения информационной безопасности в протоколе IPv6 могут быть использованы следующие заголовки расширения:

- Encapsulating Security Payload Header;
- Authentication Header.

Заголовок Encapsulating Security Payload Header обеспечивает возможность шифрования информации, размещенной в поле полезной нагрузки пакета протокола IPv6.

Заголовок Authentication Header обеспечивает выполнение функций защиты от несанкционированного изменения информации, размещенной в заголовке пакета протокола IPv6.

Указанные заголовки являются универсальными, что обеспечивает возможность их использования как в сетях протокола IPv6, так и в классической сети Интернет протокола IPv4. *(Относительно подробное описание указанных заголовков уже было приведено в гл. 8.)*

Грандиозное переселение

Переход от протокола IPv4 к протоколу IPv6 займет не один год. Если вы довольны размерами имеющихся в вашем распоряжении блоков IP-адресов и достаточно консервативны в отношении использования новых технологий, вам не о чем беспокоиться. Скорее всего, вы сможете использовать протокол IPv4 по крайней мере до тех пор, пока IPv6-совместимые продукты не вытеснят устаревшее оборудование или вы сами не примете решение о миграции.

Если вы решили мигрировать к протоколу IPv6, то сделать это можно двумя способами: с помощью двойных стеков или туннелирования. И тот, и другой метод обратно совместим с протоколом IPv4.

Двойные стеки позволяют каждому узлу протокола IP поддерживать как протокол IPv6, так и протокол IPv4. Подобные узлы принято называть IPv6/IPv4-узлами; они могут отправлять и принимать пакеты протоколов как IPv4, так и IPv6. Использование двойного стека позволяет переводить на протокол IPv6 каждый узел сети по отдельности, а использование его на маршрутизаторе позволит подключить сеть вашей организации к оператору сети Интернет, предоставляющему доступ в сеть IPv6. Такое решение позволяет постепенно перевести узлы в сети на протокол IPv6 посредством последовательной установки на них двойных стеков.

Второй метод перехода к протоколу IPv6 — туннелирование. Протокол IPv6 не является обратно совместимым с протоколом IPv4, а IPv4-совместимые устройства не могут непосредственно работать с IPv6-пакетами. Поэтому передача IPv6-пакетов через инфраструктуру протокола IPv4 невозможна без выполнения туннелирования путем помещения IPv6-пакетов в IPv4-пакеты. Сеть Интернет и практически все частные сети протокола IP базируются на инфраструктуре протокола IPv4. Таким образом, туннелирование позволит IPv4- и IPv6-узлам общаться друг с другом через "устаревшие" сети протокола IP.

Глоссарий

10BaseT

Стандарт IEEE, описывающий сеть Ethernet на неэкранированной витой паре (UDP — Unshielded Twister Pair). Сеть Ethernet не может работать с использованием обычного телефонного кабеля.

A

ACK Acknowledgement

Стандартное сообщение, используемое в процедурах квитирования для обозначения позитивного подтверждения.

AH Authentication Header

Дополнительный заголовок блока данных сетевого уровня протокола IP, предназначенный для защиты от несанкционированного изменения информации, размещенной в основном заголовке.

ANSI American National Standards Institute

Национальная стандартизирующая организация США.

ARP Address Resolution Protocol

Протокол, используемый для установления взаимного соответствия адресов канального и сетевого уровней в сетевых технологиях множественного доступа к среде передачи данных.

ARPAnet Advanced Research Project Agency network

Сеть, созданная ARPA (Агентство Передовых Проектов), основные принципы построения которой были положены в основу сети Интернет.

ARIN American Registry for Internet Numbers

Организация, координирующая процесс назначения сетевых адресов Интернет в странах региона Северной и Южной Америки.

AS Autonomous System

Область, представляющая собой группу маршрутизаторов и сетей, объединенных по иерархическому принципу, находящихся под единым управлением и реализующих единую политику маршрутизации.

ASN.1 Abstract Syntax Notation One

Стандартная форма представления информации, используемая в протоколе SNMP.

ASCII American Standard Code for Information Interchange

Набор условно закодированных текстовых и управляющих символов. Под кодами ASCII обычно подразумеваются младшие 128 формально определенных символов (расширенный набор ASCII кодов состоит из 256 символов).

B

BGP Border Gateway Protocol

Протокол внешней маршрутизации, используемый для построения маршрутов информационного обмена между различными автономными системами.

BOOTP Bootstrap Protocol

Протокол, используемый для автоматического конфигурирования бездисковых рабочих станций.

Broadcast (Широковещательный обмен)

Режим адресации, который используется для передачи блоков данных сетевого уровня всем без исключения узлам сети.

BS Bootstrap

Обозначение группового маршрутизатора, обеспечивающего информационное взаимодействие потребителей Multicast-трафика и маршрутизатора RP.

C

Cache (Кэш-память)

Буферная память, используемая для промежуточного хранения передаваемых и принимаемых блоков данных.

CCITT Consultative Committee for International Telephone and Telegraphy

Международный консультативный комитет по телефонии и телеграфии. В настоящее время функции этого комитета выполняет ITU-T.

CGI Common Gateway Interface

Спецификация, определяющая интерфейс программного взаимодействия с WWW-сервером.

CGMP Cisco Group Management Protocol

Протокол, в соответствии с которым осуществляется информационное взаимодействие между маршрутизатором и коммутатором Cisco Systems для обеспечения эффективного распространения группового трафика.

CHAP Challenge Handshake Authentication Protocol

Вспомогательный протокол, который применяется для установления подлинности партнеров PPP соединения и использует 3-этапный алгоритм квитирования.

CIDR Classless Inter-Domain Routing

Современная концепция маршрутизации, обеспечивающая возможность использования внеклассовых сетей (subnet и supernet) в Интернет.

CMIP Common Management Information Protocol

Стандартный протокол управления узлами сети Интернет. Имеет ряд дополнительных функций, отсутствующих в протоколе SNMP.

Codec (Кодек — кодер/декодер)

Устройство или программа, выполняющая функцию взаимного преобразования потокового трафика в последовательность кодов.

CoS Class of Service

Класс обслуживания. Используется в системах DiffServ для указания принадлежности трафика к одной из групп с гарантированными значениями сетевых ресурсов, выделяемых на его обработку.

CRLF Carriage Return Line Feed

Последовательность управляющих символов: возврат каретки (13), перевод строки (10).

CSMA/CD Carrier Sense Multiple Access/Collision Detection

Метод множественного доступа к среде передачи данных с прослушиванием несущей и обнаружением коллизий, положенный в основу сети Ethernet.

DARPA Defense Advanced Research Project Agency

Агентство Передовых Проектов Национальной Безопасности США.

Datagram (Дейтаграмма)

Термин, используемый для обозначения блока данных сетевого уровня Интернет.

Default Gateway (Шлюз, используемый по умолчанию)

Термин, используемый для обозначения одного из узлов непосредственно подключенной сети, которому следует направлять дейтаграммы, адресованные в сети, не представленные в таблице маршрутизации.

Dense mode

Режим трансляции группового трафика по сети при использовании которого этот трафик не получают только те узлы, которые явным образом отказались от его приема.

DCE Data Circuit Terminal Equipment

Тип последовательного интерфейса линейного оборудования передачи данных.

DHCP Dynamic Host Configuration Protocol

Протокол, обеспечивающий динамическое определение значений основных сетевых параметров оконечного узла Интернет.

DiffServ Differentiated Service Model

Модель раздельного обслуживания. Схема обеспечения качества обслуживания в сети Интернет, которая предполагает классовое разделение обрабатываемого трафика.

Direct Delivery (Непосредственная доставка)

Способ, который используется для доставки блоков данных в том случае, когда адресат сообщения находится в том, же сегменте сети, что и источник этого сообщения.

Distance-Vector algorithm (Алгоритм вектора расстояния)

Класс протоколов маршрутизации, использующих для построения маршрутной таблицы, периодически рассылаемые обновления, формируемые соседними маршрутизаторами.

DNA Digital Network Architecture

Архитектура, предложенная корпорацией Digital Equipment (DEC) для построения информационно-вычислительных сетей.

DNS Domain Name System

Распределенная система, обеспечивающая возможность определения сетевого адреса Интернет по иерархическому идентификатору сетевого ресурса.

DoS Denial of Service

Класс атак, направленных на подавление информационных ресурсов сетевого узла.

DTE Data Terminal Equipment

Тип последовательного интерфейса оконечного оборудования передачи данных.

DVMRP Distance Vector Multicast Routing Protocol

Протокол маршрутизации группового трафика, построенный аналогично классическим протоколам типа Distance — Vector.

EDEL End Delimiter

Поле, отмечающее завершение кадра, передаваемого по сети с множественным доступом к среде передачи данных.

EGP Exterior Gateway Protocol

Протокол внешней маршрутизации, используемый для построения маршрутов между различными автономными системами. В настоящее время практически не применяется.

EIA Electronic Industries Alliance

Национальная организация США, объединяющая производителей телекоммуникационного оборудования. Задачей организации EIA является развитие американского рынка высоких технологий.

EIGRP Enhanced IGRP

Универсальный гибридный протокол маршрутизации, разработанный Cisco Systems Inc.

Ethernet

Технология, предложенная корпорацией Xerox для построения локальных вычислительных сетей с использованием метода CSMA/CD обеспечения множественного доступа к среде передачи данных.

ESP Encapsulating Security Payload

Дополнительный заголовок блока данных сетевого уровня протокола IP, предназначенный для защиты информации, размещенной в поле полезной нагрузки данного блока.

F

FCS Field Checksum

Обозначение поля контрольной суммы в заголовке блока данных протокола сетевого или канального уровня.

FDDI Fiber Distributed Data Interface

Технология построения резервированных магистральных межсетевых соединений на основе волоконно-оптических линий связи.

FEC Forwarding Equivalence Class

Совокупность адресов назначения блоков данных сетевого уровня, требующих однотипной обработки на маршрутизаторе LSR.

Frame (Кадр)

Стандартный термин, используемый для обозначения блоков данных протоколов канального уровня (SLIP, PPP).

FTP File Transfer Protocol

Протокол прикладного уровня, применяемый для организации файлового обмена в сети Интернет.

G

Gateway (Шлюз)

Общий термин, применяемый для обозначения сетевых узлов, обеспечивающих информационное взаимодействие между несколькими сетями.

GDA Group Destination Address

Сетевой адрес Интернет класса D, который используется для обозначения группы приемников Multicast-трафика.

GGP Gateway-to-Gateway Protocol

Протокол обмена управляющей информацией между сетевыми маршрутизаторами.

H

HDLC High-level Data Link Control

Универсальный протокол канального уровня, обеспечивающий управление звеном передачи данных.

Hop-by-Hop

Принцип, используемый для организации доставки блоков данных сетевого уровня в Интернет.

Host (Хост)

Общий термин, применяемый для обозначения оконечных сетевых узлов.

HSRP Hot Standby Router Protocol

Протокол, применяемый Cisco Systems Inc. для обеспечения горячего резервирования маршрутизаторов ответственных сетевых узлов.

HTML Hyper Text Markup Language

Спецификация, определяющая формат документов, размещаемых на WWW-серверах в сети Интернет.

HTTP Hypertext Transfer Protocol

Протокол прикладного уровня, используемый для доступа к информации, размещенной на WWW-серверах.

I

IAB Internet Activities Board

Комитет IETF, выполняющий организационно-технические функции при разработке и сопровождении основополагающих документов и рабочих спецификаций Интернет.

IANA Internet Assigned Number Authority

Организация, занимающаяся вопросами координации выделения IP-адресов, номеров автономных систем и т. д.

ICMP Internet Control Message Protocol

Протокол сетевого уровня, который используется для выполнения технологических функций в сети Интернет.

IETF Internet Engineering Task Force

Международная организация, выполняющая весь комплекс работ по подготовке основополагающих документов и рабочих спецификаций сети Интернет.

IGMP Internet Group Management Protocol

Протокол, используемый оконечными узлами сети Интернет для указания своей потребности в приеме группового трафика.

IGP Interior Gateway Protocol

Класс протоколов маршрутизации, предназначенных для построения маршрутов, не выходящих за пределы автономной области.

IHL Internet Header Length

Поле заголовка IP-пакета, предназначенное для размещения длины заголовка, выраженной в байтах.

IMAP4 Internet Message Access Protocol version 4

Протокол прикладного уровня, предназначенный для обеспечения комплекса процедур управления сообщениями электронной почты Интернет, версия 4.

Indirect Delivery (Опосредованная доставка)

Способ, который используется для доставки блоков данных в том случае, когда адресат сообщения не находится в том же сегменте сети, что и источник этого сообщения.

IP Internet Protocol

Протокол сетевого уровня сети Интернет.

IPCP IP Control Protocol

Вспомогательный протокол, который обеспечивает согласование параметров протокола IP (например — назначение IP-адресов) на обоих концах PPP-соединения.

Ipng Internet Protocol (next generation)

Новый стандарт протокола IP (см. IPv6).

IntServ Integrated Service Model

Модель объединенного обслуживания. Схема обеспечения качества обслуживания в сети Интернет, которая не предполагает классовое разделение обрабатываемого трафика.

IPsec IP Security

Комплекс протоколов, предназначенных для обеспечения безопасности информационного обмена в сети Интернет.

IPv4 Internet Protocol version 4

Современная (четвертая) версия протокола сетевого уровня Интернет.

IPv6 Internet Protocol version 6

Перспективная (шестая) версия протокола сетевого уровня Интернет.

IPX Internetwork Packet Exchange

Протокол сетевого уровня, использовавшийся в сетях Novell NetWare.

ISDN Integrated Service Digital Network

Технология построения цифровых сетей, обеспечивающих одновременную передачу данных и голосового трафика по принципу временного разделения.

ISN Initial Sequence Number

Стартовое значение управляющей синхронизирующей последовательности, используемой протоколом TCP для обеспечения гарантированной доставки сегментов.

ISO International Standardization Organization

Официальная международная стандартизирующая организация.

ITU International Telecommunication Union

Международная организация, выполняющая комплекс работ по развитию телекоммуникационных технологий.

L

LAN Local Area Network

Сеть, информационное взаимодействие между узлами которой осуществляется с использованием протоколов канального уровня.

LCP Link Control Protocol

Вспомогательный протокол, который обеспечивает непосредственное управление функционированием канала передачи данных при использовании протокола PPP.

LDP Label Distribute Protocol

Протокол, используемый маршрутизаторами MPLS для обмена информацией о соответствии значений устанавливаемых меток и потоков FEC.

LHA Length Hardware Address

Обозначение поля размера адреса протокола канального уровня в кадре ARP.

Link-State Algorithm (Алгоритм состояния канала)

Класс протоколов маршрутизации, использующих для построения маршрутной таблицы сообщения, формируемые маршрутизаторами при изменении состояния сетевого ресурса (канала передачи данных или шлюза).

LNI Local Network Interface

Интерфейс, который используется для подключения к локальной сети.

Load Balanced (Выравнивание нагрузки)

Процедура увеличения скорости информационного обмена при одновременном использовании нескольких параллельных каналов для приема и передачи данных.

LPA Length Protocol Address

Обозначение поля размера адреса протокола сетевого уровня в кадре ARP.

LSA Link-State Advertisement

Сообщения о состояниях каналов маршрутизации, используемые в протоколе маршрутизации OSPF для формирования маршрутной таблицы.

LSR Label Switch Router

Маршрутизатор, выполняющий обработку входных блоков данных сетевого уровня в соответствии с алгоритмом коммутации меток MPLS.

M

MAC Media Access Control

Обозначение промежуточного уровня доступа к среде передачи в технологиях CSMA/CD.

Mask (Маска)

Используется для определения структуры сетевого адреса внеклассовой сети. Биты маски, соответствующие адресу сети в сетевом адресе, долж-

ны быть равны 1, а биты, соответствующие адресу узла, должны быть равными 0.

MIB Management Information Base

Информационная база, в которой размещается информация, необходимая для управления сетевыми объектами по протоколу SNMP.

MIME Multipurpose Internet Mail Extensions

Стандарт, используемый для преобразования данных в текстовые сообщения для последующей передачи по электронной почте.

MLP Multi Link PPP

Протокол расширения PPP, который обеспечивает возможность увеличения скорости передачи данных при одновременном использовании нескольких параллельных каналов.

MPLS Multi Protocol Label Switching

Универсальная технология, при использовании которой перемещение информационных блоков по сети производится в соответствии со значением специальных меток, помещаемых в эти блоки.

MTU Maximum Transfer Unit

Максимальный размер блока данных установленный для передачи между узлами сети на канальном уровне.

Multicast (Групповая передача)

Режим адресации, который используется для передачи блоков данных сетевого уровня только для установленной группы узлов сети.

N

NAK Negative Acknowledgement

Стандартное сообщение, используемое в процедурах квитирования для обозначения отрицания подтверждения.

NCP Network Communication Protocol

Протокол сетевого обмена, использовавшийся в сети ARPAnet до перехода на TCP/IP.

NCP Network Control Protocol

Вспомогательный протокол, который обеспечивает выбор и согласование конфигурации протокола PPP для взаимодействия с протоколом сетевого уровня.

NFS Network File System

Сетевая файловая система. Протокол, обеспечивающий объединение файловых систем сетевых узлов Интернет.

NIC Network Information Center

Международный информационный центр сетевых технологий. В настоящее время его функции выполняет INTERNIC.

NMS Network Management Systems

Программные комплексы, выполняющие функции управления сетевыми компонентами.

NNTP Network News Transport Protocol

Протокол прикладного уровня, предназначенный для передачи новостей, тиражирования статей и обеспечения функционирования распределенной системы дискуссий.

NVT Network Virtual Terminal

Сетевой виртуальный терминал.

O

OSI Open System Interconnect

Концепция организации информационного взаимодействия открытых систем, основные принципы которой изложены в стандарте ISO 7498.

OSPF Open Shortest Path First

Протокол класса Link-State, определенный в качестве базового протокола маршрутизации для использования в сети Интернет.

P

Packet (Пакет)

Стандартный термин, используемый для обозначения блоков данных протокола сетевого уровня.

PAP Password Authentication Protocol

Вспомогательный протокол, который применяется для установления подлинности партнеров PPP соединения и использует простую схему аутентификации.

PARC Palo Alto Research Center

Исследовательский центр корпорации Xerox, расположенный в Пало Альто.

PDH Plesiochronous Digital Hierarchy

Плезиохронная цифровая иерархия — технология построения цифровых каналов передачи данных. Основные принципы этой технологии изложены в рекомендациях ITU-T G.703 и G.704.

PDU Protocol Data Units

Термин, который может быть использован для обозначения блоков данных протокола любого уровня.

PING Packet Inter Network Groper

Название популярного приложения, которое используется для определения достижимости узлов сети с использованием протокола ICMP.

PIM Protocol Independent Multicast

Универсальный протокол, используемый для маршрутизации группового трафика в сети Интернет.

POP3 Post Office Protocol version 3

Протокол прикладного уровня, предназначенный для обеспечения процедур доступа к сообщениями электронной почты Интернет.

PPP Point-to-Point Protocol

Универсальный протокол канального уровня, предназначенный для использования на двухточечных каналах передачи данных.

Q

QoS Quality of Service

Совокупность значений сетевых ресурсов, выделяемых для обработки трафика, принадлежащего к соответствующему классу.

Queue (Очередь)

Совокупность принятых или передаваемых блоков данных, размещенных для промежуточного хранения в буферной памяти сетевого устройства.

Queuing

Универсальная процедура, используемая для регулирования темпа и порядка следования блоков данных.

R

RARP Reverse Address Resolution Protocol

Протокол, используемый для установления взаимного соответствия адресов сетевого и канального уровней в технологиях множественного доступа к среде передачи данных. В настоящее время для выполнения аналогичных функций обычно применяется протокол DHCP.

RFC Requests For Comments

Название серии документов, подготавливаемых специалистами IETF, в которых рассматриваются предварительные предложения, определяю-

щие направления модернизации и дальнейшего развития комплекса технологий сети Интернет.

RIP Routing Information Protocol

Универсальный протокол маршрутизации класса Distance Vector. Протокол RIP используется для определения маршрутов с момента возникновения сети Интернет.

RIP II Routing Information Protocol, version 2

Современная версия популярного протокола маршрутизации RIP, которая отвечает требованиям по обеспечению построения маршрутов для внеклассовых сетей Интернет.

Router (Маршрутизатор)

Специальное сетевое устройство или выделенный компьютер, реализующий комплекс процедур информационного взаимодействия на сетевом уровне.

RP Rendezvous Point

Термин, используемый в протоколе групповой маршрутизации PIM для обозначения выделенного маршрутизатора, обеспечивающего поиск источника группового трафика.

RPC Remote Procedure Call

Протокол сеансового уровня, обеспечивающий возможность вызова процедуры, выполняемой на удаленном сетевом узле.

RPF Reverse Path Forwarding

Процедура, которая используется для предотвращения возникновения циклических маршрутов при передаче группового трафика.

RR Resource Record

Запись в конфигурационном файле системы DNS, которая содержит информацию о сетевом ресурсе.

RS-232 (EIA/TIA-232)

Популярный протокол физического уровня, обеспечивающий обмен данных через последовательный порт со скоростью до 115 Кбит/с. По основным принципам построения соответствует рекомендации V.24 ITU-T.

RSVP ReSerVation Protocol

Протокол прикладного уровня, предназначенный для резервирования ресурсов, выделяемых для обработки трафика выделенной группы в системах IntServ.

RTCP Real Time Control Protocol

Вспомогательный протокол, который обеспечивает управление сетевыми компонентами при использовании протокола RTP.

RTP Real Time Protocol

Протокол прикладного уровня, обеспечивающий передачу потокового трафика по сети Интернет.

S

SA Security Association

Соединение защиты / защищенное соединение. Одностороннее соединение между узлами сети, которому соответствует согласованный набор установленных значений параметров протоколов информационной безопасности.

SDEL Start Delimiter

Поле, отмечающее начало кадра, передаваемого по сети с множественным доступом к среде передачи данных.

Segment (Сегмент)

Принятое обозначение информационного блока протокола транспортного уровня, обеспечивающего гарантированную доставку данных.

SFTP Simple File Transfer Protocol

Протокол прикладного уровня, применяемый для организации простого файлового обмена в сети Интернет.

SGMP Simple Gateway Monitoring Protocol

Протокол удаленного контроля компонентов сети Интернет. В настоящее время для выполнения аналогичных функций используется протокол SNMP.

SLIP Serial Line Internet Protocol

Протокол канального уровня, обеспечивающий подключение удаленного узла к сети Интернет по последовательному каналу.

SMTP Simple Mail Transfer Protocol

Протокол прикладного уровня, обеспечивающий передачу почтовых сообщений в сети Интернет.

SNA Systems Network Architecture

Технология, предложенная корпорацией IBM для построения информационно-вычислительных сетей.

SNMP Simple Network Management Protocol

Протокол прикладного уровня, обеспечивающий мониторинг и управление значениями параметров удаленного узла сети Интернет.

Socket (Гнездо)

Термин, используемый для идентификации соединений транспортного уровня узла сети.

Sparse mode

Режим трансляции группового трафика по сети, при использовании которого этот трафик получают только те узлы, которые явным образом заявили о необходимости его приема.

Sub net (Подсеть)

Внеклассовая сеть Интернет, образованная разделением сети, принадлежащей к классам A—C, на несколько менее крупных сетей.

Super net (Суперсеть)

Внеклассовая сеть Интернет, образованная слиянием нескольких сетей, принадлежащих к классам A—C, в более крупную сеть.

SWS Silly Window Syndrome (Синдром неоптимального окна)

Состояние транспортного протокола TCP, состоящее в том, что приемник последовательно устанавливает размер окна, соответствующий получению исключительно небольших порций данных.

T

T1, T3

Цифровые каналы североамериканской плезиохронной иерархии (PDH), обеспечивающие передачу данных со скоростями 1544 и 44 736 бит/с соответственно.

TCP Transmission Control Protocol

Протокол транспортного уровня, обеспечивающий гарантированную доставку данных в сети Интернет.

TCP/IP Internet Protocol Suite

Официальное название, используемое для обозначения иерархического комплекса (стека) протоколов Интернет.

TELNET (Протокол виртуального терминала)

Протокол прикладного уровня, предназначенный для управления сетевыми узлами Интернет при помощи виртуального терминала.

TFTP Trivial File Transfer Protocol

Протокол прикладного уровня, применяемый для организации простейшего файлового обмена в сети Интернет.

TLV Type-Length-Value

Универсальный принцип, который используется для формирования дополнительных заголовков блока данных сетевого уровня протокола IPv6.

Token Ring

Технология, используемая корпорацией IBM для построения локальных вычислительных сетей с использованием маркерного доступа к разделяемой среде передачи данных.

TOS Type of Service

Поле блока данных сетевого уровня протокола IPv4, предназначенное для указания уровня обслуживания, который должен использоваться для обработки этого блока на узлах сети.

TIA Telecommunications Industry Association

Некоммерческая национальная организация США, занимающаяся разработкой и внедрением стандартов в области телекоммуникаций.

TTL Time To Live

Время "жизни", продолжительность существования блока данных в сети или буферной памяти шлюза.

U

UDP User Datagram Protocol

Протокол транспортного уровня, используемый для негарантированной доставки данных в сети Интернет.

Unicast (Индивидуальный обмен)

Режим адресации, который используется для передачи блоков данных сетевого уровня только указанному узлу сети.

URI Uniform Resource Identifier

Идентификатор сетевого ресурса Интернет.

URL Uniform Resource Locator

Местонахождение сетевого ресурса Интернет.

URN Uniform Resource Name

Символическое имя сетевого ресурса Интернет.

UUCP UNIX to UNIX Copy Protocol

Приложение, обеспечивающее выполнение процедуры копирования файлов между удаленными компьютерами в операционной среде UNIX.

V

VAD Voice Activity Detector

Дополнительный функциональный блок, предназначенный для обнаружения и подавления неинформативных пауз в процессе преобразовании потокового трафика в последовательность кодов.

VPN Virtual Private Network

Инфраструктура, обеспечивающая надежное и безопасное соединение территориально распределенных сетей при использовании информационных ресурсов, предназначенных для общего пользования.

VRRP Virtual Router Redundancy Protocol

Стандартный протокол, предназначенный для обеспечения горячего резервирования маршрутизаторов ответственных сетевых узлов.

W

WAN Wide Area Network

Сеть, информационное взаимодействие между узлами которой осуществляется с использованием протоколов сетевого и последующих уровней.

WWW World Wide Web (Всемирная паутина)

Термин, который иногда используется для обозначения глобальной сети Интернет.

X

X.25

Рекомендация ITU-T, в которой изложены основные принципы построения сети коммутации пакетов, предназначенной для общего использования.

Предметный указатель

A

ACF 280
ACK 210, 213, 215, 216
Address Mask Reply 68, 83, 84
Address Mask Request 68, 83, 84
AD-HOC Block 236, 237
Adjacency 153
Adj-RIBs-In 194
Adj-RIBs-Out 194
Admission 278, 280, 282
Advertised Distance 164, 169
AGGREGATOR 190
AH 304—307, 311, 314
A-law 274, 297
ANSI 3
Anycast 478
Apple Talk 163
Application Layer 6
Area Border Router 151
ARIN 16
ARJ 280
ARP 10, 24, 56, 58—63, 70, 84, 85, 92, 96, 209, 478
  cache 62, 63
ARPA 337, 347, 348, 355, 359—366
ARPAnet 1, 2
ARQ 280
AS Boundary Router 151
AS_PATH 188, 192, 196
ASCII 336, 373, 375—381, 390, 392, 393, 403, 407—409, 422, 430, 456, 459, 460, 462, 463
ASN.1 99, 101, 105
ATOMIC_AGGREGATE 190
Authentication Header 97, 304, 307, 474, 475, 486
Authority 332, 338, 345, 351, 365
Autonomous System 146, 150
Auto-RP 266

B

Backbone 151, 176
Backbone Router 151
Bandwidth 267, 280
BCF 280
BDR 153, 157, 159
BGP 25, 141, 177, 184—198
BINARYMIME 391, 392
BOOTP 87
Bootstrap 266
BR 45
BRJ 280
Broadcast 14, 15, 50, 233, 236, 260, 313, 472
BRQ 280

C

CCITT 3, 38
CELP 275, 276
CGI 431, 432, 435, 436, 439—448
CGMP 249—251
CHAP 44, 45
CHUNKING 391, 392
CIDR 142, 150, 194

Cisco Systems 66, 79, 91, 92, 94, 97, 127, 143, 237, 249, 250, 297
CMIP 98
CNAME 339, 340, 344, 351, 358, 359, 361
Code-Reject 40, 43
Configure-Ack 40—42
Configure-Nak 40, 42
Configure-Reject 40, 42, 43
Configure-Request 40—42, 46
Contributing source 222
Core Architecture 173, 174, 175, 176
CoS 268
CS-ACELP 276
CSMA/CD 50, 51, 54
CSNET 338

D

DARPA 1
Datagram Data Area 20
Datagram Header 20
DataLink Layer 6
DENSE 254, 255, 256, 258, 262, 263
DES 445
Designated Forwarder 260, 261
Designated Router 152, 153
Destination Options Header 474, 475, 485, 486
Destination Unreachable 67—71, 81
DHCP 66, 84, 87—91, 293
DHCPACK 88
DHCPDISCOVER 88, 91
DHCPOFFER 88, 90
DHCPREQUEST 88
DiffServ 268
Discard-Request 41, 44
Distance-Vector 128, 135, 143, 145, 155, 163, 169, 170, 185
Distributed Tree 255
DNA 3
DNS 27, 29, 329—338, 342, 345—348, 353, 354, 359, 362—369
DoS 77, 308
DR 152, 154, 157, 159, 161
DTE/DCE 38
DTMF 296
DUAL 164—166, 168, 171
DVMRP 258—263
DVRMP 237, 263

E

EBGP 193
Echo Reply 68, 75, 76, 78
Echo Request 67, 68, 75—79
Echo-Reply 41, 44
Echo-Request 41, 44
EGP 25, 141, 173, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 188
EIA RS-232C 38
EIA RS-422 38
EIGRP 145, 147, 162—173
ELSR 318, 319
e-mail 118, 371
Encapsulating Security Payload Header 474, 475, 486
ESMTP 372, 374, 377, 390, 391
ESP 304, 307—311, 314
Ethernet 10, 18, 19, 24, 34, 50—55, 58—60, 139, 146, 152, 158, 162, 209, 238, 239
  10BaseT 123
Exterior Gateway Protocol 130, 141, 176

F

FCS 39, 49, 56
FDDI 24, 53, 54, 57, 58
Feasible Distance 164, 169
FEC 316—319
FIFO 269
Fixed-Filter 285—287
Flooding 154
Fragment Header 474, 475, 484, 485
Frame Relay 158
FTP 10, 16, 26, 29, 117, 121, 122, 208, 209, 331, 332, 350, 353, 354, 430, 445, 451—462

G

GLOP Block 236, 238
GMT 415, 425
Graft 258, 261, 262, 263
Graft Ack 262

H

HDLC 10, 37, 39
Header 345, 363
Hello 153, 155, 157, 158, 179, 180
HEMS 98
HINFO 339, 340, 361, 365
Hop-by-Hop 124, 190, 254, 283, 289
  Hop-by-Hop Options 474, 475, 481
HSRP 66, 92—97
HTML 430, 431, 435, 441, 443, 448
HTTP 16, 29, 117, 121, 122, 203, 209, 210, 293, 429—450
  HTTP/1.1 439, 448, 449
HyperText 430, 431

I

IAB 99
IANA 16, 235, 236, 238, 305, 308
IBGP 192, 193
ICMP 10, 25, 65—84, 96, 282
ICV 305—307, 309
IEEE 22
  IEEE-802.3 51, 270, 294
IESG 471
IETF 137, 138, 300, 316, 471, 472
IGMP 241, 244—254, 282
  IGMP v1 245—247
  IGMP v2 244, 246—248
IGP 143, 176, 177, 181, 186, 188, 189, 193
IGRP 131, 143—145, 162, 168, 169
IMAP 400, 402, 403
  IMAP4 28, 117—120, 371, 400—405, 408
Indirect Delivery 124
Interior Gateway Protocol 130, 143, 176
Internal Router 151
INTERNIC 16
IntServ 268, 282
IP 2, 7, 9—27, 33—39
  IPv4 471, 472, 477, 480, 484, 486, 487
  IPv6 471—487
  IP-дейтаграмма 38
  IP-телефон 294
IPCP 37, 38
IPng Directorate 471
IPsec 299, 300, 303, 304, 307, 311—314, 324—326
IPSTB 97
IPX 12, 58, 163
ISDN 24, 45
ISN 211, 215
ISO 3, 6, 7, 101, 294
ITU-T 270
  G.165 296
  G.711 274, 275, 293, 297
  G.723.1 275, 293
  G.726 275, 293
  G.728 276, 293
  G.729 276, 293
  H.221 271
  H.225 271, 274, 277, 278, 280
  H.242 271
  H.245 271—274, 277, 278
  H.261 273
  H.263 273
  H.323 270—281, 291, 292
  H.324 277
  T.120 277

J

Java 122
Join 265

K

KEEPALIVE 195, 197
KERBEROS 404

L

Label 300, 315, 317—319
LAN 2, 34, 56, 149
Last hop router 264
LCP 37, 38, 40—48
LDP 318, 319
Leave Group 245—247, 250, 252
Link-State 128, 147, 149, 150, 154, 155, 159—163
Load Balancing 127, 144
LOCAL_PREF 189, 192
Loc-RIB 194
LSA 151—154, 160, 161
LSR 317, 319, 320

M

μ-law 274
MAC 34, 50, 51, 478
Magic-Number 44
Mainframe 27
Max Response Time 244, 245, 247
MD5 445, 449
Merging 285
Metric 128, 131
MIB 99—101, 104
Microsoft 293, 297, 371, 430
Microsoft Exchange 371
Microsoft Windows 80—82
MIME 372, 375, 376, 378, 380, 390, 392, 403, 407—409, 432, 438, 442, 443
MLP 45, 46, 47, 48, 49, 50
MPLS 300, 315—320, 324, 326, 327
MP-MLQ 275
MRU 41, 46
MTU 22, 124, 144, 168, 484
MULTI_EXIT_DISC 189
Multicast 14, 50, 219, 233, 234, 236, 238, 239, 240, 246, 248, 249, 254, 256, 258, 262, 282, 313, 472, 478, 480
Multilink 45
MX 339, 340, 343, 344, 347, 361, 365—369

N

NBMA 152, 158
NCP 37, 38
Net Meeting 297
Network Layer 6
Newsgroup 414
Next Header 474, 477, 481, 483, 484, 485
Next Hop 124—126, 188, 316
NEXT_HOP 188, 189, 196
NFS 26, 30, 107, 109, 465—468
  NFSv2 468
  NFSv3 468
NIC 331, 359—367
NMS 100, 105
NNTP 28, 117, 121, 411, 413, 415, 422, 423, 426
NOTIFICATION 194, 195, 197, 198
NS 338, 340, 343, 346, 350, 351, 358—362, 366
NSLOOKUP 368, 369
NVT 112—116

O

OPEN 207, 209, 212
ORIGIN 187, 188, 192
OSI 3, 7, 8, 98, 99, 207, 248, 250, 294, 299—302
OSPF 25, 146—159, 161, 162, 164, 169, 172

P

PAP 44, 45
Parameter Problem 68, 71, 72
PathErr 288, 289
PATHPING 81, 82
PathTear 289
PDU 46, 105, 106
Physical Layer 6
PIM 237, 262, 263, 264, 265
PIM-DM 263, 264
Ping 75, 80
Point-To-Multipoint 33
Point-To-Point 33
Poison Reverse 135, 136, 145
Poisoned Reverse 259
Policing 268
POP3 28, 117—120, 371, 393—400, 404
POTS 292, 295, 296, 298
PPP 10, 24, 36—49
Presentation Layer 6
PRI 45
Probe 261, 262
Protocol-Reject 41, 43
Proxy 433, 436, 446, 449
  Proxy ARP 62, 63
Prune 254, 258, 261—263
PSTN 279, 295
PUSH 213

Q

QCIF 273
QCLASS 347, 355

QNAME 347, 351, 355, 363—366
QoS 268, 269, 281, 282, 315
QTYPE 347, 351, 355, 363, 365, 366
Querier 241
Query 243—247, 252, 253
Queue 268
Queuing 268

R

RARP 24, 58, 59, 61, 84—86
RAS 278
Redirect 68, 73—75, 96
Register 265
  STOP 265
Rendezvous Point 264
Report 243—247, 250—253, 261, 262
Reservation-Request 288
ResvErr 288, 289
ResvTear 289
RFC 10, 11, 13
  RFC-1122 79
  RFC-1918 17
  RFC-1752 471
  RFC-1945 442
  RFC-2060 403
  RFC-821 382
  RFC-822 376
  RFC-913 461
RIB 191, 194, 195
RIP 25, 131, 135, 137—149, 161, 162, 169, 172, 173
  RIP II 237
Route Decision 315
Routing Domain 129
Routing Header 474, 475, 478, 483
RPC 8, 26, 30, 65, 66, 98, 107—110, 465—468
RPC_portmapping 468
RPF 257, 259, 263, 264, 266
RR 337—339, 342, 343, 345
RSVP 81, 268, 281—290
RTCP 219, 222, 223, 226—232, 270
RTP 202, 217—226, 228, 270

S

SAD 312, 314
SAP 238
SBELT 355, 358, 366, 367
SDP 236, 238
SDP/SAP 236, 238
Security Association 303, 311, 312, 314
Server-DTP 452—456
Server-PI 451—455
Session Layer 6
SFTP 29, 458, 460, 461
SGMP 98, 99
Shaping 268
Shared Explicit 285, 287
Shared Tree 264
Shortest Path Tree 264
SLIP 34—36
SLIST 355, 358, 359, 367
SMI 99, 103
SMTP 26, 28, 117—119, 209, 210, 341, 371, 372, 377, 380—388, 390—393, 403, 414, 415, 422, 445
Smurf 77—79
SNA 3
SNAME 355, 358
SNMP 26, 30, 36, 65, 98—100, 104—106, 204, 208
SOA 338—341, 347, 350, 353, 360, 361, 365
Source Quench 67, 68, 72, 73
SPARSE 254—256, 262, 264, 265
SPD 312
Split Horizon 135—138, 145
SSL 445
Standby Group 93, 94
STD 56 137, 138
Stub-Area 153
Successor 165, 166
SWS 216
SYN 212, 214, 215
Synchronization source 221

T

TBT 260
TCB 210
TCP 10, 11, 27, 34, 60, 79, 81, 84, 104, 109, 112, 116—118, 121, 202, 205—218, 341, 345, 368, 371, 384, 392—395, 434, 451, 458, 460, 468—474
TCP/IP 1, 2, 7, 10—19, 24—26, 34, 123, 135, 147, 233, 235, 238, 392

Telnet 293
TELNET 10, 29, 30, 65, 66, 111—122, 354
Terminate-Ack 40, 43
Terminate-Request 40, 43, 46
TFTP 29, 458—460
Time Exceeded 68, 70, 71, 81
Time To Live 124
Timestamp Reply 68, 82, 83
Timestamp Request 68, 82, 83
TLV 482, 486
Token Ring 19, 24, 34, 53—57
TOS 21, 144
TRACEROUTE 80, 81
Traffic Engineering 320
Transport Layer 6
Triggered Update 141
TTL 70, 71, 80, 81, 211, 237, 338, 341, 343, 344, 359—363, 365

U

UDP 10, 202—205
UID 402, 403, 406
UIDVALIDITY 402, 405, 406, 409
Unicast 50, 220, 233, 256, 282, 288, 313, 478, 480
UNIX 2, 30, 35, 80, 110, 411, 466
URG 214, 215
URI 429, 430, 432, 435—449
URL 379, 429, 430, 432, 449
URN 429, 430, 432
UseNet 28, 121, 411—416, 418—422
User-DTP 451, 452, 453, 454
User-P, 451—457

V

V.24 10
V.35 38
VAD 276, 296
Virtual Private Network 320
VoIP 269, 291—298, 323
VPN 321—325
  Client 325
VRRP 92, 97

W

WAN 2, 34
Web 122
Web browser 429, 431, 432, 442
Wildcard-Filter 285, 286
WKS 339, 341
WWW 11, 28, 209, 429—432, 441—445

X

X.25 7, 34, 338, 371, 384
XDP 26
XDR 465, 466

А

АДИКМ 275
Адреса:
  класса A 14
  класса B 14
  класса C 14
  класса D 13, 14
АТС 291, 296, 297

В, Г, Д, З

Видеодекодер 273
ВОК 57
Групповые IP-адреса 17
Домен 330, 336, 337, 359
Заголовок 4, 20

И, К, Л

Интернет 1—3, 9—36, 58, 65, 66, 67, 70, 75, 80, 82, 84, 87, 89, 98, 99, 101, 103, 111, 119, 201—203, 206—209, 213, 217, 218, 233, 236, 237, 239, 241, 245, 247, 254, 256, 258
Интерфейс 7, 207
Канальный уровень 6, 9, 20
ЛВС 51, 144

М

Маршрутизатор 24, 73, 94, 95, 98, 125, 131, 132, 134, 140, 159, 162, 172, 180, 183, 185, 187, 194, 241, 245, 247, 259, 264

MAC 84, 85, 86, 93, 94, 96, 250, 251
Метка 319, 336, 351
Многоуровневая модель 3, 4

О

ОЗУ 164

П

Привратник 271, 279
Прикладная программа 3
Протокол 6, 19, 26, 28, 29, 66, 202—222, 304, 307, 319, 334, 345, 371, 380, 381, 386, 387, 393, 400, 401, 403, 408, 411, 422, 451—454, 458—461, 465—468
Процесс 210

С

Сетевой уровень 6
СМОТ 99
Сообщение APP 226
  BYE 226
  RR 226
  SR 226
Стек 7

Т

Транслятор 221
Транспортный уровень 6, 8, 25

У

Узел 336
Уровень:
  представления 6, 8
  приложения 6, 8
  сессии 6, 8

Ф, Х, Ш

Физический уровень 6, 10
Хост 333
Шлюз 240