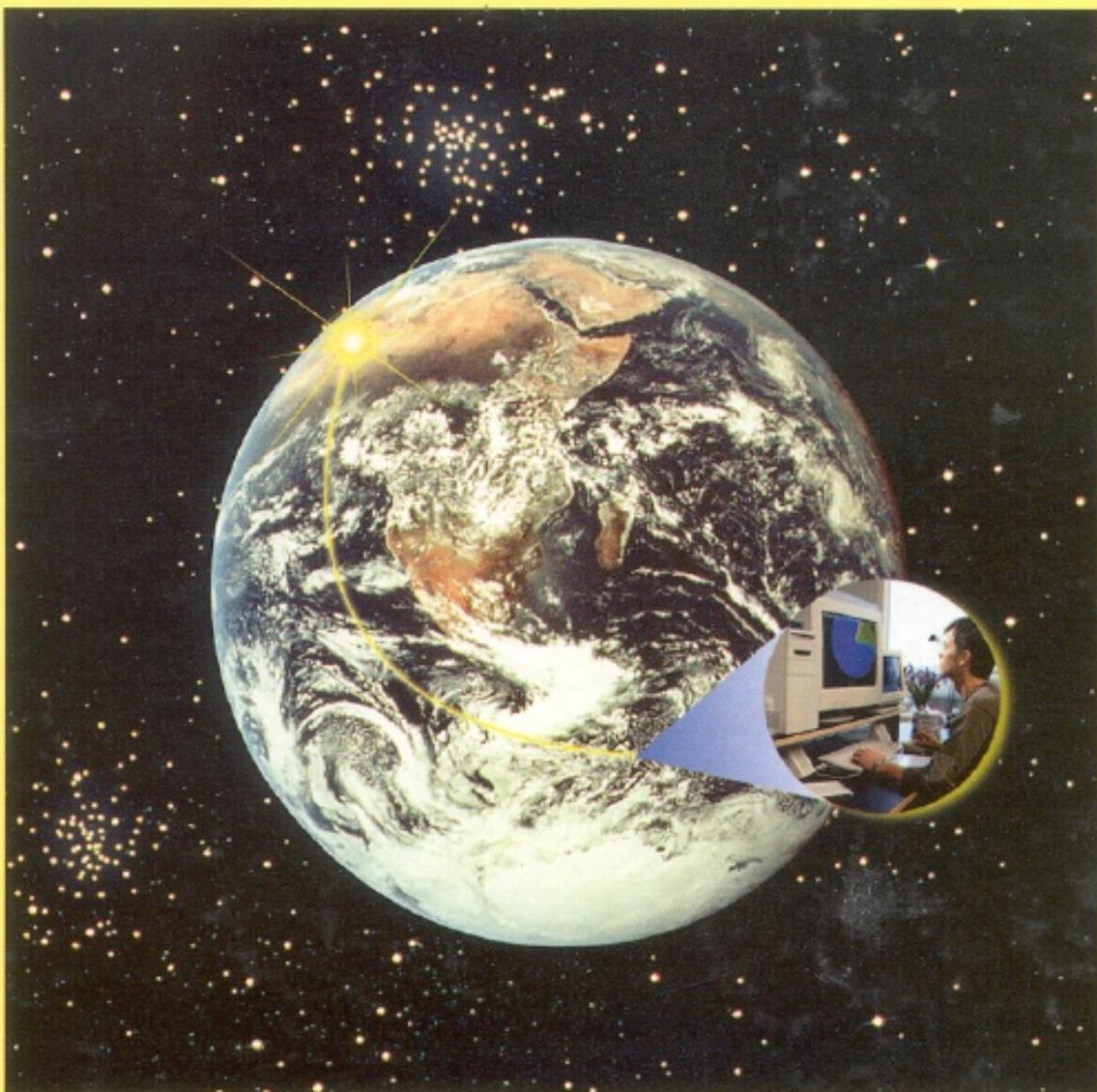


Том  
1

# СЕТИ TCP/IP

ПРИНЦИПЫ, ПРОТОКОЛЫ И СТРУКТУРА

ЧЕТВЕРТОЕ ИЗДАНИЕ



ДУГЛАС Э. КАМЕР





## **Оглавление**

<b>Глава 1. Общие сведения</b>	<b>31</b>
<b>Глава 2. Обзор основных сетевых технологий</b>	<b>49</b>
<b>Глава 3. Основы и структура межсетевого взаимодействия</b>	<b>89</b>
<b>Глава 4. Классовая адресация</b>	<b>99</b>
<b>Глава 5. Преобразование IP-адресов в физические адреса (ARP)</b>	<b>115</b>
<b>Глава 6. Определение IP-адреса при начальной загрузке (RARP)</b>	<b>127</b>
<b>Глава 7. Протокол IP: доставка дейтаграмм без установки соединения</b>	<b>135</b>
<b>Глава 8. Протокол IP: маршрутизация дейтаграмм</b>	<b>157</b>
<b>Глава 9. Протокол IP: обработка ошибок и управляющие сообщения (ICMP)</b>	<b>173</b>
<b>Глава 10. Бесклассовая адресация и подсети (CIDR)</b>	<b>191</b>
<b>Глава 11. Уровни протоколов</b>	<b>223</b>
<b>Глава 12. Передача пользовательских дейтаграмм (UDP)</b>	<b>243</b>
<b>Глава 13. Надежная потоковая транспортная служба (TCP)</b>	<b>255</b>
<b>Глава 14. Основы маршрутизации</b>	<b>305</b>
<b>Глава 15. Маршрутизация между автономными системами (BGP)</b>	<b>323</b>
<b>Глава 16. Маршрутизация внутри автономной системы (RIP, OSPF, HELLO)</b>	<b>349</b>
<b>Глава 17. Режим многоадресатной передачи в объединенной сети</b>	<b>379</b>
<b>Глава 18. Протокол TCP/IP в сетях ATM</b>	<b>417</b>
<b>Глава 19. Протокол мобильной связи с IP-сетями</b>	<b>445</b>
<b>Глава 20. Взаимодействие частных сетей (NAT, VPN)</b>	<b>457</b>
<b>Глава 21. Модель взаимодействия клиент/сервер</b>	<b>471</b>
<b>Глава 22. Интерфейс сокетов</b>	<b>481</b>
<b>Глава 23. Начальная загрузка и автоконфигурация (BOOTP, DHCP)</b>	<b>511</b>

<b>Глава 24. Система доменных имен (DNS)</b>	<b>531</b>
<b>Глава 25. Приложения: удаленный вход в систему (TELNET, rlogin)</b>	<b>555</b>
<b>Глава 26. Приложения: передача файлов и удаленный доступ к ним (FTP, TFTP, NFS)</b>	<b>569</b>
<b>Глава 27. Приложения: система электронной почты (SMTP, POP, IMAP, MIME)</b>	<b>583</b>
<b>Глава 28. Приложения: World Wide Web (HTTP)</b>	<b>601</b>
<b>Глава 29. Приложения: передача голосовых и видеоданных по IP-сети (RTP)</b>	<b>613</b>
<b>Глава 30. Приложения: управление объединенной сетью (SNMP)</b>	<b>629</b>
<b>Глава 31. Обзор структуры протоколов</b>	<b>651</b>
<b>Глава 32. Безопасность в объединенной сети и брандмауэры (IPsec)</b>	<b>657</b>
<b>Глава 33. Будущее протокола TCP/IP (IPv6)</b>	<b>675</b>
<b>Приложение 1. Справочник по документам RFC</b>	<b>699</b>
<b>Приложение 2. Словарь терминов и аббревиатур по сетевым технологиям</b>	<b>787</b>
<b>Список литературы</b>	<b>845</b>
<b>Предметный указатель</b>	<b>852</b>

# **Содержание**

Об авторе	23
Предисловие	23
Предисловие покойного Джона Постела к первому изданию книги	24
Пролог	26
От издательства	29
Мнение специалистов по поводу четвертого издания книги	30
<b>Глава 1. Общие сведения</b>	<b>31</b>
1.1. Необходимость интеграции сетей	31
1.2. Глобальная сеть TCP/IP	32
1.3. Службы Internet	33
1.3.1. Службы Internet уровня приложений	34
1.3.2. Службы Internet сетевого уровня	35
1.4. История развития Internet	37
1.5. Архитектурный совет Internet	39
1.6. Реорганизация архитектурного совета Internet	40
1.7. Сообщество Internet	42
1.8. Запросы на комментарии в Internet	42
1.9. Протоколы Internet и стандартизация	43
1.10. Перспективы роста и технология	43
1.11. Структура книги	45
1.12. Резюме	46
Материал для дальнейшего изучения	46
Упражнения	47
<b>Глава 2. Обзор основных сетевых технологий</b>	<b>49</b>
2.1. Введение	49
2.2. Два подхода к проблеме сетевого взаимодействия	49
2.3. Глобальные и локальные сети	51
2.3.1. Адресация сетевого оборудования	52
2.4. Технология Ethernet	52
2.4.1. Тонкий Ethernet	56
2.4.2. Сеть Ethernet на основе витой пары	57
2.4.3. Пропускная способность сети Ethernet	58
2.4.4. Технология быстрого Ethernet	59
2.4.5. Технология 10/100 Ethernet	59
2.4.6. Гигабитовый Ethernet	60
2.4.7. Отличительные особенности технологии Ethernet	60
2.4.8. Обнаружение коллизий и методика восстановления	61
2.4.9. Адресация оборудования в сетях Ethernet	61
2.4.10. Формат фрейма Ethernet	63
2.4.11. Удлинение сегментов сети Ethernet с помощью повторителей	64
2.4.12. Удлинение сегментов сети Ethernet с помощью мостов	65
2.5. Волоконно-оптические сети FDDI	67
2.5.1. Отличительные особенности сетей FDDI	67
2.5.2. Топология с двумя встречными кольцами	68
2.5.3. Формат фрейма FDDI	69

2.6. Асинхронный режим передачи (ATM)	70
2.6.1. Размер ячейки ATM	70
2.6.2. Сетевая технология, требующая установки соединения	71
2.7. Технологии создания глобальных сетей: ARPANET	71
2.7.1. Система адресации ARPANET	74
2.8. Сетевая программа Национального научного фонда	75
2.8.1. Первый магистральный канал NSFNET	75
2.8.2. Второй магистральный канал NSFNET (1988-1989)	76
2.8.3. Магистральный канал NSFNET 1989-1990 гг.	78
2.9. Сеть ANSNET	78
2.10. Сверхвысокоскоростной магистральный канал (vBNS)	80
2.10.1. Коммерческие магистральные каналы Internet	80
2.11. Другие технологии, использующие протокол TCP/IP	80
2.11.1. Технология X25NET и туннелирование	80
2.11.2. Сети с двухточечным подключением	82
2.11.3. Работа с сетью по коммутируемым телефонным каналам	84
2.11.4. Сетевые технологии с передачей маркера	84
2.11.5. Беспроводные сетевые технологии	85
2.12. Резюме	86
Материал для дальнейшего изучения	86
Упражнения	86
<b>Глава 3. Основы и структура межсетевого взаимодействия</b>	<b>89</b>
3.1. Введение	89
3.2. Взаимодействие на уровне приложений	89
3.3. Взаимодействие на сетевом уровне	90
3.4. Особенности объединенной сети	91
3.5. Структура объединенной сети	92
3.6. Объединение сетей с помощью IP-маршрутизаторов	93
3.7. Преимущества взаимодействия на сетевом уровне	94
3.8. Все сети равноправны	95
3.9. Вопросы, оставшиеся без ответа	95
3.10. Резюме	96
Материал для дальнейшего изучения	96
Упражнения	97
<b>Глава 4. Классовая адресация</b>	<b>99</b>
4.1. Введение	99
4.2. Универсальная система идентификации	99
4.3. Оригинальная классовая система адресации	100
4.4. Адреса определяют сетевые соединения	101
4.5. Сетевые и направленные широковещательные адреса	101
4.6. Ограниченная широковещательная передача	102
4.7. Интерпретация нуля как значения "это"	103
4.8. Механизмы адресации подсетей и суперсетей	103
4.9. Многоадресатная передача	104
4.10. Недостатки IP-адресации	104
4.11. Точечная десятичная форма записи	106
4.12. Адрес петли обратной связи	106
4.13. Соглашение по использованию специальных адресов	107
4.14. Управление адресацией в Internet	107
4.15. Зарезервированные префиксы IP-адресов	108
4.16. Пример	109

4.17. Порядок следования байтов в сети	111
4.18. Резюме	112
Материал для дальнейшего изучения	112
Упражнения	113
<b>Глава 5. Преобразование IP-адресов в физические адреса (ARP)</b>	<b>115</b>
5.1. Введение	115
5.2. Необходимость преобразования адресов	115
5.3. Два типа физических адресов	116
5.4. Преобразование адресов методом прямого отображения	116
5.5. Преобразование адресов методом динамической привязки	117
5.6. Кэширование запросов ARP	118
5.7. Тайм-аут кэширования запросов ARP	119
5.8. Усовершенствования протокола ARP	120
5.9. Взаимосвязь ARP с другими протоколами	120
5.10. Реализация протокола ARP	121
5.11. Инкапсуляция и идентификация пакетов ARP	122
5.12. Формат ARP-сообщений	123
5.13. Резюме	124
Материал для дальнейшего изучения	124
Упражнения	125
<b>Глава 6. Определение IP-адреса при начальной загрузке (RARP)</b>	<b>127</b>
6.1. Введение	127
6.2. Протокол обратного преобразования адресов RARP	128
6.3. Время выполнения транзакции RARP	130
6.4. Основной и резервный RARP-серверы	130
6.5. Резюме	131
Материал для дальнейшего изучения	132
Упражнения	132
<b>Глава 7. Протокол IP: доставка дейтаграмм без установки соединения</b>	<b>135</b>
7.1. Введение	135
7.2. Виртуальная сеть	135
7.3. Структура и принцип действия объединенной сети	136
7.4. Принцип организации служб	136
7.5. Служба доставки, не требующая установки соединения	136
7.6. Назначение протокола IP	137
7.7. Межсетевая дейтаграмма	137
7.7.1. Формат дейтаграммы	138
7.7.2. Способы обработки дейтаграммы и схемы дифференцированного обслуживания	139
7.7.3. Инкапсуляция дейтаграмм	141
7.7.4. Размеры дейтаграммы, MTU и процесс фрагментации	142
7.7.5. Сборка фрагментов	145
7.7.6. Управление фрагментацией	145
7.7.7. Время жизни дейтаграммы (TTL)	147
7.7.8. Остальные поля заголовка дейтаграммы	148
7.8. Параметры IP-дейтаграммы	149
7.8.1. Параметры регистрации маршрута следования	150
7.8.2. Параметры маршрутизации от источника	151
7.8.3. Параметр регистрации временных меток	153
7.8.4. Обработка параметров при фрагментации дейтаграммы	154

7.9. Резюме	154
Материал для дальнейшего изучения	155
Упражнения	155
<b>Глава 8. Протокол IP: маршрутизация дейтаграмм</b>	<b>157</b>
8.1. Введение	157
8.2. Маршрутизация в объединенной сети	157
8.3. Прямая и непрямая доставка дейтаграмм	160
8.3.1. Доставка дейтаграмм по одной физической сети	160
8.3.2. Непрямая доставка	160
8.4. Алгоритм табличной IP-маршрутизации	161
8.5. Маршрутизация “на шаг вперед”	162
8.6. Стандартный маршрут следования дейтаграмм	164
8.7. Маршрутизация отдельных узлов сети	164
8.8. Алгоритм IP-маршрутизации	164
8.9. Маршрутизация с использованием IP-адресов	165
8.10. Обработка входящих дейтаграмм	167
8.11. Создание таблиц маршрутизации	169
8.12. Резюме	169
Материал для дальнейшего изучения	170
Упражнения	170
<b>Глава 9. Протокол IP: обработка ошибок и управляющие сообщения (ICMP)</b>	<b>173</b>
9.1. Введение	173
9.2. Протокол межсетевых управляющих сообщений (ICMP)	173
9.3. Уведомление об ошибках или коррекция ошибок?	174
9.4. Доставка ICMP-сообщений	175
9.5. Формат ICMP-сообщений	176
9.6. Проверка связи с получателем и его состояния (ping)	177
9.7. Форматы запроса на эхо и ответного сообщения	178
9.8. Извещение об отсутствии связи с получателем	178
9.9. Перегрузка сети и управление потоком дейтаграмм	180
9.10. Формат сообщения о подавлении источника данных	181
9.11. Запрос на изменение маршрута следования дейтаграмм от маршрутизатора	181
9.12. Обнаружение замкнутых и слишком длинных маршрутов	183
9.13. Уведомление об остальных проблемах	184
9.14. Синхронизация часов и оценка времени передачи	184
9.15. Информационные запросы и ответы	186
9.16. Определение маски подсети	186
9.17. Поиск маршрутизатора	186
9.18. Запрос адреса маршрутизатора	188
9.19. Резюме	188
Материал для дальнейшего изучения	189
Упражнения	189
<b>Глава 10. Бесклассовая адресация и подсети (CIDR)</b>	<b>191</b>
10.1. Введение	191
10.2. Обзор основных моментов	191
10.3. Принцип минимизации количества адресов сетей	192
10.4. Прозрачная маршрутизация	193
10.5. Агенты ARP	194

10.6. Адресация подсетей	196
10.7. Выбор способа адресации подсетей	198
10.8. Подсети переменной длины	200
10.9. Реализация адресации подсетей с помощью масок	201
10.10. Способы представления масок подсети	201
10.11. Маршрутизация при наличии подсетей	202
10.12. Алгоритм маршрутизации подсетей	204
10.13. Унифицированный алгоритм маршрутизации	205
10.14. Работа с масками подсети	206
10.15. Широковещательная передача в подсетях	206
10.16. Анонимные двухточечные сети	207
10.17. Бесклассовая адресация и суперсети	209
10.18. Влияние суперсети на маршрутизацию	210
10.19. Блоки адресов CIDR и битовые маски	211
10.20. Блоки адресов и форма их записи в CIDR	211
10.21. Пример бесклассовой адресации	212
10.22. Структуры данных и алгоритмы для бесклассового поиска	213
10.22.1. Хеширование и классовая адресация	213
10.22.2. Поиск по длине маски	214
10.22.3. Структуры с использованием двоичных деревьев	214
10.23. Метод наилучшего совпадения и смешанный тип маршрутизации	216
10.23.1. Двоичное дерево PATRICIA и сжатие уровней	217
10.24. Блоки CIDR, выделенные частным сетям	218
10.25. Резюме	218
Материал для дальнейшего изучения	219
Упражнения	220
<b>Глава 11. Уровни протоколов</b>	<b>223</b>
11.1. Введение	223
11.2. Необходимость нескольких протоколов	223
11.3. Концепция уровней протоколов	224
11.4. Разделение функциональных обязанностей между уровнями	226
11.4.1. Семиуровневая модель ISO	226
11.5. Протокол X.25 и модель ISO	227
11.5.1. Пятиуровневая модель протокола TCP/IP	229
11.6. Отличие модели ISO от семейства протоколов TCP/IP	231
11.6.1. Различие в надежности	231
11.6.2. Распределение вычислительных средств и принятие решений	232
11.7. Принцип разделения протоколов на уровни	233
11.7.1. Иерархия уровней протоколов в объединенной сети TCP/IP	234
11.8. Многоуровневая структура сложной сети	235
11.9. Две основные границы раздела многоуровневой модели TCP/IP	237
11.9.1. Граница раздела логических адресов	237
11.9.2. Граница раздела программ операционной системы и пользовательских приложений	238
11.10. Недостатки многоуровневой модели	238
11.11. Основная идея мультиплексирования и демультиплексирования	239
11.12. Резюме	240
Материал для дальнейшего изучения	241
Упражнения	241
<b>Глава 12. Передача пользовательских дейтаграмм (UDP)</b>	<b>243</b>
12.1. Введение	243
12.2. Идентификация конечного получателя	243

12.3.	Протокол передачи пользовательских дейтаграмм (UDP)	244
12.4.	Формат UDP-сообщения	245
12.5.	Псевдозаголовок пользовательской дейтаграммы	246
12.6.	Инкапсуляция и разделение на уровни	247
12.7.	Вычисление контрольной суммы UDP-дейтаграммы при разделении на уровни	248
12.8.	Мультиплексирование и демультиплексирование UDP-дейтаграмм с помощью портов	249
12.9.	Зарезервированные и свободные номера портов UDP	250
12.10.	Резюме	252
	Материал для дальнейшего изучения	253
	Упражнения	253
<b>Глава 13. Надежная потоковая транспортная служба (TCP)</b>		<b>255</b>
13.1.	Введение	255
13.2.	Необходимость в потоковой доставке дейтаграмм	255
13.3.	Особенности надежной службы доставки пакетов	256
13.4.	Обеспечение надежности	258
13.5.	Использование движущихся окон	259
13.6.	Протокол управления передачей (TCP)	262
13.7.	Порты, соединения и конечные точки	263
13.8.	Пассивное и активное открытие соединения	265
13.9.	Сегменты, потоки и порядковые номера	265
13.10.	Окна переменного размера и управление потоком данных	266
13.11.	Формат TCP-сегмента	267
13.12.	Передача данных вне очереди	269
13.13.	Параметр максимального размера сегмента	270
13.14.	Вычисление контрольной суммы TCP-сегмента	271
13.15.	Подтверждение приема и повторная передача	272
13.16.	Время ожидания и повторная передача сегментов	273
13.17.	Точный замер полного времени доставки пакета	275
13.18.	Алгоритм Карна и коррекция тайм-аута	277
13.19.	Обработка большого разброса значений задержки	278
13.20.	Реакция на перегрузку сети	280
13.21.	Перегрузка сети и усечение хвоста очереди	284
13.22.	Произвольное раннее обнаружение (RED)	285
13.23.	Установка TCP-соединения	288
13.24.	Начальный порядковый номер	289
13.25.	Закрытие TCP-соединения	290
13.26.	Сброс TCP-соединения	291
13.27.	Протокол TCP и теория конечных автоматов	291
13.28.	Принудительная доставка данных	293
13.29.	Зарезервированные номера портов протокола TCP	294
13.30.	Производительность протокола TCP	295
13.31.	Синдром полного окна и короткие пакеты	296
13.32.	Как избежать синдрома полного окна	297
13.32.1.	Борьба с синдромом полного окна на стороне получателя	298
13.32.2.	Задержка сигналов подтверждения приема	298
13.32.3.	Борьба с синдромом полного окна на стороне отправителя	299
13.33.	Резюме	301
	Материал для дальнейшего изучения	302
	Упражнения	302

<b>Глава 14. Основы маршрутизации</b>	<b>305</b>
14.1. Введение	305
14.2. Понятие о таблицах маршрутизации	305
14.3. Маршрутизация при неполной информации	307
14.4. Первоначальная структура сети Internet и ее ядро	308
14.5. Базовая система маршрутизации	309
14.6. От сетевого ядра до системы равноправных магистралей	312
14.7. Автоматическое распространение маршрутной информации	314
14.8. Дистанционно-векторная маршрутизация (метод Беллмана-Форда)	315
14.9. Межшлюзовый протокол (GGP)	317
14.10. Упорядочение расстояния	318
14.11. Надежность и протоколы маршрутизации	318
14.12. Маршрутизация, отслеживающая состояние соединения (SPF)	319
14.13. Резюме	320
Материал для дальнейшего изучения	321
Упражнения	321
<b>Глава 15. Маршрутизация между автономными системами (BGP)</b>	<b>323</b>
15.1. Введение	323
15.2. Усложнение структурной модели	323
15.3. Определение практических ограничений на размер группы	324
15.4. Проблема дополнительных переходов	325
15.5. Скрытые сети	327
15.6. Понятие автономной системы	328
15.7. От ядра сети к независимым автономным системам	329
15.8. Протокол внешнего шлюза	330
15.9. Особенности протокола BGP	331
15.10. Функции протокола BGP и виды сообщений	332
15.11. Заголовок BGP-сообщения	333
15.12. Сообщение об открытии BGP-соединения	334
15.13. BGP-сообщение об обновлении	335
15.14. Сжатое представление пары “маска-адрес”	336
15.15. Параметры маршрута протокола BGP	337
15.16. BGP-сообщение о поддержке соединения в активном состоянии	338
15.17. Информация для конечного получателя	339
15.18. Основное ограничение протоколов внешнего шлюза	340
15.19. Арбитражная система маршрутизации в объединенной сети	342
15.20. Формат уведомляющих BGP-сообщений	343
15.21. Децентрализация структуры объединенной сети	344
15.22. Резюме	345
Материал для дальнейшего изучения	345
Упражнения	346
<b>Глава 16. Маршрутизация внутри автономной системы (RIP, OSPF, HELLO)</b>	<b>349</b>
16.1. Введение	349
16.2. Сравнение статических и динамических методов внутренней маршрутизации	349
16.3. Протокол маршрутной информации (RIP)	352
16.3.1. История возникновения протокола RIP	352
16.3.2. Функционирование протокола RIP	353
16.3.3. Решение проблемы медленной сходимости	356
16.3.4. Формат сообщения протокола RIP1	358

16.3.5. Соглашения об адресах протокола RIP1	359
16.3.6. Интерпретация и агрегация маршрутов в протоколе RIP1	359
16.3.7. Расширения протокола RIP2	360
16.3.8. Формат сообщения протокола RIP2	360
16.3.9. Передача сообщений протокола RIP	361
16.3.10. Недостатки метода подсчета количества переходов в протоколе RIP	361
16.4. Протокол HELLO	362
16.5. Оценка задержек и стабильность работы программ маршрутизации	363
16.6. Совместное использование протоколов RIP, HELLO и BGP	365
16.7. Маршрутизация между автономными системами	366
16.8. Программа gated и обмен информацией между автономными системами	367
16.9. Открытый протокол SPF (OSPF)	367
16.9.1. Формат OSPF-сообщения	369
16.9.2. Формат сообщения HELLO протокола OSPF	370
16.9.3. Формат OSPF-сообщения описания базы данных	371
16.9.4. Формат OSPF-запроса о состоянии соединения	372
16.9.5. Формат OSPF-сообщения об обновлении состояния соединения	373
16.10. Маршрутизация при неполной информации	374
16.11. Резюме	375
Материал для дальнейшего изучения	375
Упражнения	376
<b>Глава 17. Режим многоадресатной передачи в объединенной сети</b>	<b>379</b>
17.1. Введение	379
17.2. Аппаратное широковещание	379
17.3. Аппаратная поддержка режима многоадресатной передачи	380
17.4. Многоадресатная передача в сети Ethernet	381
17.5. Многоадресатная передача в протоколе IP	381
17.6. Три аспекта многоадресатной передачи	382
17.7. Многоадресатные IP-адреса	383
17.8. Особенности использования многоадресатного адреса	385
17.9. Преобразование многоадресатного IP-адреса в физический адрес Ethernet	385
17.10. Сетевые узлы и многоадресатная доставка	386
17.11. Область действия многоадресатной рассылки	386
17.12. Поддержка многоадресатной передачи узлом сети	387
17.13. Межсетевой протокол управления группами (IGMP)	388
17.14. Реализация протокола IGMP	389
17.15. Переходы состояния членов группы	390
17.16. Формат IGMP-сообщений	391
17.17. Пересылка многоадресатных пакетов и многоадресатная маршрутизация	393
17.17.1. Необходимость в динамической маршрутизации	393
17.17.2. Недостаточность информации об адресе получателя	394
17.17.3. Произвольные отправители	394
17.18. Основные принципы многоадресатной маршрутизации	394
17.19. Последствия использования TRPF	396
17.20. Дерево многоадресатной передачи	397
17.21. Противоречия в системе многоадресатной маршрутизации	398
17.22. Многоадресатная передача по обратному маршруту	399

17.23. Дистанционно-векторный протокол многоадресатной маршрутизации	400
17.24. Программа mroute	401
17.25. Альтернативные протоколы	404
17.26. Наращающиеся от центра деревья (СВТ)	405
17.27. Независящая от протокола маршрутизации многоадресатная передача (PIM)	406
17.27.1. Уплотненный режим протокола PIM (PIM-DM)	407
17.27.2. Независимость от протокола маршрутизации	407
17.27.3. Разреженный режим протокола PIM (PIM-SM)	407
17.27.4. Переключение с общих деревьев на деревья кратчайшего пути	408
17.28. Многоадресатные расширения протокола OSPF (MOSPF)	409
17.29. Надежная многоадресатная передача и перегрузка уведомлениями	410
17.30. Резюме	412
Материал для дальнейшего изучения	413
Упражнения	413
<b>Глава 18. Протокол TCP/IP в сетях ATM</b>	<b>417</b>
18.1. Введение	417
18.2. Оборудование для сетей ATM	418
18.3. Большие сети ATM	418
18.4. Логическое представление сети ATM	419
18.5. Два типа подключения к сети ATM	420
18.5.1. Постоянные виртуальные каналы	420
18.5.2. Коммутируемые виртуальные каналы	420
18.6. Маршруты, каналы и идентификаторы	421
18.7. Механизм передачи ячеек в сети ATM	422
18.8. Уровень адаптации ATM	423
18.9. Протокол адаптации ATM уровня 5	424
18.10. Сходимость, сегментация и сборка пакетов в протоколе AAL5	425
18.11. Инкапсуляция дейтаграмм и размер MTU в протоколе IP	426
18.12. Тип пакета и мультиплексирование	427
18.13. Механизм привязки IP-адресов в сети ATM	428
18.14. Понятие логической IP-подсети	429
18.15. Управление соединениями	430
18.16. Привязка адресов в логической подсети	431
18.17. Формат пакета протокола ATMAP	431
18.17.1. Формат полей длины в адресе ATM	432
18.17.2. Типы операций протокола ATMAP	433
18.18. Использование ATMAP-пакетов для определения физических адресов	433
18.18.1. Постоянные виртуальные каналы	433
18.18.2. Коммутируемые виртуальные каналы	434
18.19. Сбор информации для базы данных ATMAP-сервера	435
18.20. Достоверность информации, хранящейся на ATMAP-сервере	435
18.21. Достоверность информации, хранящейся на узле и маршрутизаторе	436
18.22. Технологии коммутации IP-пакетов	436
18.23. Принцип работы коммутаторов	437
18.24. Оптимизация пересылки IP-пакетов	438
18.25. Классификация, информационные потоки и коммутация на верхнем уровне	438
18.26. Распространение технологий коммутации	439

18.27. Резюме	440
Материал для дальнейшего изучения	441
Упражнения	441
<b>Глава 19. Протокол мобильной связи с IP-сетями</b>	<b>445</b>
19.1. Введение	445
19.2. Мобильность, маршрутизация и адресация	445
19.3. Свойства мобильного протокола IP	446
19.4. Принципы работы мобильного протокола IP	446
19.5. Особенности адресации мобильного протокола IP	447
19.6. Поиск внешнего агента	448
19.7. Регистрация агента	450
19.8. Формат сообщения о регистрации	450
19.9. Взаимодействие с внешним агентом	451
19.10. Передача и получение дейтаграмм	452
19.11. Проблема двух пересечений	452
19.12. Взаимодействие с компьютерами “домашней” сети	454
19.13. Резюме	455
Материал для дальнейшего изучения	455
Упражнения	456
<b>Глава 20. Взаимодействие частных сетей (NAT, VPN)</b>	<b>457</b>
20.1. Введение	457
20.2. Частные и гибридные сети	457
20.3. Виртуальная частная сеть	458
20.4. Адресация и маршрутизация в виртуальной частной сети	460
20.5. Виртуальная частная сеть с локальными адресами	461
20.6. Преобразование сетевых адресов (NAT)	462
20.7. Создание таблицы преобразования сетевых адресов	463
20.8. Многоадресная NAT	464
20.9. Преобразование сетевого адреса с распределением по портам	465
20.10. Взаимодействие между NAT и ICMP	466
20.11. Взаимодействие между NAT и прикладными программами	466
20.12. Абстрактные домены адресов	467
20.13. Программы slirp и masquerade	468
20.14. Резюме	468
Материал для дальнейшего изучения	469
Упражнения	469
<b>Глава 21. Модель взаимодействия клиент/сервер</b>	<b>471</b>
21.1. Введение	471
21.2. Модель взаимодействия клиент/сервер	471
21.3. Простой пример: эхо-сервер UDP	472
21.4. Служба времени и даты	473
21.4.1. Представление даты и времени	474
21.4.2. Местное и универсальное время	474
21.5. Сложность серверных программ	475
21.6. Сервер RARP	477
21.7. Альтернативные модели взаимодействия	477
21.8. Резюме	478
Материал для дальнейшего изучения	479
Упражнения	479

<b>Глава 22. Интерфейс сокетов</b>	<b>481</b>
22.1. Введение	481
22.2. Принцип ввода-вывода в системе UNIX	481
22.3. Возможности сетевого ввода-вывода в системе UNIX	482
22.4. Концепция сокетов	483
22.5. Создание сокета	484
22.6. Наследование сокетов и завершение процессов	485
22.7. Определение локального адреса	485
22.8. Связывание сокетов с адресами получателей	487
22.9. Пересылка данных через сокет	487
22.10. Получение данных через сокет	489
22.11. Получение локального и удаленного адресов сокета	490
22.12. Получение и установка параметров сокета	491
22.13. Определение длины очереди для сервера	492
22.14. Процесс принятия соединений сервером	492
22.15. Серверы, предоставляющие несколько служб	493
22.16. Получение и назначение имен узлов сети	494
22.17. Получение и назначение имени внутреннего домена узла сети	495
22.18. Функции библиотеки сокетов	495
22.19. Процедуры для преобразования сетевого порядка следования байтов	496
22.20. Подпрограммы обработки IP-адресов	497
22.21. Получение доступа к системе доменных имен	498
22.22. Получение информации об узлах сети	500
22.23. Получение информации о сетях	501
22.24. Получение информации о протоколах	501
22.25. Получение информации о сетевых службах	502
22.26. Пример клиентской части программы	502
22.27. Пример серверной части программы	504
22.28. Резюме	507
Материал для дальнейшего изучения	507
Упражнения	508
<b>Глава 23. Начальная загрузка и автоконфигурация (BOOTP, DHCP)</b>	<b>511</b>
23.1. Введение	511
23.2. Необходимость в альтернативе протоколу RARP	511
23.3. Определение IP-адреса с помощью протокола IP	513
23.4. Принцип повторной передачи в протоколе BOOTP	514
23.5. Формат BOOTP-сообщения	514
23.6. Двухэтапный процесс начальной загрузки	516
23.7. Поле, определяемое производителем	516
23.8. Необходимость динамической конфигурации	518
23.9. Динамическая конфигурация узлов сети	519
23.10. Динамическое назначение IP-адресов	519
23.11. Получение нескольких адресов	520
23.12. Состояние получения адреса	521
23.13. Досрочное прекращение использования адреса	522
23.14. Состояние ожидания продления срока использования адреса	523
23.15. Формат сообщения протокола DHCP	524
23.16. Параметры и типы сообщений протокола DHCP	525
23.17. Переопределение полей DHCP-запроса	526
23.18. Протокол DHCP и доменные имена	526
23.19. Резюме	527

Материал для дальнейшего изучения	528
Упражнения	528
<b>Глава 24. Система доменных имен (DNS)</b>	<b>531</b>
24.1. Введение	531
24.2. Идея присвоения имен машинам	531
24.3. Одноуровневая система имен	532
24.4. Иерархическая система имен	533
24.5. Делегирование полномочий в системе имен	534
24.6. Дальнейшее разделение полномочий	534
24.7. Имена доменов сети Internet	535
24.8. Официальные и неофициальные имена доменов в глобальной сети Internet	536
24.9. Именованные элементы и синтаксис имен	538
24.10. Преобразование доменных имен в адреса	539
24.11. Распознавание доменных имен	541
24.12. Эффективное преобразование имен	542
24.13. Кэширование — залог эффективности	543
24.14. Формат сообщений системы доменных имен	544
24.15. Сжатый формат представления имен	547
24.16. Сокращение доменных имен	547
24.17. Инверсные преобразования	548
24.18. Запросы указателя	549
24.19. Типы объектов и содержимое записи ресурса	549
24.20. Получение полномочий для управления поддоменом	551
24.21. Резюме	552
Материал для дальнейшего изучения	552
Упражнения	552
<b>Глава 25. Приложения: удаленный вход в систему (TELNET, rlogin)</b>	<b>555</b>
25.1. Введение	555
25.2. Дистанционные интерактивные вычисления	555
25.3. Протокол TELNET	556
25.4. Учет особенностей оборудования и программного обеспечения	558
25.5. Передача команд управления удаленным устройством	560
25.6. Привлечение внимания сервера к управляющим кодам	562
25.7. Параметры протокола TELNET	563
25.8. Согласование параметров протокола TELNET	564
25.9. Служба rlogin (BSD UNIX)	565
25.10. Резюме	566
Материал для дальнейшего изучения	567
Упражнения	567
<b>Глава 26. Приложения: передача файлов и удаленный доступ к ним (FTP, TFTP, NFS)</b>	<b>569</b>
26.1. Введение	569
26.2. Удаленный доступ к файлам и их передача	569
26.3. Коллективный доступ в интерактивном режиме	570
26.4. Коллективный доступ к файлу посредством его передачи	571
26.5. FTP — основной протокол передачи файлов семейства TCP/IP	572
26.6. Особенности протокола FTP	572
26.7. Модель FTP-процесса	572
26.8. Назначение номеров портов протокола TCP	573

26.9. Протокол FTP с точки зрения пользователя	574
26.10. Пример анонимного FTP-сеанса	575
26.11. Протокол TFTP	577
26.12. Система NFS	579
26.13. Реализация системы NFS	579
26.14. Дистанционный вызов процедур (RPC)	580
26.15. Резюме	581
Материал для дальнейшего изучения	581
Упражнения	582
<b>Глава 27. Приложения: система электронной почты (SMTP, POP, IMAP, MIME)</b>	<b>583</b>
27.1. Введение	583
27.2. Служба электронной почты	583
27.3. Имена почтовых ящиков и псевдонимы	584
27.4. Расширение псевдонимов и пересылка электронной почты	585
27.5. Взаимосвязь между межсетевым обменом и электронной почтой	586
17.6. Протоколы семейства TCP/IP для службы электронной почты	588
27.7. Адреса электронной почты	589
27.8. Псевдодоменные адреса	590
27.9. Простой протокол передачи электронной почты (SMTP)	591
27.10. Получение электронной почты и протоколы управления почтовыми ящиками	593
27.10.1. Почтовый протокол (POP3)	594
27.10.2. Протокол доступа к сообщениям в сети Internet (IMAP4)	594
27.11. Расширение MIME для данных, представленных не в ASCII-формате	595
27.12. Комбинированные сообщения MIME	596
27.13. Резюме	597
Материал для дальнейшего изучения	598
Упражнения	598
<b>Глава 28. Приложения: World Wide Web (HTTP)</b>	<b>601</b>
28.1. Введение	601
28.2. Важность службы Web	601
28.3. Структурные компоненты	601
28.4. Унифицированные указатели информационного ресурса	602
28.5. Пример HTML-документа	603
28.6. Протокол передачи гипертекста	604
28.7. HTTP-запрос GET	604
28.8. Сообщения об ошибках	605
28.9. Использование постоянных соединений	605
28.10. Длина данных, получаемых в результате работы программы	606
28.11. Кодирование длины и заголовков	607
28.12. Согласование потенциальных возможностей программ	608
28.13. Условные запросы	609
28.14. Поддержка proxy-серверов	609
28.15. Кэширование	610
28.16. Резюме	611
Материал для дальнейшего изучения	611
Упражнения	612

<b>Глава 29. Приложения: передача голосовых и видеоданных по IP-сети (RTP)</b>	<b>613</b>
29.1. Введение	613
29.2. Аудиоклипы и стандарты кодировки	613
29.3. Передача и воспроизведение аудио- и видеопотока	614
29.4. Дрожание и задержка воспроизведения	615
29.5. Протокол передачи данных в реальном масштабе времени (RTP)	616
29.6. Потоки, мультиплексирование и многоадресатная передача	618
29.7. Инкапсуляция RTP-пакетов	618
29.8. Протокол управления в реальном масштабе времени (RTCP)	619
29.9. Работа протокола RTCP	619
29.10. IP-телефония и система сигнализации	621
29.10.1. Стандарты семейства H.323	622
29.10.2. Протокол инициирования сеанса связи (SIP)	623
29.11. Резервирование ресурсов и качество обслуживания	623
29.12. Качество обслуживания, загрузка и максимальная пропускная способность сети	624
29.13. Протокол RSVP	625
29.14. Протокол COPS	626
29.15. Резюме	627
Материал для дальнейшего изучения	627
Упражнения	628
<b>Глава 30. Приложения: управление объединенной сетью (SNMP)</b>	<b>629</b>
30.1. Введение	629
30.2. Уровень протоколов управления	629
30.3. Структурная модель	630
30.4. Структура протоколов	632
30.4.1. Стандартный протокол сетевого управления	632
30.4.2. Стандарт для управляющей информации	632
30.5. Примеры переменных базы MIB	633
30.6. Структура управляющей информации	635
30.7. Формальные определения с использованием ASN.1	635
30.8. Структура и представление имен объектов базы MIB	636
30.9. Простой протокол сетевого управления (SNMP)	640
30.9.1. Выполнение поиска по таблицам при использовании имен	642
30.10. Формат SNMP — сообщений	642
30.11. Пример закодированного SNMP-сообщения	645
30.12. Новые возможности протокола SNMP версии 3	647
30.13. Резюме	647
Материал для дальнейшего изучения	648
Упражнения	649
<b>Глава 31. Обзор структуры протоколов</b>	<b>651</b>
31.1. Введение	651
31.2. Взаимосвязи между протоколами	651
31.3. Модель песочных часов	653
31.4. Доступ со стороны прикладных программ	653
31.5. Резюме	654
Материал для дальнейшего изучения	655
Упражнения	655

<b>Глава 32. Безопасность в объединенной сети и брандмауэры (IPsec)</b>	<b>657</b>
32.1. Введение	657
32.2. Безопасность ресурсов	657
32.3. Информационная стратегия	659
32.4. Безопасность глобальной сети Internet	659
32.5. Безопасность протокола IP (IPsec)	659
32.6. Заголовок аутентификации семейства протоколов IPsec	660
32.7. Ассоциация обеспечения безопасности	661
32.8. Инкапсуляция зашифрованных данных в протоколе IPsec	662
32.9. Аутентификация и изменяемые поля заголовка	663
32.10. Туннелирование по закрытому каналу на основе протоколов IPsec	664
32.11. Обязательные алгоритмы шифрования	664
32.12. Механизм защищенных сокетов	665
32.13. Брандмауэры и доступ к сети Internet	665
32.14. Множественные соединения и самые слабые связи	665
32.15. Реализация брандмауэра	666
32.16. Фильтры пакетов	666
32.17. Безопасность и параметры фильтра пакетов	667
32.18. Влияние ограничения доступа на клиентов	668
32.19. Доступ к Internet в обход брандмауэра через proxy-серверы	669
32.20. Подробнее о структуре брандмауэра	670
32.21. Тупиковая сеть	671
32.22. Альтернативный вариант реализации брандмауэра	671
32.23. Текущий контроль и регистрация	672
32.24. Резюме	673
Материал для дальнейшего изучения	673
Упражнения	674
<b>Глава 33. Будущее протокола TCP/IP (IPv6)</b>	<b>675</b>
33.1. Введение	675
33.2. Зачем что-то менять?	675
33.3. Новые правила	676
33.4. Причины пересмотра стандарта IPv4	676
33.5. Путь к новой версии протокола IP	677
33.6. Название следующей версии протокола IP	678
33.7. Свойства протокола IPv6	678
33.8. Общая структура дейтаграммы протокола IPv6	679
33.9. Формат основного заголовка дейтаграммы IPv6	679
33.10. Дополнительные заголовки дейтаграммы протокола IPv6	681
33.11. Анализ дейтаграммы протокола IPv6	681
33.12. Фрагментация и сборка дейтаграммы в протоколе IPv6	682
33.13. Важность сквозной фрагментации	683
33.14. Маршрутизация от источника в протоколе IPv6	684
33.15. Параметры протокола IPv6	684
33.16. Размер адресного пространства протокола IPv6	686
33.17. Шестнадцатеричная запись с разделением двоеточием	686
33.18. Три основных типа адресов протокола IPv6	687
33.19. Взаимозаменяемость широковещательной и многоадресатной передачи	688
33.20. Проблема выбора, стоящая перед разработчиками	688
33.21. Проект распределения адресного пространства протокола IPv6	689
33.22. Кодирование адресов протокола IPv4 и переход к адресам IPv6	690
33.23. Неопределенные адреса и адреса петли обратной связи	691

33.24. Иерархия одноадресатных адресов	692
33.25. Структура сгруппированных глобальных одноадресатных адресов	693
33.26. Идентификаторы интерфейса	694
33.27. Дополнительный уровень иерархии	695
33.28. Локальные адреса	695
33.29. Автоконфигурация и перенумерация сетей	695
33.30. Резюме	696
Материал для дальнейшего изучения	697
Упражнения	697
<b>Приложение 1. Справочник по документам RFC</b>	<b>699</b>
Введение	699
Значение документов, содержащих требования к узлам сети и маршрутизаторам	700
Магические числа RFC	701
Как найти нужный документ RFC в Internet	701
Поиск нужного документа в списке RFC	701
Список RFC, упорядоченный по темам	702
Документы RFC, упорядоченные по основным категориям и подтемам	703
1. Административные	703
2. Документы с требованиями и пересмотры основных протоколов	710
3. Уровень сетевого интерфейса (см. также раздел 8)	710
4. Уровень Internet	714
5. Уровень узла сети	725
6. Уровень приложений	730
7. Программная документация	767
8. Специфические сети (см. также раздел 3)	768
9. Система измерений	773
10. Конфиденциальность, безопасность и аутентификация	775
11. Опыт использования сети и демонстрация ее работы	779
12. Документация сетевого центра	779
13. Стандарты протоколов, принятые другими группами, заинтересованными в использовании Internet	779
14. Взаимодействие с другими приложениями и протоколами	781
15. Разнообразная информация	782
16. Список неиспользуемых номеров документов RFC	786
<b>Приложение 2. Словарь терминов и аббревиатур по сетевым технологиям</b>	<b>787</b>
Терминология TCP/IP	787
Словарь сетевых терминов и аббревиатур, собранных в алфавитном порядке	787
<b>Список литературы</b>	<b>845</b>
<b>Предметный указатель</b>	<b>852</b>



*Посвящается Крис*

## Об авторе



Доктор *Дуглас Э. Камер* (Douglas Comer) — признанный в мире специалист по протоколу TCP/IP и Internet. Он один из тех, кто стоял у истоков Internet и внес огромный вклад в развитие и становление глобальной сети в конце 70-х и начале 80-х годов XX века. Он был членом Архитектурного совета Internet (Internet Architecture Board, или IAB) — группы специалистов, определявших стратегию развития глобальной сети. Дуглас также был председателем технического и членом исполнительного комитета CSNET.

Мистер Камер консультирует различные компании по вопросам разработки и реализации сетей, а также проводит по всему миру семинары по протоколу TCP/IP и построению сетей на его основе. Его лекции рассчитаны как на профессионалов, так и на рядовых пользователей Internet. Дуглас написал операционную систему Xinu и создал собственную реализацию протокола TCP/IP. Все это он отразил в своих книгах. Созданное им программное обеспечение используется во многих коммерческих продуктах.

Мистер Камер является профессором факультета информатики университета Пердью (Purdue University); он преподает и занимается научно-исследовательской работой в области локальных и глобальных компьютерных сетей и операционных систем. Кроме написания серии научно-технических книг, ставших бестселлерами во всем мире, Дуглас выполняет редакторскую работу в североамериканском журнале *Software — Practice and Experience*. Мистер Камер — действительный член ассоциации ACM (Association of Computing Machinery — ассоциация пользователей вычислительных машин). Дополнительную информацию о нем вы можете получить в Internet по адресу:

[http://www.cs.purdue.edu/people/comer.](http://www.cs.purdue.edu/people/comer)

## Предисловие

Вы держите в руках четвертое издание эпохальной книги, ознаменовавшей наступление эры Internet. Разработка протоколов для Internet началась примерно в 1974 году, а широкое использование — с начала 80-х годов. Однако до 1987 года не существовало хороших инструкций и описаний протоколов, в которых бы объяснялись принципы их работы и методика написания программ. Конечно, документы стандартов для TCP, IP и других основных протоколов Internet были созданы еще на заре развития глобальной сети. Но чтобы применить их в деле и создать реально действующий протокольный стек, разработчик должен был обладать немалым багажом знаний и опыта. Это была тайна за семью печатями, доступная только узкому кругу избранных. Такое положение дел, естественно, не могло устроить профессионалов, поскольку они были вынуждены по крупицам собирать необходимый материал и создавать на его основе собственные рабочие документы и инструкции. Сетевой мир замер в ожидании хорошей книги, описывающей протокол TCP/IP, которая бы послужила толчком к широкому использованию разработанных ранее протоколов.

И в этот момент Дуглас решил написать книгу.

В ожидании выхода книги мы, подшучивая над ним, сравнивали количество выпущенных книг по разной тематике, справедливо полагая, что это может служить мерилом успеха. Я лично обошел несколько книжных магазинов и подсчитал, сколько там было книг, посвященных проектированию компиляторов.

(Впрочем, тема по компиляторам сейчас снова стала популярной и, видимо, пришло время заново пересчитать все книги.) Тогда книги по “созданию хороших компиляторов” было в избытке, имелись даже книги по принципам проектирования баз данных. Однако я не нашел ни одной книги, посвященной протоколу TCP/IP. И вот, наконец, такая книга вышла.

Естественно, моего мнения о том, что это эпохальная книга, может быть недостаточно, чтобы склонить вас к покупке. Коллекционеры, возможно захотят приобрести первое издание книги, но оно было актуальным 12 лет назад, а это огромный срок для Internet. За эти годы в мире Internet очень многое изменилось, поэтому сейчас выходит уже четвертое издание книги. Это время было богато событиями: мы многому научились, стремительно развивалась Internet, появились совершенно новые сетевые протоколы. Все это заставило Дугласа переписывать книгу три раза. Количество ее переизданий служит ярким свидетельством того, насколько обширна охватываемая книгой область знаний, как быстро все изменяется в мире Internet и сколько нужно приложить усилий для того, чтобы книга не утратила актуальности. В четвертое издание включен совершенно новый материал, а также весь тот багаж знаний, полученных нами на протяжении многих лет работы в этой области.

За 12 лет многое изменилось не только в мире Internet. Все мои коллеги заметно постарели, некоторые из них уже умерли. Предисловие к первому изданию этой книги написал Джон Постел (Jon Postel), один из пионеров освоения Internet. К моему большому сожалению Джон умер осенью 1998 года. Ниже мы полностью приводим его предисловие. Обратите внимание: многие вещи в нем до сих пор актуальны, хотя и многое уже значительно изменилось. Сегодня эта книга — одна из самых читаемых, поскольку в ней подробно описано семейство протоколов TCP/IP, а также приведены основные сведения по другим коммуникационным протоколам. Но в 1987 году Джон писал: “В настоящее время системы компьютерной связи и телефонные сети никак не связаны между собой иdezintegrirovani. Целью создания комплексов вычислительных устройств и объединения их в сеть является получение единой мощной компьютерной коммуникационной инфраструктуры. Эти требования были положены в основу при разработке протокола TCP/IP”. Всего 12 лет назад компьютерные сети были полностью dezintegrirovani, а сегодня Internet объединяет весь мир. И основным средством такого объединения, можно сказать, ядром Internet, т.е. тем, что заставляет все это работать, является протокол TCP/IP. И эта книга продолжает оставаться бесценным источником информации для его изучения.

— Дэвид Кларк (*David Clark*),  
Массачусетский технологический институт,  
декабрь 1999 года

## Предисловие покойного Джона Постела к первому изданию книги

В этой книге профессор Дуглас Камер собрал максимум информации по протоколу TCP/IP и описал принципы его работы. Мне часто доводилось встречать пользователей, которые были заняты поисками статьи, отчета или книги, в которых бы излагались основы протокола TCP/IP. И вот, наконец, появилась такая книга. Написать книгу, целиком посвященную протоколу TCP/IP, так, чтобы она была доступна для понимания простым смертным, — задача не из легких. Успешно комбинируя описание основ системы компьютерной связи с примерами использования протокола TCP/IP, Дуглас Камер создал легко читаемую и интересную книгу.

Хотя в книге во всех деталях рассмотрено семейство протоколов TCP/IP, она является хорошей отправной точкой для изучения общих принципов и протоколов компьютерных средств связи. Все дело в том, что основы структуры, иерархии, уплотнения (мультиплексирования), инкапсуляции, назначения и преобразования адресов, маршрутизации и присвоения имен одинаковы для любого семейства протоколов, хотя, конечно, в деталях существуют некоторые отличия (см. главы 3, 10, 17 и 18)<sup>1</sup>. Коммуникационные компьютерные протоколы сами по себе ничего не делают. Подобно операционным системам, они относятся к средствам обслуживания прикладных программ (процессов). Запрос на установку соединения поступает от прикладного процесса. Только ему известны адреса отправителя и получателя передаваемых данных. Что касается разных уровней сетевых протоколов, то их можно сравнить с иерархической структурой операционной системы (особенно, ее файловой системы). Структура протоколов чем-то напоминает архитектуру операционной системы. В этой книге Дуглас Камер выбрал восходящий принцип изложения материала. Он начинает рассмотрение с описания физических основ работы сетей и по мере изложения материала про-двигается вверх по абстрактным уровням вплоть до уровня приложений.

Поскольку обмен данными осуществляется по инициативе прикладных процессов, а выполняет его семейство протоколов, то протокол TCP/IP можно считать своеобразным механизмом межпроцессного взаимодействия, или IPC (*interprocess communication*). И хотя сейчас разрабатываются механизмы передачи сообщений на основе протокола IP, аналогичные тем, что используются в операционных системах и при вызове процедур, в этой книге основное внимание сосредоточено на более традиционных приложениях, в которых используются дейтаграммы протокола UDP или форма логической связи протокола TCP, применяемая для межпроцессного взаимодействия (см. главы 11, 12, 17, 18 и 19).

Одной из ключевых целей разработки протокола TCP/IP является возможность создания глобальной сети. Мощность коммуникационных систем напрямую зависит от числа элементов, составляющих эти системы. Так, например, телефонная сеть чрезвычайно удобна в использовании, поскольку, с точки зрения пользователя, практически все абоненты находятся в одной сети. В настоящее время системы компьютерной связи и телефонные сети никак не связаны между собой и дезинтегрированы. Целью создания комплексов вычислительных устройств и объединения их в сеть является получение единой мощной компьютерной коммуникационной инфраструктуры. Эти требования были положены в основу при разработке протокола TCP/IP. В основе любой глобальной сети лежит система адресации (см. главы 4, 5 и 6) и универсальный протокол — протокол Internet, или протокол межсетевого взаимодействия (*Internet Protocol*, или IP) (см. главы 7, 8 и 9).

Для создания глобальной сети нужно каким-то образом объединить самостоятельные локальные сети. В качестве связующих устройств используются так называемые *шлюзы* (*gateways*). Каждый шлюз должен работать по определенной программе, управляющей процессом пересылки данных из одной сети в другую. Данные в сетях пересыпаются в виде IP-дейтаграмм, а получатель определяется по IP-адресу. Решение о том, как нужно пересыпать данные между сетями (или маршрутизировать их), шлюз принимает на основе IP-адреса получателя и некоторой информации о физическом подключении сетей, составляющих Internet. Процедуры распространения информации о подключении сетей среди шлюзов называются *алгоритмами маршрутизации*. Они являются предметом детального изучения (см. главы 13, 14, 15 и 16).

---

<sup>1</sup> По сравнению с первым изданием, нумерация глав изменилась. — *Прим. ред.*

Подобно всем коммуникационным системам, семейство протоколов TCP/IP — незавершенная система. Эволюция протоколов продолжается: в них вносятся изменения, которые должны удовлетворить возникающие потребности и реализовать новые возможности. Таким образом, книга в некотором смысле отражает состояние протокола TCP/IP к 1987 году. И, как отметил Дуглас Камер, большая часть работы еще не закончена (см. главу 20).

В конце каждой главы, как правило, приводится раздел, озаглавленный “Материал для дальнейшего изучения”, в котором даны ссылки на дополнительный материал. Чаще всего вам будут встречаться ссылки на набор документов RFC. В нем описаны идеи работы и спецификации протоколов, созданные сообществом разработчиков и исследователей протокола TCP/IP. Благодаря этим документам информация становится доступной широкому кругу людей. Доступность базовой информации о протоколах, а также сведений об их предыдущих реализациях обуславливает их широкое применение. Принятая в сообществе разработчиков открытая публикация рабочей информации нехарактерна для исследовательских кругов, однако она, несомненно, приносит огромную пользу в процессе создания систем компьютерной связи (см. приложение 3).

Автор предпринял попытку собрать воедино разрозненную информацию об архитектуре сетей TCP/IP и их протоколах и сделал ее доступной широкому кругу читателей. Публикация книги — значительная веха в эволюции систем компьютерной связи.

— Джон Постел,  
разработчик протокола IP и  
исполняющий обязанности главного архитектора Internet,  
декабрь 1987 года

## Пролог

Бурный рост Internet продолжается. Когда пять лет назад я писал третье издание этой книги Internet объединяла 4,8 млн компьютеров (на момент написания первого издания книги в глобальной сети было всего 5000 компьютеров!). Сегодня Internet насчитывает более 56 млн компьютеров, а это означает, что по сравнению с 1995 годом, она увеличилась десятикратно. Еще в начале 90-х годов XX века те немногие, кому приходилось сталкиваться с Internet, восхищались масштабами и сложностью этого проекта. А теперь Internet проникла практически во все сферы человеческой деятельности.

Семейство протоколов TCP/IP обладает высокой устойчивостью к изменениям. За последние два десятилетия оно пережило экспоненциальный рост сети и связанное с этим резкое увеличение объемов передаваемых данных. Эти протоколы продолжают использоваться в новых высокоскоростных сетевых технологиях, на их основе создаются приложения, о которых раньше даже нельзя было мечтать. Естественно, семейство протоколов в целом не может оставаться неизменным. Появляются новые протоколы, разрабатываются новые сетевые технологии, все это не может не отразиться на существующих стандартах.

Текст этого издания книги был существенно переработан, в нее также включен новый материал, описывающий современные технологии и происходящие в них изменения. Например, сейчас стала широко использоваться бесклассовая система адресации, поэтому в описание технологии пересылки IP-пакетов включена методика бесклассового поиска. Кроме того, в главах, посвященных протоколу IP, описаны схемы дифференцированного обслуживания (Differentiated Services, или DiffServe) для классов служб, а также способы определения MTU и ненумерованные (анонимные) сети. В главах, посвященных протоколу TCP,

рассматривается методика произвольного раннего обнаружения (Random Early Drop, или RED). В главы, где описываются протоколы внешней маршрутизации (exterior routing), были внесены изменения (теперь во главу угла поставлен протокол BGP). Описание многих протоколов, таких как RIP, IGMP, SNMP и IPv6, было существенно переработано (добавлена информация о новых версиях протоколов и внесенных в них изменениях). И наконец, в главе, посвященной проблемам безопасности, рассматривается протокол IPsec.

В четырех новых главах содержатся подробные сведения о важных разработках. В главе 19 рассматривается технология мобильного IP, которая позволяет перемещать компьютер из одной физической сети в другую без смены его IP-адреса. В главе 20 описаны две технологии, используемые для подключения частных локальных сетей (intranets) к глобальной сети Internet. Речь идет о виртуальной частной сети (Virtual Private Network, или VPN) и преобразовании сетевых адресов (Network Address Translation, или NAT). Каждая из этих технологий широко используется для решения определенного круга проблем. В главе 28 описан язык гипертекстовой разметки HTML и протокол HTTP, которые лежат в основе самой популярной службы Internet — World Wide Web. В главе 29 вниманию читателя предлагается новая увлекательная область — передача по IP-сетям голосовых и видеоданных в реальном масштабе времени. Здесь рассматривается протокол RTP, позволяющий получателю согласовывать во времени потоки таких данных и воспроизводить их. Кроме того, рассматриваются протоколы RSVP и COPS, используемые для резервирования ресурсов, а также семейство протоколов H.323, применяемых в IP-телефонии.

В целом общее содержание и структура четвертого и третьего изданий книги совпадают. Большая часть материала посвящена концепции межсетевого взаимодействия, в частности технологии создания глобальных сетей TCP/IP. Эта концепция относится к высокому уровню абстракции, позволяющему скрыть от конечного пользователя сложность коммуникационных технологий, лежащих в ее основе. Речь идет о деталях использования сетевого оборудования и создании коммуникационного интерфейса верхнего уровня для сетевых приложений. Будут рассмотрены методы объединения сетей и принципы работы используемых при этом протоколов, благодаря которым сложные объединенные сети функционируют как одна унифицированная коммуникационная система. Также будет описано использование глобальной сети для организации распределенных вычислений.

Прочитав эту книгу, вы поймете, почему стало возможным объединение различных физических сетей в одну согласованную систему, как работают протоколы Internet в такой среде и как прикладные программы используют полученную глобальную систему. На конкретном примере мы подробно рассмотрим работу глобальной сети Internet на основе протокола TCP/IP, принципы построения ее систем маршрутизации и поддерживающие протоколы. Кроме того, будут описаны некоторые ограничения, имеющие место при использовании глобальной сети.

Книга задумывалась как учебник для вузов и справочное руководство для специалистов, поэтому она написана на высоком профессиональном уровне. Специалисты могут почертнуть в ней подробное описание технологии сетей TCP/IP и структуры Internet. Автор книги неставил перед собой цель заменить описание существующих стандартов протоколов. Тем не менее книгу можно рассматривать как великолепную отправную точку в изучении технологии глобальных сетей, поскольку в ней изложены основы и сделан акцент на принципах их работы. Кроме того, книга дает читателю ориентиры для поиска дополнительной информации, которые было бы трудно получить на основе изучения отдельных стандартов протоколов.

Материала книги более чем достаточно для преподавания односеместрового курса по сетям в вузе, а также спецкурсов для студентов старших курсов и аспирантов. При наличии дополнительного материала в виде готовых программных проектов и соответствующей литературы программу курса можно расширить до двух семестров. Во время преподавания материала студентам младших курсов многие детали можно опустить. Аудитория должна уловить основные идеи, описанные в книге, уметь их сформулировать на семинарах и использовать на практических занятиях. Студенты старших курсов могут использовать материал книги как базу для проведения дальнейших исследований. Они должны достаточно глубоко проработать материал и решить предлагаемые упражнения или поставленную перед ними задачу, которая требует проведения самостоятельного исследования в предметной области и преодоления возможных трудностей и ловушек. В большинстве упражнений такие трудности есть. Поэтому слушателю придется прочитать соответствующие документы стандартов, а также применить всю свою созидательную энергию для анализа полученных результатов.

В любом случае преподаватель должен заострить внимание студентов на определенных вещах и помочь им развить интуицию. Поэтому я всячески поддерживаю тех преподавателей, которые придумывают такие проекты, вынуждающие студентов применять на практике службы и протоколы Internet. Например, в односеместровом спецкурсе по глобальным сетям, который преподается выпускникам университета Пердью, рассматривается проект создания IP-маршрутизатора. Мы подготовили необходимое оборудование и исходный код программ для операционной системы, включая драйверы сетевых интерфейсов. Студенты должны самостоятельно создать маршрутизатор, объединяющий три сети с разными значениями MTU. Преподавание курса ведется очень строго, студенты работают в группах, и результат всего этого впечатляет (наших выпускников с удовольствием берут к себе на работу многие крупные корпорации). Поскольку наша учебная сеть полностью изолирована от промышленных компьютерных сетей, эксперименты с маршрутизатором можно считать безопасными. Однако мы обнаружили, что студенты работают с большим энтузиазмом (да и польза от такой работы существенно выше), когда у них есть полноценный доступ к функционирующей глобальной сети TCP/IP.

Книга разбита на четыре большие части. Главы 1 и 2 можно считать обзорными. В них описываются существующие сетевые технологии. В частности, в главе 2 сделан обзор сетевого аппаратного обеспечения и физических сред передачи информации. Наша цель — дать читателю необходимые знания и представления о существующих возможностях, не рассматривая подробно особенности оборудования. В главах 3–13 описана глобальная сеть TCP/IP с точки зрения узлового компьютера (хоста). Здесь приведена информация по протоколам, используемым хостом, и рассказано, как все это работает. В этих главах рассмотрены основы адресации и маршрутизации в Internet, а также приведены сведения по иерархической структуре протоколов. В главах 14–20 и 32 описана структура Internet при взгляде на нее как бы со стороны. Здесь речь пойдет об общих принципах маршрутизации в Internet и протоколах, используемых для обмена маршрутной информацией между узлами сети. И наконец, в главах 21–31 обсуждаются службы Internet, относящиеся к уровню приложений. Здесь описывается модель взаимодействия клиент/сервер и приводятся примеры программ клиентской и серверной части.

Материал книги подается в восходящем порядке. Сначала приводятся необходимые сведения по аппаратному обеспечению, а затем на их основе строится материал последующих глав. При этом функциональность изложения все время расширяется. С таким подходом должны быть хорошо знакомы разработчики

программ для Internet, он практически всегда используется при создании сетевых приложений. Концепция иерархии протоколов не упоминается вплоть до 11 главы. При обсуждении уровней иерархии подчеркивается разница между концептуальными уровнями функциональности и реальным программным обеспечением, реализующим эти иерархические протоколы, в котором на каждом уровне может находиться несколько объектов.

Для понимания материала книги достаточно скромных познаний в предметной области. Предполагается, что читатель знаком с основами построения компьютерных систем и имеет понятие о базовых структурах данных, таких как стеки, очереди и деревья. Кроме того, читатель должен знать структуру программного обеспечения операционной системы, поддерживающей многозадачность, и иметь общие представления о прикладных программах, запускаемых пользователем для выполнения вычислений. От читателя не требуется глубоких познаний в области математики и теории информации, не нужно также знание теорем, применяемых в системах передачи данных. В книге физический уровень сети описан в виде черного ящика, на основе которого строится система межсетевого взаимодействия; устанавливаются четкие принципы проектирования и обсуждаются причины и следствия.

В заключение хочу поблагодарить всех тех, кто внес свой посильный вклад в создание книги. Огромную помощь в редактировании этого издания и классификации документов RFC оказал Майкл Евангелиста (Michael Evangelista). Пример использования протокола SNMPv3 предоставил Джейф Кейс (Jeff Case). Джон Лин (John Lin) и Денис Тотин (Dennis Totin) высказали мне свое мнение по поводу некоторых новых глав. Корректируя текста книги выполнили Джин Жанг (Jin Zhang), Кечунь Хе (Kechun He) и Сейра Штейнбрюк (Sara Steinbrueck). Особая благодарность моей жене и партнеру Крис, чье скрупулезное редактирование существенно улучшило книгу.

*Дуглас Э. Камер, январь 2000 года*

## От издательства

Вы, читатель этой книги, и есть главный ее критик и комментатор. Мы ценим ваше мнение и хотим знать, что было сделано нами правильно, что можно было сделать лучше и что еще вы хотели бы увидеть изданным нами. Нам интересно услышать и любые другие замечания, которые вам хотелось бы высказать в наш адрес.

Мы ждем ваших комментариев и надеемся на них. Вы можете прислать электронное письмо или просто посетить наш Web-сервер, оставив свои замечания, — одним словом, любым удобным для вас способом дайте нам знать, понравилась или нет вам эта книга, а также выскажите свое мнение о том, как сделать наши книги более полезными для вас.

Посылая письмо или сообщение, не забудьте указать название книги и ее авторов, а также ваш e-mail. Мы внимательно ознакомимся с вашим мнением и обязательно учтем его при отборе и подготовке к изданию последующих книг. Наши координаты:

E-mail: [info@williamspublishing.com](mailto:info@williamspublishing.com)

WWW: <http://www.williamspublishing.com>

## **Мнение специалистов по поводу четвертого издания книги**

“Это та книга, из которой можно почерпнуть информацию об основополагающих принципах и последних изменениях технологии TCP/IP. Она обязательно должна быть у каждого специалиста по сетям.”

— *Доктор Ральф Дром (Ralph Droms), профессор Бакнельского университета (Bucknell University)*

“Как только Нобелевский комитет обратит внимание на Internet, Дуглас обязательно получит премию в области литературы. Эта книга — обновленное издание классического труда — самый лучший источником информации для желающих освоить технологию Internet.”

— *Доктор Пол В. Мокапетрис (Paul V. Mockapetris), изобретатель системы доменных имен (Domain Name System, или DNS)*

“Лучшая книга с описанием технологии TCP/IP, которую я когда-либо читал. Доктор Камер объясняет сложные понятия ясно и четко, дополняя их превосходно подобранными иллюстрациями и примерами.”

— *Доктор Джон Лин (John Lin), Bell Laboratories*

“Четвертое издание этой книги по-прежнему остается самым полным справочником по протоколам Internet, а доктор Камер — столпом глобальных сетей.”

— *Доктор Винтон Церф (Vinton Cerf), вице-президент корпорации MCI WorldCom*

“Среди множества книг, посвященных протоколу TCP/IP, особое место занимает книга Дугласа Камера. В ней вы найдете ответы практически на все вопросы, касающиеся технологии Internet.”

*Доктор Лаймен Чапин (Lyman Chapin), старший научный сотрудник корпорации BBN Technologies*

# 1

## Общие сведения

### 1.1. Необходимость интеграции сетей

Обмен информацией через Internet стал неотъемлемой частью нашей жизни. В World Wide Web можно найти самую разную информацию, например прогноз погоды, данные об урожае прошлого года, курсы акций и расписание полетов самолетов. Группы добровольцев создают электронные списки рассылки, позволяющие распространять информацию, которая относится к сфере их интересов. Партнеры по бизнесу и коллеги по работе обмениваются друг с другом электронными сообщениями, а родственники посыпают друг другу электронные поздравления.

К несчастью, так сложилось, что большинство сетевых технологий было разработано под конкретные нужды. При создании сети каждое предприятие выбирает необходимое оборудование с учетом объема и необходимой скорости передачи данных, а также стоимости этого оборудования. Однако на основе одной сетевой технологии невозможно создать универсальную сеть, поскольку никакая отдельная сетевая технология, взятая сама по себе, не может подходить для всех случаев жизни. Одним требуется создать высокоскоростную сеть, связывающую компьютеры в пределах здания; однако дешевые сетевые технологии, удовлетворяющие конкретным нуждам, нельзя применять в случае огромных по протяженности территорий. Другие готовы мириться с низкой скоростью работы сети, лишь бы только она связывала компьютеры, находящиеся на расстоянии в несколько тысяч километров.

За прошедшие два десятилетия были разработаны новые технологии, с помощью которых стало возможным соединить между собой разнородные физические сети и заставить работать эту систему как единую скординированную сеть. Технология *объединения сетей* (*internetworking*) заставляет работать множество разнородных аппаратных средств благодаря обеспечению способов подключения гетерогенных сетей и набору коммуникационных протоколов, которые делают возможным это взаимодействие. Технология Internet (или технология глобальных сетей) позволяет скрыть от конечного пользователя особенности применения аппаратурного обеспечения и позволяет компьютерам обмениваться информацией независимо от того, к какой физической сети они подключены.

Технология глобальных сетей, описанная в этой книге, является отличным примером *взаимодействия открытых систем* (*open system interconnection*). Они называются *открытыми*, потому что спецификация на них полностью открыта и доступна для использования всеми желающими. Это в корне отличает их от *закрытых* (*proprietary*) коммуникационных систем, стандарты на которые являются правом собственности конкретного производителя. Таким образом, любой разработчик может создать программу, которая будет обмениваться данными

с другими программами через Internet. Однако, самое важное то, что была разработана единая технология для обмена данными между компьютерами, функционирующими на различных аппаратных платформах, использующих практически любое оборудование для сетей с коммутацией пакетов, поддерживающая широкий спектр программного обеспечения и большое количество типов операционных систем.

Чтобы по достоинству оценить технологию глобальных сетей, задумайтесь над тем, насколько кардинально она изменила производственную деятельность человека, особенно сферу бизнеса. Уже никого не удивляет возможность высокоскоростного обмена данными по локальной сети предприятия. Технологии глобальных сетей сделали возможной оперативную обратную связь между производственной сферой, отделом продаж и маркетинга, а также конечными потребителями. В результате существенно выросла скорость планирования, производства, реализации и переоборудования производства. Причем изменения эти поистине разительные!

## 1.2. Глобальная сеть TCP/IP

Потенциальные возможности и важность сетевых технологий в США осознаны много лет тому назад. Государственный департамент выделил средства на исследования, благодаря которым впоследствии была создана Internet. В этой книге описываются принципы и идеи, лежащие в основе технологии глобальных сетей, которые были получены в результате исследований, финансируемых Управлением перспективного планирования научно-исследовательских работ (*Advanced Research Projects Agency*, или ARPA)<sup>1</sup>. В основу технологии ARPA положен набор сетевых стандартов, в которых оговорены детали взаимодействия компьютеров, а также соглашения по объединению сетей и методы маршрутизации трафика. Официально все это было названо семейством протоколов TCP/IP (TCP/IP Internet Protocol Suite), однако чаще всего используется название протокол TCP/IP (по имени двух основных стандартов семейства). Этот протокол может использоваться для обмена информацией между двумя компьютерами, проходящей через любое количество объединенных сетей. Например, в некоторых крупных компаниях протокол TCP/IP применяется во всех связанных между собой корпоративных сетях, даже если эти сети не имеют выхода во внешний мир. Другие же используют этот протокол для обмена информацией между представительствами, разбросанными по всему Земному шару.

Хотя технология применения TCP/IP заслуживает внимания сама по себе, особый интерес к ней проявляется в крупномасштабных проектах, поскольку она прошла проверку временем. Благодаря этой технологии стало возможным создать глобальную сеть Internet, которая сегодня объединяет свыше 170 млн пользователей в разных странах. Причем все эти люди могут находиться как дома, так и в школе, в офисе или научной лаборатории. В США в финансировании Internet принимали участие такие организации, как Национальный научный фонд (*National Science Foundation*, или NSF), министерство энергетики (*Department of Energy*, или DOE), министерство обороны (*Department of Defense*, или DOD), министерство здравоохранения и социальной защиты (*Health and Human Services Agency*, или HHS) и национальное агентство по аeronautике и космическим исследованиям (*National Aeronautics and Space Agency*, или NASA). Во всех этих организациях протокол TCP/IP используется для связи их

<sup>1</sup> ARPA позже была переименована в DARPA (Defense Advanced Research Projects Agency), или Управление перспективного планирования научно-исследовательских работ при Министерстве обороны США.

исследовательских центров. В результате проведенных исследований была создана глобальная коммуникационная система, которую назвали *ARPA/NSF Internet*, *TCP/IP Internet*, *глобальная сеть Internet*, или просто *Internet*<sup>2</sup>. Благодаря ей абоненты, находящиеся в разных уголках Земного шара, обмениваются информацией так же легко, словно они находятся в соседних комнатах. Громадный успех Internet в первую очередь обусловлен высокой устойчивостью семейства протоколов TCP/IP к изменениям и его способностью адаптироваться к основным сетевым технологиям.

Большую часть материала книги можно применить к любой сети на основе протокола TCP/IP, однако в некоторых главах описаны особенности глобальной сети Internet. Поэтому читатели, интересующиеся какой-либо одной технологией, должны быть внимательны и понимать различия (там, где они есть) между существующей структурой глобальной Internet и локальными сетями TCP/IP, построенными по общим правилам. Однако будет глубокой ошибкой пропустить разделы книги, посвященные глобальной сети, поскольку структура внутренних сетей многих крупных корпораций зачастую во много раз сложнее Internet двадцатилетней давности. Поэтому многие проблемы, с которыми вам предстоит столкнуться, наверняка уже были решены раньше при разработке глобальной сети.

### 1.3. Службы Internet

Понять технические подробности работы протокола TCP/IP невозможно без знакомства со службами, реализованными в Internet. В этом разделе мы рассмотрим эти службы очень кратко. Особое внимание будет уделено тем из них, к которым чаще всего обращаются пользователи. В следующих главах будет описано, как компьютеры подключаются к сети TCP/IP и как реализуется нужная для служб функциональность.

При обсуждении служб речь в основном пойдет о стандартах, которые называются *протоколами (protocols)*. Протокол, такой как TCP или IP, представляет собой набор синтаксических и семантических правил, использующихся при обмене данными между двумя компьютерами. В нем оговаривается формат блоков сообщений, описывается реакция компьютера на получение определенного типа сообщения и указываются способы обработки ошибок и других необычных ситуаций. И что самое важное, благодаря протоколам, мы можем описать процесс обмена данными между компьютерами, не привязываясь к какой-то определенной компьютерной платформе или сетевому оборудованию конкретного производителя. В некотором смысле можно сказать, что при передаче данных протокол является тем же самым, чем алгоритм при выполнении вычислений. Как и протокол, алгоритм позволяет описать или понять процесс вычислений без привязки к системе команд конкретного процессора. Точно так же коммуникационный протокол позволяет описать или понять процесс обмена данными, не вдаваясь в детали функционирования сетевого оборудования конкретного производителя.

Скрытие низкоуровневых особенностей процесса передачи данных способствует повышению производительности труда разработчиков. Во-первых, поскольку программистам приходится иметь дело с протоколами, относящимися к достаточно высокому уровню абстракции, им не нужно держать в голове (и даже изучать!) технические подробности используемого аппаратного обеспечения. Поэтому новую программу можно написать достаточно быстро. Во-вторых, поскольку

---

<sup>2</sup> В литературе часто можно встретить названия Internet и internet. Когда речь идет о глобальной сети пишут Internet, а если имеется в виду локальная сеть предприятия, которая построена на основе технологии TCP/IP, пишут internet. Чтобы не путать два близких по написанию слова часто вместо internet пишут intranet.

программы разрабатываются на основе модели, относящейся к высокому уровню абстракции, который не зависит от конкретной архитектуры компьютера или типа сетевого оборудования, в них не нужно вносить никаких изменений при переходе на другой тип оборудования или изменении конфигурации сети. В-третьих, поскольку в прикладных программах используются протоколы высокого уровня, которые опять же не зависят от применяемого оборудования, они (программы) могут обеспечивать непосредственное соединение между двумя любыми компьютерами сети. При этом программистам не нужно создавать специальные версии программ для каждого типа используемого компьютера или сети. Программы разрабатываются с использованием универсальных протоколов. Поэтому один и тот же код можно скомпилировать и запустить на любом компьютере.

Ниже мы увидим, что подробности функционирования служб Internet оговорены в отдельных протоколах. В следующем разделе будут рассмотрены несколько протоколов, относящихся к сетевым службам Internet уровня приложений, а также к службам, формирующими сетевой уровень. В последующих главах каждый из этих протоколов будет рассмотрен во всех подробностях.

### 1.3.1. Службы Internet уровня приложений

С точки зрения конечного пользователя, Internet состоит из набора прикладных программ, которые используют ресурсы существующей сети для выполнения полезной работы по обмену информацией. Для обозначения возможности кооперации разнородных систем при решении вычислительных проблем мы будем использовать специальный термин *степень взаимодействия (interoperability)*. Так вот, выражаясь научным языком, приложения для Internet имеют высокую степень взаимодействия. Большинство пользователей для доступа к Internet просто запускают одну из прикладных программ. При этом они даже не задумываются над тем, к компьютерам какого типа происходит обращение, как работает технология TCP/IP, какова структура сетей и по каким маршрутам проходят данные. Они во всем полагаются на прикладную программу и обслуживающее ее сетевое программное обеспечение, в которых все эти детали учитываются. Подробности работы сети на основе протокола TCP/IP и понимание происходящих при этом процессов необходимы только программистам, разрабатывающим сетевое программное обеспечение.

К наиболее популярным и широко распространенным службам Internet можно отнести те, что перечислены ниже.

- *World Wide Web*. Эта служба позволяет клиентам просматривать документы, содержащие текст и графику, и перемещаться по гиперссылкам от одного документа к другому. Бурный рост Web произошел в начале 1995 года. Как раз на это время приходится резкое увеличение потока данных этой службы по глобальной сети Internet. Служба Web не сдает своих позиций и в настоящее время. Некоторые из поставщиков услуг (провайдеров) Internet отмечают, что на Web приходится примерно 80% их общего трафика.
- *Электронная почта (e-mail)*. Программы электронной почты позволяют создавать текстовые документы и рассыпать их копии одному или нескольким адресатам. Полученные сообщения можно прочитать с помощью тех же почтовых программ. Введение новых технологий позволяет включать в почтовые сообщения файлы любого типа. Развитие системы электронной почты было столь успешным, что теперь большинство пользователей Internet не представляет себе, как можно обходиться без электронных сообщений. Одна из причин огромной популярности службы электронной почты Internet состоит в хорошо продуманном проекте ее реализации:

забота по доставке сообщений возложена на сам протокол. В рамках протокола оговорены не только способы установки прямой связи между почтовыми серверами отправителя и получателя, но и указывается, что сообщение не может быть удалено с компьютера отправителя до тех пор, пока получатель не поместит его копию в надежное для хранения место.

- **Пересылка файлов.** Как следует из названия, эта служба позволяет обмениваться данными путем пересылки файлов. Служба пересылки файлов — одна из старейших в Internet, однако она продолжает интенсивно использоваться до сих пор. И хотя файлы небольших размеров можно отправить в виде вложения по электронной почте, без службы пересылки файлов не обойтись при передаче больших файлов. В этой службе предусмотрены способы авторизации пользователей, а также возможность запретить доступ всем пользователям. Пересылка файлов через сеть TCP/IP так же надежна, как и передача электронной почты, поскольку два компьютера непосредственно устанавливают соединение между собой, без участия третьего компьютера, выполняющего промежуточное хранение данных.
- **Удаленный доступ к системе.** Эта служба позволяет подключиться к удаленной машине и установить с ней интерактивный сеанс связи. При этом на локальном компьютере пользователя появляется окно, в котором отображается информация, выводимая удаленным компьютером в ответ на ввод команд с клавиатуры локального компьютера. Как только сеанс связи с удаленной машиной завершается, управление возвращается операционной системе локального компьютера.

Работа этих и других служб Internet будет детально описана в следующих главах. В них мы более конкретно расскажем, как в перечисленных службах используется семейство протоколов TCP/IP и почему применение протоколов уровня приложений гарантирует полную универсальность программ.

### 1.3.2. Службы Internet сетевого уровня

Программисты, разрабатывающие приложения с использованием протоколов TCP/IP, имеют совершенно другие представления о сети по сравнению с пользователями, которые только запускают прикладные программы типа почтового клиента. На сетевом уровне в семействе протоколов TCP/IP предусмотрено два обширных класса служб, которые используются во всех приложениях. И хотя на данном этапе мы не будем вдаваться в подробности их функционирования, ни одно описание протоколов TCP/IP без них не обходится.

- **Служба доставки пакетов, не требующая установки соединения** (*Connectionless Packet Delivery Service*). Эта служба лежит в основе всех остальных сетевых служб (ее подробное описание будет приведено в следующих главах). Она относится к абстрактным типам служб и реализована практически во всех сетях с коммутацией пакетов. В основе работы сети TCP/IP лежит принцип пересылки небольших сообщений (пакетов) от одного компьютера к другому, выполняющийся с учетом маршрутной информации, которая находится в самом сообщении. Поскольку служба, не требующая установки соединения, пересылает каждый пакет независимо друг от друга, она не может гарантировать их надежную доставку, а также доставку в заданном порядке. Однако, так как эта служба обычно тесно привязана к сетевому аппаратному обеспечению, эффективность ее работы чрезвычайно высока. И что самое важное, поскольку служба доставки пакетов, не требующая установки соединения, лежит в основе всех сетевых

служб, это позволяет адаптировать семейство протоколов TCP/IP практически к любому типу сетевого оборудования.

- *Надежная потоковая транспортная служба* (*Reliable Stream Transport Service*). Для работы большинства приложений недостаточно только одной службы, не гарантирующей доставку пакетов. Причина в том, что прикладные программы часто требуют от сетевого программного обеспечения автоматического восстановления данных при возникновении ошибок при передаче, потери пакетов или отказов одного из звеньев сети между отправителем и получателем. Надежная потоковая транспортная служба как раз занимается решением этих проблем. Она позволяет приложению, запущенному на одном компьютере, установить “соединение” с приложением, запущенным на другом компьютере, а затем переслать большой объем данных, как если бы эти два компьютера имели постоянный прямой канал связи между собой. Естественно, что при этом сетевые протоколы нижнего уровня разбивают поток данных на небольшие сообщения и передают их последовательно друг за другом, ожидая от получателя подтверждения успешного приема.

Службы, аналогичные описанным выше, реализованы практически во всех типах сетей, поэтому интересно рассмотреть, чем же отличаются службы сети TCP/IP. Ниже перечислены только основные отличия.

- *Независимость от сетевой технологии*. Несмотря на то что в семействе протоколов TCP/IP используется традиционная технология коммутации пакетов, она не зависит от применяемого сетевого оборудования. В глобальной сети Internet используется множество сетевых технологий, причем не все из них изначально были ориентированы для работы на больших расстояниях. В семействе протоколов TCP/IP определяется единица передачи информации, которая называется *дейтаграммой* (*datagram*), и оговариваются методы передачи дейтаграмм по отдельным сетям.
- *Всеобщность подключения*. Сеть TCP/IP позволяет обмениваться информацией любым двум подключенными к ней компьютерам. Каждому компьютеру назначается *адрес*, который интерпретируется по одним и тем же правилам во всей сети. В любой дейтаграмме присутствует как адрес отправителя, так и адрес получателя пакета. Адрес получателя используется промежуточными звенями сети для правильной маршрутизации этой дейтаграммы.
- *Подтверждение получения пакетов*. В сетях на основе протокола TCP/IP оговаривается процедура подтверждения получения пакетов. Конечный получатель посылает уведомление отправителю даже в том случае, если они не подключены к одной физической сети. При этом промежуточные звенья сети, через которые проходят пакеты, уведомления о получении отправителю не шлют.
- *Стандарты протоколов прикладных программ*. Кроме основных служб транспортного уровня, рассмотренных выше, в семейство протоколов TCP/IP включены стандарты для широко распространенных прикладных служб, таких как электронная почта, пересылка файлов и удаленный вход в систему. Поэтому, разрабатывая прикладные программы с использованием протоколов TCP/IP, программисты часто обнаруживают, что нужные им сетевые компоненты уже созданы.

В следующих главах будут описаны службы, предназначенные для использования сетевыми программистами, а также стандарты протоколов уровня приложений.

## 1.4. История развития Internet

Несомненно, что основная причина столь высокой популярности и привлекательности технологии TCP/IP связана с ее универсальностью, а также с быстрым ростом глобальной сети Internet. В середине 1970-х годов по инициативе ARPA были начаты работы над новой перспективной технологией, которые к концу 1970-х принесли свои плоды — были сформированы структура и набор основных протоколов. В тот период организация ARPA выступала в роли основного источника финансирования в области исследования сетей с коммутацией пакетов. Оно внедрило множество новейших идей в известном проекте сети ARPANET. Хотя сеть ARPANET была построена на традиционных двухточечных проводных выделенных линиях, ARPA финансировала исследования технологии передачи пакетов данных по радиоканалу и каналам спутниковой связи. Наличие множества сетевых технологий передачи данных заставило ARPA заняться проблемой межсетевого взаимодействия, что в свою очередь подхлестнуло развитие глобальной сети.

Доступность результатов исследований, финансируемых ARPA, привлекла внимание нескольких групп разработчиков и вызвала их профессиональный интерес. Особенно это касается тех инженеров, кто уже имел определенный опыт в технологии коммутации пакетов, использовавшейся в проекте ARPANET. Поэтому ARPA стала периодически проводить неформальные конференции разработчиков, на которых они могли обменяться идеями и обсудить результаты экспериментов. Это “сборище” стали неформально называть *группой исследователей Internet (Internet Research Group)*. Однако уже к 1979 году в исследование технологии TCP/IP было вовлечено такое количество людей, что ARPA было вынуждено создать неформальный комитет для координации их действий и руководства работами в области создания протоколов и структуры зарождающейся Internet. Эту организацию решили назвать *Советом по управлению и структуре Internet (Internet Control and Configuration Board, или ICCB)*. Заседания Совета регулярно проходили вплоть до 1983 года, после чего он был реорганизован.

История глобальной сети Internet начинается примерно с 1980 года, когда ARPA стало переводить компьютеры, подключенные к своим исследовательским сетям, на использование нового семейства протоколов TCP/IP. Модернизацию, естественно, начали с ARPANET, поэтому она быстро стала основой зарождающейся Internet, ее *магистральным каналом (backbone)*, который в дальнейшем стали использовать для экспериментов с создававшимся семейством протоколов TCP/IP. Повсеместный переход на новую технологию Internet был завершен в январе 1983 года, когда штаб-квартира Министерства обороны США объявила, что во всех компьютерах министерства, которые подключены к сетям, протянувшимся на большие расстояния, используется протокол TCP/IP. Тогда же *управление связи* Министерства обороны США (*Defense Communication Agency, или DCA*) разделило сеть ARPANET на две независимые сети. Одна была отведена для нужд науки, а вторая — для военных целей. За исследовательской сетью было сохранено название ARPANET, а военную сеть (которая была несколько больше) стали называть *MILNET (military network)*.

Чтобы заинтересовать университетскую общественность в использовании новых протоколов, ARPA стало продавать их программную реализацию по низкой цене. В то же время на большинстве компьютерных кафедр университетов использовалась версия операционной системы UNIX, разработанная в Калифорнийском университете. Она называлась *Berkeley Software Distribution*, но чаще всего ее называли *Berkeley UNIX*, или просто *BSD UNIX*. На средства ARPA корпорация *Bolt Beranek and Newman, Incorporated (BBN)* выполнила программную реализацию семейства протоколов TCP/IP для системы UNIX, а разработчики из Беркли включили их в дистрибутив своей операционной системы.

В результате ARPA удалось внедрить свое детище более чем в 90% университетских компьютерных лабораторий и факультетов. Новая программная реализация протоколов TCP/IP появилась в очень подходящее время, поскольку на большинстве факультетов имелось, как правило, несколько компьютеров и велись работы по их объединению в локальную сеть. Вот где пригодились новые коммуникационные протоколы!

Версия BSD UNIX оказалась чрезвычайно популярной еще и потому, что помимо основных протоколов TCP/IP в ней было реализовано много полезных вещей. Разработчики из Беркли предложили набор утилит для реализации сетевых служб, которые напоминали службы UNIX, использовавшиеся на локальной машине. Основным отличием этих утилит было то, что они удовлетворяли принятым стандартам UNIX. Поэтому, опытному пользователю UNIX не составляло большого труда понять, как работает, например, утилита удаленного копирования файлов `rcp`, разработанная в Беркли, поскольку она функционировала аналогично команде копирования файлов системы UNIX.

Кроме набора сетевых утилит, в версии Berkeley UNIX было впервые введено новое абстрактное понятие *сокета* (*socket*), которое позволяло прикладным программам обращаться к коммуникационным протоколам. По сути сокет являлся обобщением стандартного механизма ввода-вывода системы UNIX и, кроме протокола TCP/IP, позволял работать еще с несколькими сетевыми протоколами. Структура системы сокетов вызвала полемику у специалистов после ее принятия, поэтому многие разработчики операционных систем придумали им замену. Но, независимо от достоинств и недостатков, введение в операционную систему абстрактного уровня сокетов было значительным достижением, поскольку они позволяли программистам легко пользоваться семейством протоколов TCP/IP. Все это подогрело интерес разработчиков к выполнению экспериментов с семейством протоколов TCP/IP.

Успешное использование технологии TCP/IP и Internet ведущими исследовательскими центрами привело к ее быстрому распространению. Понимая важность перспективы исследований в области сетевых коммуникаций, *Национальный научный фонд США* (*National Science Foundation*, или *NSF*) поставил задачу охватить технологией TCP/IP и Internet как можно больше научных учреждений. С этой целью в конце 1970-х годов NSF финансировал проект сети *CSNET* (*Computer Science NETwork*), которая и связала все научные компьютерные лаборатории. В 1985 году NSF начал программу по обеспечению доступа к сети, объединившую его шесть суперкомпьютерных центров. В 1986 году NSF расширяет исследования в области сетевых технологий и выделяет средства на создание проекта новой глобальной магистральной сети, которую назвали *NSFNET*<sup>3</sup>. В конечном итоге эта сеть связала все суперкомпьютерные центры США и объединила их с ARPANET. В том же 1986 году NSF делает начальные инвестиции во множество проектов региональных сетей, каждая из которых связала основные научно-исследовательские учреждения в своем регионе. Во всех региональных сетях, финансируемых NSF, использовалось семейство протоколов TCP/IP, и все они стали частью глобальной сети Internet.

В течение семи лет с момента появления к Internet были подключены сотни частных сетей в США и Западной Европе. Общее количество компьютеров, объединенных в глобальную сеть, составило порядка 20 тыс. В основном их пользователи работали в учебных, государственных и корпоративных научных учреждениях. При этом масштабы роста и использования Internet значительно превысили

<sup>3</sup> Термин NSFNET иногда используется в более широком смысле и означает всю программу исследований в области сети, финансируемую фондом NSF. Однако в этой книге мы будем употреблять его в узком смысле, понимая при этом глобальный магистральный канал. В следующих главах эта технология будет описана более подробно.

все прогнозы. Исследования, проведенные в конце 1987 года, показали, что темпы роста Internet составляют 15% в месяц. К началу 2000 года глобальная сеть Internet охватывала более 50 млн компьютеров в 209 странах мира.

Выбор семейства протоколов TCP/IP на заре развития Internet и последовавший за этим грандиозный его рост не были обусловлены только проектами, финансируемыми государством. К Internet подключались как основные компьютерные гиганты, так и много других больших корпораций — нефтяные компании, автомобильные заводы, электронные фирмы, фармацевтические и телекоммуникационные компании. Средние и мелкие предприятия стали подключаться к Internet начиная с 1990 года. Кроме того, во внутренних сетях многих организаций использовалось семейство протоколов TCP/IP, даже если они не были подключены к Internet.

Темпы расширения Internet вызвали проблемы, непредвиденные на этапе проектирования. Это заставило разработчиков выработать технологию управления больших распределенных систем. Например, на этапе проектирования имена и адреса всех компьютеров, подключенных к Internet, хранились в одном файле, который редактировался вручную и после этого рассыпался всем абонентам, подключенным к сети. Однако уже к середине 1980-х годов стало очевидно, что подобный подход централизованного хранения информации неэффективен. Во-первых, поскольку к Internet подключалось все больше и больше компьютеров, запросы на обновление этого файла стали превышать физические возможности человека, который их обрабатывал. Во-вторых, даже если предположить, что с файлом центральной базы данных не возникло бы никаких проблем, существующих в то время сетевых мощностей было явно недостаточно для частой рассылки информации об обновлении на каждый компьютер или предоставлению оперативного доступа к этой информации каждому абоненту сети.

Поэтому были разработаны новые протоколы, и по всей глобальной сети Internet запущена специальная служба системы имен, которая позволяла любому пользователю автоматически преобразовывать имя удаленного компьютера в его адрес. Эту службу называли *системой доменных имен* (*Domain Name System*, или *DNS*). Ее основу составляли специальные компьютеры, называвшиеся *серверами имен* (*name servers*), в задачи которых входила обработка запросов клиентов на преобразование имен. При этом ни на одном из серверов не хранилась база данных доменных имен целиком — данные были распределены между группами серверов, которые использовали семейство протоколов TCP/IP для связи между собой при выполнении запроса на преобразование имени.

## 1.5. Архитектурный совет Internet

Поскольку семейство протоколов TCP/IP не было разработано конкретным производителем или признанным сообществом профессионалов, естественно, возникает вопрос: “Кто осуществляет техническое руководство и решает когда протокол должен стать стандартом?” Ответ на этот вопрос таков: существует специальная группа лиц, которая называется *архитектурным советом Internet* (*Internet Architecture Board*, или *IAB*<sup>4</sup>). IAB направляет усилия разработчиков на решение конкретных проблем, координирует исследования и разработку семейства протоколов TCP/IP, а также руководит процессом роста Internet. IAB решает, какие протоколы нужно включить в семейство TCP/IP, и проводит официальную политику.

IAB был основан в 1983 году после того, как ARPA реорганизовала Совет по управлению и структуре Internet (ICCB). Он стал правопреемником всех ранних

---

<sup>4</sup> IAB первоначально называлась как Internet Activities Board (Координационный совет Internet).

групп, разрабатывавших стандарты. Основными целями IAB были: всестороння поддержка обмена мнениями и идеями между разработчиками, вовлеченными в процесс исследования технологии TCP/IP и Internet; сосредоточение усилий разработчиков на решении основных проблем. За первые шесть лет IAB прошел путь от исследовательской группы при ARPA до самостоятельной организации. За это время каждый член IAB возглавил *специальную группу по решению конкретной задачи Internet* (*Internet Task Force*, или *ITF*). Каждая такая группа занималась конкретной проблемой или рядом важных проблем. IAB выделил около десяти направлений. Круг его интересов был широк: от исследования такой глобальной проблемы, как влияние трафика различных приложений на функционирование Internet, до решения текущих технических задач. Архитектурный совет Internet проводил свои заседания несколько раз в год. На них заслушивались отчеты каждой из групп, анализировались и пересматривались технические инструкции, обсуждалась текущая политика и проводился обмен информацией с представителями таких организаций как ARPA и NSF, finanziровавшими работы по Internet и исследования в области сетевых технологий.

Председателю архитектурного совета присваивалось звание *Главного архитектора Internet* (*Internet Architect*). Он отвечал за принятие технических решений, а также координировал работу различных специальных групп IETF. Кроме того, на заседаниях совета IAB председатель ставил новые задачи специальному группам IETF, а также представлял IAB в других организациях.

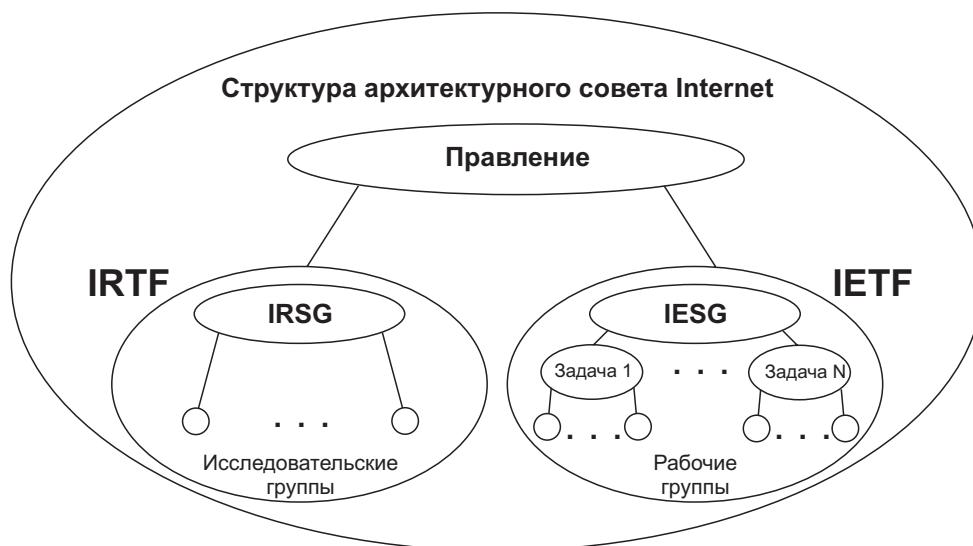
Новички часто удивляются, когда узнают о том, что IAB не располагает большим бюджетом. Принимая основные решения, IAB не финансирует большинство исследований и инженерных разработок. Большую часть работы выполняют добровольцы. Каждый из членов IAB отвечает за привлечение добровольцев для решения поставленной задачи, созыв и проведение собрания специальной группы, а также за составление отчета для IAB. Как правило, добровольцы работают в научно-исследовательских или коммерческих организациях, занимающихся разработкой или использованием технологии TCP/IP. Активные исследователи принимают участие в заседаниях специальных групп IETF по двум причинам. С одной стороны, это позволяет им ознакомиться с новыми исследовательскими задачами и планами. С другой стороны, поскольку участники специальных групп занимаются разработкой новых идей, решением возникающих проблем и проведением экспериментов, которые впоследствии становятся частью технологии TCP/IP, применяемой в Internet, они осознают, что результаты их работы оказывают непосредственное и положительное влияние на развитие системы в целом.

## 1.6. Реорганизация архитектурного совета Internet

К лету 1989 года масштабы использования технологии TCP/IP и Internet вышли далеко за рамки, которые были предусмотрены первоначальным исследовательским проектом. Они активно стали применяться в производственной сфере, в которую ежедневно были вовлечены десятки тысяч людей, причем деятельность этих людей непосредственно зависела от новой технологии. Поэтому стало невозможным отрабатывать новые идеи на реально работающем оборудовании, изменяя по ночам его конфигурацию. В большей степени от этого пострадали бы сотни коммерческих компаний, которые выпускали продукты, использовавшие технологию TCP/IP, поскольку разработчики определяли степень операционной совместимости своих продуктов путем внесения изменений в их программный код. Исследователи, создавая черновые варианты спецификаций стандартов и отрабатывая новые идеи, не могли больше рассчитывать на быстрое признание и внедрение результатов их работы. По иронии судьбы те, кто разрабатывал и внедрял

технологию TCP/IP, оказались под влиянием коммерческого успеха созданного ими детища. Короче говоря, технология TCP/IP имела колossalный коммерческий успех, и интересы рынка стали превалировать над интересами науки.

Реакцией на изменение политических и коммерческих реалий, связанных с семейством протоколов TCP/IP и самой Internet, стала реорганизация IAB летом 1989 года. Были изменены обязанности председателя совета. Исследователи были выведены из состава IAB, и из них была образована отдельная дочерняя группа. Состав нового совета IAB расширился, в него были введены представители от многочисленной сетевой общественности. На рис. 1.1 показана организационная структура нового совета IAB и взаимосвязи между его дочерними группами.



*Рис. 1.1. Структура архитектурного совета Internet (IAB) после реорганизации, проведенной в 1989 году*

Как видно из рис. 1.1, кроме правления, в структуру IAB входит еще две большие специальные группы: исследовательская (*Internet Research Task Force*, или *IRTF*) и инженерная (*Internet Engineering Task Force*, или *IETF*).

Как следует из названия, инженерная группа IETF занимается решением краткосрочных или среднесрочных технических проблем. Эта группа существовала в первоначальной структуре IAB, и достигнутые ею успехи в определенной степени послужили причиной реорганизации Совета в целом. В отличие от большинства специальных групп IAB, состоящих из нескольких человек, которые решали одну конкретную задачу, до реорганизации инженерная группа IETF была очень многочисленна, в нее входили десятки активных сотрудников, работающих одновременно над несколькими проблемами. Поэтому ее разделили на 20 рабочих групп (*working groups*), каждая из которых должна заниматься решением конкретной задачи. Для поиска методов решения задачи рабочие группы периодически проводят совещания. Кроме того, регулярно проводится общее собрание инженерной группы IETF, на котором заслушиваются отчеты рабочих групп и обсуждаются предложения или дополнения, которые нужно внести в стандарты протоколов семейства TCP/IP. Обычно проводится три собрания в год, в которых участвуют сотни людей — как участников инженерной группы IETF, так и приглашенных. Группа IETF стала настолько большой, что ею уже трудно управлять одному человеку.

Поскольку репутация группы IETF в Internet была чрезвычайно высока, а также потому, что собрания группы привлекали огромное внимание широких масс сетевой общественности, при реорганизации IAB было решено сохранить IETF в целом, но поделить ее примерно на десять подразделений, возглавляемых отдельным руководителем. Председатель инженерной группы IETF и руководители подразделений образовали так называемую *руководящую инженерную группу Internet (Internet Engineering Steering Group, или IESG)*, которая должна координировать действия рабочих групп IETF. В результате аббревиатура IETF стала относиться ко всей структуре инженерного совета в целом, включая председателя, руководителей подразделений и всех членов рабочих групп.

Во время реорганизации IAB была образована специальная *исследовательская группа*, которую назвали *Internet Research Task Force (IRTF)*. В ее задачи входит координация исследований, связанных с семейством протоколов TCP/IP и со структурой Internet в целом. Как и IETF, в IRTF создана небольшая *руководящая исследовательская группа (Internet Research Steering Group, или IRSG)*, в задачи которой входит расстановка приоритетов и координация направлений проводимых исследований. В отличие от IETF, группа IRTF малочисленней и ее члены проявляют гораздо меньшую активность. Фактически основная часть исследований проводится инженерной группой IETF.

## 1.7. Сообщество Internet

В 1992 году Internet, фактически, вышел из под контроля своего опекуна — правительства США. Поэтому, чтобы поддержать дух партнерства было создано специальное *сообщество Internet (Internet Society, или ISOC)*. Эта международная организация создана по инициативе *Национального географического общества* США. Сообщество Internet выступает в роли головной организации для IAB и служит делу объединения людей во всем мире посредством использования Internet.

## 1.8. Запросы на комментарии в Internet

Выше мы уже говорили о том, что у технологии TCP/IP нет конкретного разработчика, за ней не стоит какое-либо признанное профессиональное сообщество, отдельный человек или группа лиц. В результате конечные пользователи не могут получить описание протоколов, стандартов и правила приемлемого использования сети непосредственно от производителя. Поэтому вся документация помещена в специальное хранилище, доступ к которому в оперативном режиме (online) открыт для всех пользователей Internet совершенно бесплатно.

Вся документация, относящаяся к работе Internet, планы введения новых и пересмотра старых протоколов, а также полное описание семейства протоколов TCP/IP хранится в виде набора технических статей, которые называются *запросами на комментарии в Internet (Requests For Comments, или RFC)*. Статьи RFC могут иметь объем от нескольких до сотен страниц в зависимости от того, что в них описано, — общие концепции или детали работы конкретного протокола, существующий стандарт или только предложение по его введению<sup>5</sup>. И хотя документы RFC не относятся к научным академическим трудам, информация в них тщательно выверяется и редактируется. Этой работой много лет занимался

<sup>5</sup> Подробнее узнать о том, что такие документы RFC, какими пользоваться, какое множество их существует, какова их тематика и какие ходят по поводу их шутки и розыгрыши, вы узнаете, прочитав приложение 1, “Справочник по документам RFC”.

покойный ныне Джон Постел<sup>6</sup>. Сейчас задача редактирования документов RFC возложена на руководителей подразделений инженерной группы IETF. Утверждением новых документов RFC занимается руководящая инженерная группа Internet IESG в полном составе.

Кроме RFC, на заре развития Internet издавалась параллельная серия документов под названием *технические заметки об Internet* (*Internet Engineering Notes*, или *IEN*). И хотя IEN больше не выходят, тем не менее не вся содержащаяся в них информация была перенесена в документы RFC. Поэтому в тексте книги, помимо ссылок на RFC, вы столкнетесь со ссылками на документы IEN.

Документы RFC последовательно нумеруются в хронологическом порядке, соответствующем времени их написания. Каждому вновь созданному или просмотренному документу назначается новый номер. Поэтому при поиске нужной информации будьте внимательны и используйте версию документа с наивысшим номером. В этом вам поможет полный список документов RFC, в котором перечислены их номера и названия.

Чтобы сделать документы RFC доступными широкой общественности, их копии размещены на многих серверах в Internet. Копии документов можно получить также по обычной почте, электронной почте или загрузить их из Internet с помощью FTP. Кроме того, существуют еще предварительные версии документов RFC, которые называются *рабочими документами Internet* (*Internet drafts*). Чтобы получить доступ к документам RFC или черновикам Internet в вашем регионе, обратитесь к местному специалисту по сетям. За инструкциями можно также обратиться к приложению 1, “Справочник по документам RFC”.

## 1.9. Протоколы Internet и стандартизация

Искушенный в вопросах сетевой технологии читатель наверняка знает, что существует огромное количество стандартов сетевых протоколов. Причем многие из них были созданы еще до появления Internet. Поэтому возникает закономерный вопрос: “Зачем создатели Internet стали придумывать новые протоколы, если на тот момент существовало большое количество международных стандартов протоколов?” Однозначно ответить на этот вопрос сложно, но в приведенном ниже абзаце, определенно, есть здравый смысл.

*Существующие стандарты протоколов следует использовать там, где это возможно. Новые протоколы создаются только в том случае, когда нельзя обойтись существующими стандартами. Однако, как только новые стандарты, обеспечивающие нужную функциональность будут приняты, их нужно сразу же использовать.*

Итак, при создании семейства протоколов TCP/IP не преследовалась цель проигнорировать или отменить существующие стандарты, хотя внешне это может выглядеть именно так. Разработки велись, поскольку ни один из существующих стандартов не позволял взаимодействовать между собой сетевым коммуникационным системам.

## 1.10. Перспективы роста и технология

Развитие технологии TCP/IP и Internet продолжается. Время от времени предлагаются к принятию новые протоколы и пересматриваются старые стандарты. В свое время фонд NSF создал сложную коммуникационную систему,

<sup>6</sup> Джон умер осенью 1998 года. Он был одним из тех, кто внес значительный вклад в развитие Internet и технологии TCP/IP. Для тех, кто знал его лично, — это большая потеря.

предложив архитектуру с магистральным каналом, к которому подключились региональные сети и сотни сетей учебных заведений. Сегодня к Internet продолжают подключаться тысячи различных организаций, разбросанных по всему миру. Однако наиболее существенные изменения в Internet происходят не от подключения дополнительных сетей, а от роста трафика. По мере подключения новых пользователей в Internet увеличивается количество запущенных приложений, что в свою очередь приводит к изменению общей картины потока данных. Например, после того как к Internet подключились физики, химики и биологи, они стали обмениваться друг с другом данными экспериментов. По сравнению с сообщениями электронной почты файлы с ними имели просто гигантские размеры. По мере роста популярности Internet пользователи стали использовать для получения информации новые службы, наподобие *World Wide Web*, что привело к еще большему увеличению трафика.

Чтобы справиться с возрастающим трафиком, пропускную способность магистрального канала NSFNET увеличивали уже три раза. Его последний вариант называют *ANSNET* (по имени модернизировавшей его компании). Пропускная способность этого канала в 840 раз больше первоначальной! Начиная с 1995 года многие коммерческие компании, называемые *поставщиками услуг Internet* (*Internet Service Provider*, или *ISP*), создают собственные магистральные каналы, причем пропускная способность многих из них существенно выше, чем у магистрального канала, финансируемого государством (даже после последней модернизации). В настоящее время трудно предсказать, когда прекратится наращивание мощности этих каналов.

Возросшая потребность в межсетевом обмене не является неожиданностью. Компьютерная индустрия постоянно нуждается в увеличении мощности вычислительных устройств и расширении хранилищ данных, способных вместить информацию, накопленную за много лет. Пользователи только сейчас начинают понимать, как следует пользоваться сетевой технологией. Поэтому нетрудно предсказать, что в будущем человечество будет испытывать постоянную потребность в увеличении мощности средств связи. Например, уже сейчас технология TCP/IP используется для передачи голоса и видеоизображения (а не только традиционных данных). Таким образом, современные высокоеемкие коммуникационные технологии требуют расширения мощности каналов связи.

	Число сетей	Число компьютеров	Количество пользователей	Количество менеджеров
1980	10	$10^2$	$10^2$	$10^0$
1990	$10^3$	$10^5$	$10^6$	$10^1$
2000	$10^5$	$10^7$	$10^8$	$10^2$

Рис. 1.2. Обобщенные данные роста Internet за два десятилетия. По мере расширения сети, помимо роста трафика, существенно увеличилась ее сложность, что явилось следствием децентрализованного управления как на этапах разработки, так и внедрения

На рис. 1.2 приведены обобщенные данные расширения Internet и выделены важные компоненты роста. Как следует из таблицы, изменения коснулись как количества пользователей, так и тех, кто управляет сетью. Поскольку технология

разрабатывалась в то время, когда один человек из ARPA мог держать под контролем все аспекты Internet, структура многих подсистем зависела от централизованного управления и контроля. По мере роста и расширения Internet ответственность за функционирование и управление сетью были разделены между несколькими организациями. В частности, как только Internet стала глобальной сетью, в структуру ее управления и поддержания функционирования было вовлечено множество стран. С начала 1990-х годов было потрачено много сил на поиск путей децентрализации управления.

## 1.11. Структура книги

Описание технологии TCP/IP потребовало трех томов. В первом томе представлены общие сведения по семейству протоколов TCP/IP, описаны приложения, использующие эти протоколы, а также структура глобальной сети Internet. В нем рассматриваются протоколы TCP и IP и их взаимодействие в локальной сети предприятия. Кроме деталей, в тексте первого тома приведены основополагающие принципы работы сетевых протоколов, а также рассказано, почему семейство протоколов TCP/IP без проблем может использоваться в различных физических сетях. Во втором томе подробно описаны внутренние особенности семейства протоколов TCP/IP и способы их реализации. Там же приведен программный код, взятый автором из реально функционирующих систем, на примере которого продемонстрировано взаимодействие отдельных протоколов. Материал второго тома будет в первую очередь интересен специалистам, создающим внутреннюю сеть предприятия. В третьем томе показано использование протокола TCP/IP для обмена данными в распределенных приложениях. Основное внимание уделено модели взаимодействия клиент/сервер, которая лежит в основе всех методов распределенного программирования. Описан интерфейс между прикладными программами и протоколами<sup>7</sup> и структура программ, использующих технологию клиент/сервер. Кроме того, в третьем томе обсуждается концепция дистанционного вызова процедур (RPC) и программного обеспечения среднего уровня (middleware), а также рассматриваются средства, используемые программистами для создания клиентской и серверной части программной системы.

До сих пор мы говорили о технологии TCP/IP и Internet в общих словах, описали стандартный набор служб и историю их развития. В следующей главе приведен краткий обзор различных типов сетевого оборудования, которое используется в Internet. При этом не ставилась цель осветить нюансы оборудования конкретного производителя. Основное внимание уделяется возможности технологий передачи данных, которые чаще других используются при создании локальных и глобальных сетей. В последующих главах будут очень подробно описаны протоколы и Internet, причем акцент сделан на трех важных вещах: основных идеях и структурной модели Internet, семействе протоколов TCP/IP и стандартах служб высокого уровня, таких как электронная почта и пересылка файлов. В главах с 3 по 14 излагаются основополагающие принципы и описывается сетевое программное обеспечение, которое имеется на любом компьютере, использующем протокол TCP/IP. В следующих главах будут описаны службы, для работы которых требуется взаимодействие нескольких компьютеров. Речь идет об обмене маршрутной информацией, преобразовании имен и приложениях наподобие электронной почты.

---

<sup>7</sup> Существует три варианта третьего тома. В первом из них в примерах используется интерфейс сокетов системы UNIX. Во втором варианте используется интерфейс транспортного уровня (*Transport Layer Interface*, или *TLI*), а в третьем варианте примеры построены на интерфейсе сокетов системы Windows (*Windows Sockets Interface*), разработанный фирмой Microsoft.

Книгу завершают два приложения. В первом из них даны инструкции по работе с документами RFC. Это приложение дополняет информацию, приведенную в настоящей главе. Там же вы найдете примеры того, что содержится в документах RFC. Вы узнаете, как можно получить копии документов RFC по электронной и обычной почте, а также через FTP. Поскольку стандартный индекс RFC составлен в хронологическом порядке, в приложении приведен список документов RFC, отсортированный в тематическом порядке. Это поможет начинающим пользователям найти нужный документ по соответствующей теме.

Во втором приложении приведен словарь терминов и аббревиатур, часто встречающихся в литературе и повсеместно использующихся в этой книге. Поскольку новая терминология часто приводит в замешательство новичков и обычно трудна для восприятия, мы упорядочили словарь в алфавитном порядке, чтобы не заставлять читателя перелистывать страницы книги в поисках объяснения нужного понятия.

## 1.12. Резюме

Глобальная сеть состоит из множества связанных между собой сетей, которые функционируют как единое целое. Ключевым преимуществом глобальной сети Internet является то, что она обеспечивает универсальный способ взаимодействия систем, в каждой из которых используется различное аппаратное обеспечение, наилучшим образом удовлетворяющие потребности разработчиков. Ниже мы опишем принципы, лежащие в основе межсетевого обмена данными, и подробности функционирования одного из семейств протоколов, использующихся для этих целей. Предметом нашего рассмотрения является технология TCP/IP (ее название происходит от названия двух основных протоколов), которая была разработана Управлением перспективного планирования научно-исследовательских работ (ARPA). Она лежит в основе сети Internet — огромной глобальной сети, связавшей воедино учебные, научные, коммерческие и государственные организации во многих странах мира. В настоящее время происходит бурный рост Internet.

## Материал для дальнейшего изучения

Чтобы почитать увлекательные истории о развитии Internet и познакомиться с ранними исследовательскими работами, посвященными протоколу TCP/IP и межсетевому взаимодействию, обратитесь к книге Церфа (Cerf) [18] и [RFC 1160] *History of the Internet Activities Board (История Архитектурного совета Internet)*. В статье Деннинга (Denning) [40] автор высказывает собственное мнение по вопросам истории ARPANET. Вопросам важности компьютерных сетей для научных исследований посвящена статья Дженнингса (Jennings) [72] и др. Важность межсетевых взаимодействий и один из возможных сценариев развития глобальной сети Internet рассматриваются также в работе Деннинга [40]. *Федеральный координационный комитет по науке, технике и технологиям США (Federal Coordinating Committee for Science, Engineering and Technology, или FCCSET)* предложил считать область исследований межсетевых взаимодействий приоритетной в масштабах страны.

С протоколами очередных заседаний инженерной группы IETF можно ознакомиться на ее Web-сервере [www.ietf.org](http://www.ietf.org). Информационные бюллетени, в которых обсуждаются масштабы развития Internet в разных странах мира, издает сообщество Internet ([www.isoc.org](http://www.isoc.org)). Протоколы и стандартны, посвященные технологии Web издаются консорциумом World Wide Web ([www.w3c.org](http://www.w3c.org)). И наконец, не стоит забывать, что семейство протоколов TCP/IP и сам Internet находятся

в непрерывном развитии. Поэтому за новой информацией периодически обращайтесь к документам RFC и материалам конференций, таких как ежегодный симпозиум ACM SIGCOMM и конференция NETWORLD+INTEROP. Их проведение является значительным событием и привлекает к себе внимание ученых всего мира.

## Упражнения

- 1.1.** Попробуйте экспериментировать с программами, использующими протокол TCP/IP, которые имеются на вашем компьютере и в локальной сети.
- 1.2.** Изобразите графически рост влияния технологии TCP/IP и Internet на вашу организацию. Уточните, сколько компьютеров, пользователей и сетей подключались к Internet каждый год.
- 1.3.** Ежегодная валовая прибыль, полученная от продаж продуктов, в которых используется технология TCP/IP, оценивается в несколько миллиардов долларов. Просмотрите несколько отраслевых изданий и составьте список производителей подобной продукции.



# 2

## Обзор основных сетевых технологий

### 2.1. Введение

Важно понимать, что Internet не является еще одной разновидностью физической сети. По сути, Internet представляет собой метод объединения физических сетей и набор соглашений, использующихся в этих сетях, с помощью которых любые два компьютера могут свободно взаимодействовать друг с другом. Следует учитывать, что сетевое аппаратное обеспечение играет второстепенную роль при разработке структуры сети. Поэтому при изучении технологии глобальных сетей следует различать низкоуровневые средства передачи данных, обеспечиваемые самим оборудованием и высокоуровневые возможности, реализованные в протоколе TCP/IP. Кроме того, также важно понимать влияние интерфейсов, предоставляемых используемой технологией коммутации пакетов на выбор абстракций высокого уровня.

В этой главе описана концепция сети с коммутацией пакетов и используемая при этом терминология. Затем будет сделан краткий обзор основных сетевых технологий и оборудования, которые применяются при построении сетей TCP/IP. В последующих главах будут описаны методы объединения этих сетей и рассказано, почему семейство протоколов TCP/IP может легко работать на различном сетевом оборудовании. И хотя приведенный ниже список сетевых технологий нельзя назвать полным, тем не менее он позволяет показать весь спектр сетевого аппаратного обеспечения, на котором может работать протокол TCP/IP. Читатель может пропустить технические детали изложения, однако при этом он должен уловить основную идею технологии, основанной на коммутации пакетов, и понять методику реализации однородной коммуникационной системы с использованием разнородного аппаратного обеспечения. И что самое важное, читатель должен глубоко понимать систему физической адресации, применяемой в различных сетевых технологиях, поскольку в последующих главах речь пойдет об использовании физических адресов в протоколах высокого уровня.

### 2.2. Два подхода к проблеме сетевого взаимодействия

Независимо от области применения (например, для обмена данными между двумя компьютерами или только для связи компьютера и терминала), сетевые коммуникации можно разделить на два основных типа: *требующие (connection-oriented)* и *не требующие (connectionless)* установки соединения. Сети первого типа называют сетями с коммутацией каналов (*circuit-switched*), а второго —

сетями с коммутацией пакетов (*packet-switched*)<sup>1</sup>. Для работы сети с коммутацией каналов требуется установка *непосредственного соединения* или *канала связи* между двумя абонентами. Например, технология с коммутацией каналов используется в телефонной сети США. При этом в момент набора номера происходит установка прямого канала связи между вызывающим абонентом и местной телефонной станцией. Далее канал связи прокладывается по кабельному хозяйству до телефонной станции вызываемого абонента, а затем — от телефонной станции до самого вызываемого абонента. После того как соединение между абонентами установлено, телефонное оборудование периодически считывает сигналы микрофона, кодирует выборки в цифровом виде и передает их получателю. При этом гарантируется, что все выборки будут корректно доставлены и воспроизведены на принимающем конце соединения, поскольку ширина полосы пропускания канала связи составляет 64 Кбит/с (65536 бит в секунду), чего достаточно для передачи оцифрованных голосовых данных. Таким образом, преимущество систем с коммутацией каналов — в гарантированной пропускной способности: после того, как соединение между двумя абонентами установлено, на его пропускную способность не могут повлиять другие сетевые соединения. Одним из основных недостатков таких систем является стоимость соединения. Она всегда фиксирована, независимо от того, используется канал или нет. Например, за подключение к телефонной сети при наборе номера взимается фиксированная оплата, даже если абоненты не говорили друг с другом.

Сети с коммутацией пакетов часто применяются для подключения компьютеров. В них используется противоположный подход, чем в сетях с коммутацией каналов. В сетях с коммутацией пакетов передаваемые данные разбиваются на небольшие фрагменты, называемые *пакетами*, которые затем уплотняются и передаются по одному каналу с высокой пропускной способностью, соединяющему множество компьютеров. Размер пакета обычно составляет несколько сотен байт. Для того чтобы сетевое оборудование могло распознать получателя, в каждый пакет, кроме данных, помещается специальная служебная информация. Например, при передаче большого файла между двумя компьютерами, он разбивается на множество пакетов, которые поочередно передаются по сети. Сетевое оборудование доставляет эти пакеты по указанному в них адресу, после чего на принимающем конце специальное программное обеспечение восстанавливает файл из полученных пакетов. Основное преимущество технологии с коммутацией пакетов — возможность одновременного и независимого обмена данными между двумя различными компьютерами, подключенными к одной физической сети. Недостаток подобного вида взаимодействия очевиден. С увеличением активности отдельных узлов сети, пропускная способность общего канала связи падает, поэтому скорость обмена данными между двумя узлами сети будет также падать. Таким образом, при перегрузке сети с коммутацией пакетов ее абоненты вынуждены простоять в ожидании возможности послать очередную порцию пакетов.

Однако несмотря на все недостатки, а также на невозможность гарантировать пропускную способность канала связи сети с коммутацией пакетов стали чрезвычайно популярными. Это объясняется их стоимостью и пропускной способностью. Поскольку несколько компьютеров могут совместно использовать один канал связи, стоимость оборудования существенно ниже по сравнению с сетью на основе коммутации каналов. Кроме того, при использовании современного высокоскоростного сетевого оборудования проблема нехватки пропускной способности канала связи обычно не возникает. Таким образом, в сети с коммутацией пакетов множество компьютеров может обмениваться данными без заметного

---

<sup>1</sup> На самом деле можно построить сеть на основе гибридной технологии. Однако для нас важно только различие в функциональных возможностях.

снижения ее пропускной способности. Поэтому далее в книге, если не сказано обратное, под термином *сеть (network)* мы будем подразумевать сеть с коммутацией пакетов.

## 2.3. Глобальные и локальные сети

Принципы построения сетей, охватывающих большие расстояния (например, тех, что протянулись по территории материковой части США), принципиально отличаются от принципов создания сетей передачи данных на короткие расстояния (например, в пределах комнаты или одного здания). Поэтому, чтобы упростить классификацию сетей с коммутацией пакетов по пропускной способности и назначению, их обычно делят на две большие категории: *глобальные (wide area network, или WAN)* и *локальные (Local Area Networks, или LAN)*. Эти две категории не имеют четких определений. Поэтому производители часто весьма вольно обращаются с терминами, преследуя только одну цель — помочь покупателям сделать правильный выбор среди огромного количества технологий.

Глобальные сети (их иногда называют *протяженными сетями (long haul networks)*) используются для обмена данными на огромных расстояниях. В большинстве таких сетей максимальное расстояние, на котором могут взаимодействовать их абоненты, ничем не ограничивается, т.е. два узла сети могут находиться на произвольном удалении друг от друга. Например, глобальная сеть может охватывать целый континент или соединять компьютеры по обе стороны океана. Обычно считается, что скорость передачи данных в глобальных сетях ниже, чем в локальных, а задержки при передаче пакетов — выше. Как правило, скорость передачи данных по глобальным сетям составляет от 1,5 до 155 Мбит/с (миллионов бит в секунду), а задержки при прохождении пакетов — от нескольких миллисекунд до нескольких десятков секунд<sup>2</sup>.

Локальные сети обеспечивают наивысшую скорость обмена данными между компьютерами, однако зона их действия ограничивается небольшими расстояниями. Например, типичная локальная сеть может охватывать одно здание или небольшой университетский городок. При этом скорость передачи данных может составлять от 10 Мбит/с до 2 Гбит/с (миллиардов бит в секунду). Поскольку локальные сети охватывают небольшие расстояния, задержки, возникающие в них при передаче пакетов данных, существенно меньше, чем в глобальных сетях, и составляют от 0,1 до 10 миллисекунд.

Таким образом, только что мы установили закономерность между скоростью передачи данных и расстоянием: чем выше скорость передачи данных по сети, тем на меньшие расстояния эти данные могут быть переданы. Кроме этой, в рассматриваемых сетевых технологиях существуют и другие закономерности. При построении локальной сети в каждый компьютер обычно устанавливается специальное устройство, называемое *платой сетевого интерфейса (Network Interface Card, или NIC)*, с помощью которого компьютеры напрямую подключаются к сети. При этом к самой сети не предъявляются специальные требования. Дело в том, что многое зависит от типа плат интерфейса, установленных в компьютерах, которые передают и принимают сложные электрические сигналы. Глобальная сеть строится на основе ряда сложных компьютеров, называемых *коммутаторами пакетов (packet switches)*, которые связаны между собой протяженными линиями передачи данных. При этом длину сети можно увеличить, добавив новый коммутатор и новую линию передачи данных. Чтобы подключить компью-

---

<sup>2</sup> Такие большие задержки обычно вызваны использованием спутниковой связи, при которой радиосигнал проходит значительные расстояния от поверхности Земли до орбиты спутника и обратно.

тер пользователя к глобальной сети, нужно соединить его с одним из коммутаторов пакетов. Каждый коммутатор, находящийся на пути передачи данных, вносит определенную задержку при прохождении пакетов. Причина в том, что прием пакета и его последующая передача до следующего коммутатора происходят не мгновенно. Таким образом, чем длиннее глобальная сеть, тем больше времени тратится на передачу данных по ней.

Рассматриваемое в этой книге программное обеспечение скрывает отличия в технологиях передачи данных, применяемых в сетях различных типов. С его помощью удается наладить взаимодействие между сетями так, что оно не будет зависеть от применяемого аппаратного обеспечения. Однако, чтобы понять идеи, заложенные при разработке программ, необходимо хотя бы в общих чертах представлять себе, как работает сетевое аппаратное обеспечение. Ниже рассматриваются основные сетевые технологии, применяемые в Internet, и различия между ними. В последующих главах мы рассмотрим, как при разработке программного обеспечения на основе протокола TCP/IP следует учитывать эти различия и сделать их незаметными для пользователя.

### 2.3.1. Адресация сетевого оборудования

В каждой сетевой технологии определяется так называемый *механизм адресации* (*addressing mechanism*), используемый компьютерами для указания абонента сети, которому посыпается пакет. Каждому компьютеру сети назначается уникальный адрес, который, как правило, является целым числом. Таким образом, в любом пакете, посылаемом по сети, есть так называемое *поле адреса получателя* (*destination address field*), в которое записывается числовой адрес получателя конкретного пакета. Важно то, что адрес получателя располагается на одном и том же месте (поле) во всех пакетах, поэтому сетевое оборудование может легко его распознать. Прежде чем передать пакет по сети, отправитель должен узнать адрес получателя и поместить его в соответствующее поле пакета.

Методика назначения адресов оговаривается в каждой сетевой технологии и зависит от применяемого оборудования. Например, от этого зависит, сколько битов выделяется под поле адреса получателя и где оно располагается в пакете. Следует учитывать, что далеко не во всех сетевых технологиях используются совместимые системы адресации. Ниже в этой главе будут приведены примеры некоторых аппаратных систем адресации, а в последующих главах мы расскажем о том, как в семействе протоколов TCP/IP учитываются различия между ними.

## 2.4. Технология Ethernet

В начале 1970-х годов в исследовательском центре Пало Альто (Palo Alto Research Center, PARC) корпорации Xerox была разработана технология передачи данных по локальным сетям с коммутацией пакетов, которая впоследствии стала одной из самых популярных. Ее называли *Ethernet*. В 1978 году фирмы Xerox Corporation, Intel Corporation и Digital Equipment Corporation приняли эту сетевую технологию в качестве стандарта. Чуть позже *Институт инженеров по электротехнике и электронике* (*Institute of Electrical and Electronics Engineers*, или *IEEE*) выпустил совместимую версию стандарта и присвоил ей номер 802.3. Таким образом Ethernet стала одной из самых популярных технологий передачи данных по локальным сетям. В настоящее время она применяется практически во всех корпоративных сетях и в подавляющем большинстве небольших локальных сетей. Поскольку технология Ethernet пользуется огромной популярностью, было выпущено несколько ее вариантов. И хотя первоначальная схема подключения компьютеров к сети постепенно вытесняется новыми, более совершенными

схемами, ее понимание поможет нам разобраться в принципах работы сети и уяснить некоторые важные конструкторские решения. Поэтому сначала мы рассмотрим первоначальную схему подключения компьютеров к сети, а затем обсудим ее варианты. Официально оригинальная технология Ethernet называется *10Base5*. Для подключения абонентов сети в ней применяется толстый коаксиальный кабель, поперечное сечение которого показано на рис. 2.1.

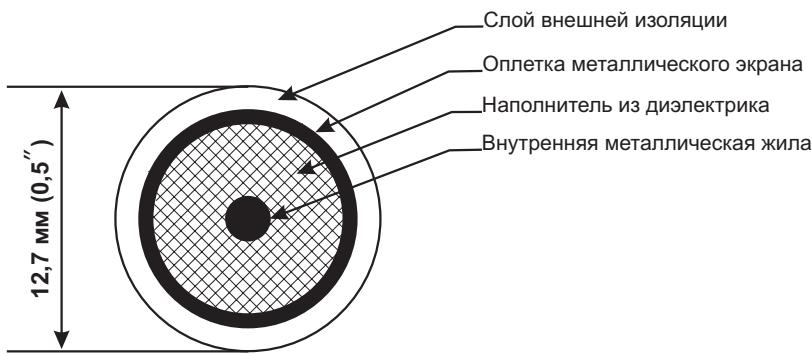


Рис. 2.1. Поперечное сечение толстого коаксиального кабеля, используемого в оригинальной технологии Ethernet

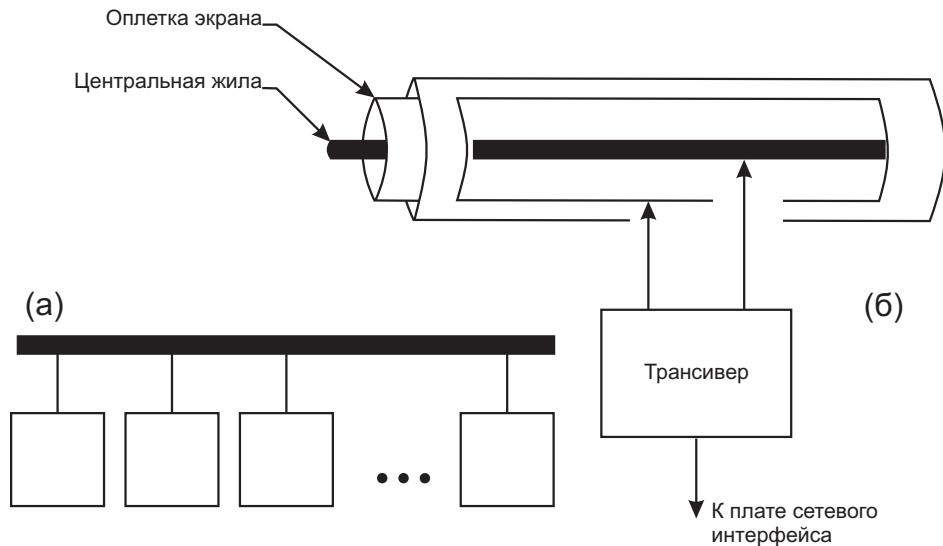
Кабель называется *средой передачи данных* или *эфиром (ether)* и является полностью пассивным элементом. Все активные электронные компоненты, необходимые для функционирования сети, сосредоточены в платах сетевого интерфейса, установленных в компьютерах. Диаметр кабеля Ethernet составляет 12,7 мм (0,5"), а длина сегмента может достигать 500 м. Для того чтобы предотвратить отражение радиосигнала на концах кабеля, между центральной жилой и экраном подключают согласующие резисторы, сопротивление которых равно волновому сопротивлению кабеля.

Для подключения компьютера к оригинальному кабелю Ethernet требовалось специальное устройство, называемое *трансивером (transceiver)*, или *приемопередатчиком*. Физически подключение трансивера ко внутренней жиле коаксиального кабеля Ethernet осуществлялось через небольшое отверстие, проеданное в наружной оболочке, как показано на рис. 2.2. Специалисты для такого типа подключения часто используют термин *отвод (tap)*. Обычно в трансивере устанавливали небольшие металлические штыри, с помощью которых осуществлялся электрический контакт с центральной жилой и металлической оплеткой экрана. Некоторые производители предлагали для подключения трансивера использовать специальные Т-образные ответвители. При этом кабель разрезался на небольшие сегменты, и на его концах устанавливались коаксиальные разъемы.

Для подключения компьютера к оригинальному кабелю Ethernet требовалось два основных электронных компонента: трансивер и плата сетевого интерфейса. С помощью трансивера компьютер подключался к центральной жиле и экрану коаксиального кабеля. В его функции входили прием и отправка сигналов в передающую среду. *Плата сетевого интерфейса*, или *сетевой адаптер*, подключался непосредственно к системнойшине компьютера (например, вставлялся в один из свободных слотов расширения материнской платы) и трансиверу.

Трансивер представлял собой небольшое электронное устройство, обычно располагавшееся в непосредственной близости от кабеля. Кроме аналоговой части, которая принимала и отправляла электрические сигналы в кабель, в состав

трансивера входил цифровой блок, с помощью которого выполнялся обмен данными с компьютером. Трансивер улавливал аналоговые электрические сигналы, проходящие по кабелю, выделял из них цифровые данные и отправлял их в компьютер, а также выполнял обратное преобразование цифровых данных, полученных от компьютера в аналоговые и отправлял их в эфир.



*Рис. 2.2. Схематическое изображение сети Ethernet, к которой подключено несколько компьютеров (а); электрическая схема подключения трансивера к оригинальному кабелю Ethernet (б);*

Трансивер подключался к сетевому адаптеру компьютера с помощью специального интерфейсного кабеля подключаемых устройств, или AUI-кабеля (*Attachment Unit Interface*). На техническом языке AUI-кабель назывался *кабелем трансивера*. Он состоял из ряда проводников, по которым проходили электрические сигналы управления трансивером, пакеты передаваемых и принимаемых данных, а также подавалось напряжение питания, необходимое для работы трансивера. На рис. 2.3 показана схема подключения сетевых компонентов к системной шине компьютера и кабелю Ethernet.

Каждый сетевой адаптер управлял одним трансивером согласно командам, получаемым от программного обеспечения компьютера. С точки зрения операционной системы, сетевой интерфейс представлял собой устройство ввода-вывода, на которое от компьютера поступали команды передачи данных и сами данные. Плата сетевого интерфейса передавала эти данные трансиверу и далее в эфир, генерировала сигнал прерывания процессору после выполнения операции и сообщала о состоянии сетевого устройства. По сравнению с трансивером, представлявшим собой довольно простое устройство, плата сетевого интерфейса была существенно сложнее. В некоторых платах интерфейса использовался встроенный микропроцессор, который управлял обменом информацией между памятью компьютера и эфиром.

На практике, в организациях, где использовалась оригинальная технология Ethernet для офисных применений, коаксиальный кабель прокладывался под потолком помещений. В местах, где располагались компьютеры, организовывались точки подключения к общему кабелю. Общая схема подключения компьютеров к сети Ethernet показана на рис. 2.4.

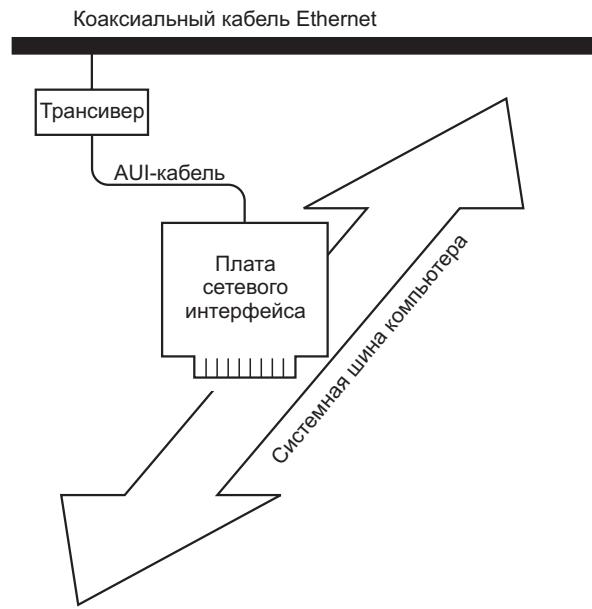


Рис. 2.3. Оригинальная схема подключения шины компьютера к кабелю Ethernet с помощью двух основных электронных компонентов. По AUI-кабелю на трансивер подавалось напряжение питания, сигналы управления и передавались или принимались пакеты данных

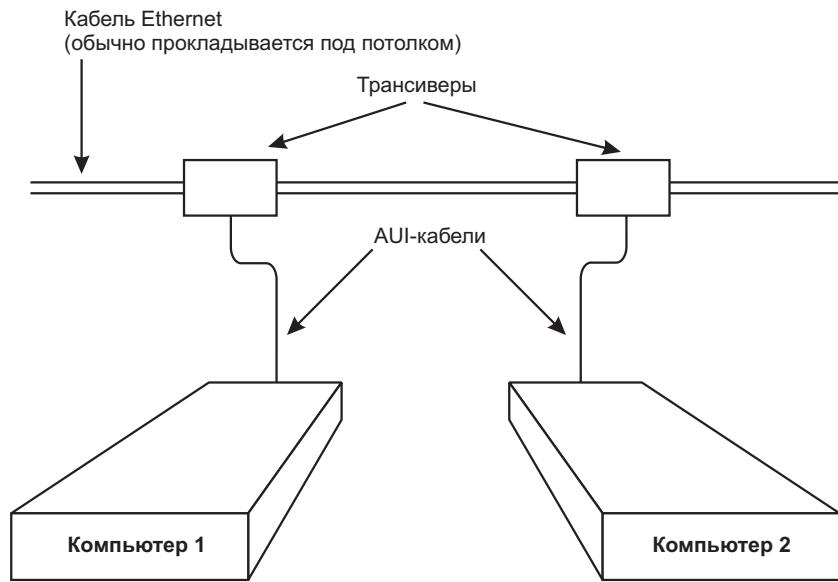


Рис. 2.4. Схема физического подключения двух компьютеров к сети Ethernet с использованием оригинального коаксиального кабеля. В офисных помещениях кабель обычно прокладывался под потолком. В местах расположения компьютеров к коаксиальному кабелю подключался трансивер, который присоединялся к плате сетевого интерфейса с помощью специального AUI-кабеля

### 2.4.1. Тонкий Ethernet

Оригинальная технология Ethernet имела существенные недостатки. Например, поскольку трансивер являлся электронным устройством, он имел определенную отличную от нуля стоимость, которая умножалась на количество компьютеров, подключенных к сети. Более того, поскольку трансиверы располагались в непосредственной близости от кабеля, а не от компьютеров, задача поиска и замены таких устройств с случае выхода их из строя была не из легких. Кроме того, прокладка коаксиального кабеля также была затруднена. Дело в том, что для защиты кабеля от внешних электромагнитных помех, возникающих при работе электрических устройств, таких как, например, электродвигатели, экран коаксиального кабеля изготавливали достаточно толстым и плотным, что затрудняло его изгиб при прокладке. Ну и, наконец, AUI-кабель также был достаточно толстым и трудно поддавался укладке.

Поэтому, чтобы уменьшить стоимость оборудования, необходимого для создания локальных сетей, работающих при небольшом уровне электромагнитных помех (например, в офисе), был разработан альтернативный вариант подключения к сети Ethernet. Официально его назвали *10Base2*, но чаще всего специалисты называли новую технологию подключения *тонким Ethernet* (*thin-wire Ethernet*) или *сетью Ethernet с тонким коаксиальным кабелем*<sup>3</sup> (*thinnet*). Альтернативный коаксиальный кабель был тоньше, дешевле и более гибкий, хотя и не лишен недостатков. Поскольку тонкий коаксиальный кабель не обеспечивал эффективной защиты от электромагнитных помех, его нельзя было прокладывать рядом с мощным электрическим оборудованием, которое используется в промышленных целях. Кроме того, по сравнению с толстым Ethernet тонкий кабель обеспечивал гораздо меньшую дальность связи и позволял подключать к локальной сети меньшее количество компьютеров.

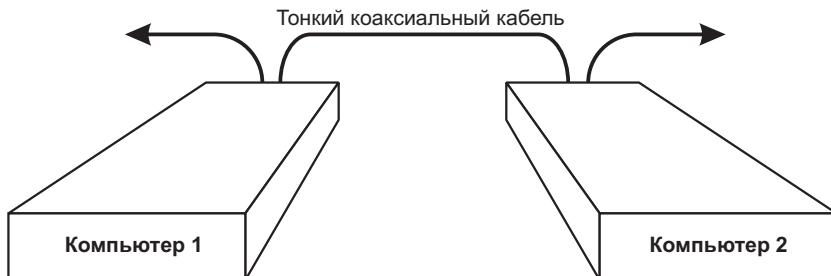


Рис. 2.5. Схема физического подключения двух компьютеров к сети на основе тонкого кабеля Ethernet. Компьютеры соединяются друг с другом напрямую, без использования трансивера

При разработке новой технологии Ethernet достаточно дорогостоящий трансивер был заменен специальными высокоскоростными цифровыми схемами, которые располагались прямо на плате сетевого интерфейса. Это позволило подключать компьютеры напрямую к коаксиальному кабелю. Таким образом, при использовании технологии тонкого Ethernet в компьютере находились и плата сетевого интерфейса и специальное устройство, позволяющее непосредственно подключаться к коаксиальному кабелю. Производители небольших компьютеров и рабочих станций сочли технологию тонкого Ethernet очень привлекательной, поскольку она позволяла легко встраивать сетевое оборудование в компьютеры

<sup>3</sup> Для отличия оригинальный коаксиальный кабель стали называть толстым Ethernet, а оригинальную технологию — сетью Ethernet на основе толстого коаксиального кабеля (*thicknet*).

на базе микропроцессоров. При этом сетевой разъем располагался на задней стенке системного блока компьютера.

Поскольку компьютеры подключались непосредственно к тонкому коаксиальному кабелю, монтаж сети облегчался, когда в одной комнате находились много компьютеров. Тогда достаточно было протянуть кабель от одного компьютера к другому, а чтобы подключить к сети новый компьютер, достаточно было добавить к цепочке новое звено. Общая схема подключения компьютеров к сети на основе тонкого кабеля Ethernet показана на рис. 2.4.

Технология тонкого Ethernet была разработана для того, чтобы облегчить подключение и отключение компьютеров от сети. При этом использовались коаксиальные разъемы, фиксирующиеся поворотом замка на 90°, так называемые *BNC-разъемы*<sup>4</sup> (*BNC connectors*). Для подключения кабеля к компьютеру не требовались специальные устройства. Таким образом, с этой работой самостоятельно мог справиться любой пользователь, не прибегая к помощи специалиста. Конечно, в этом есть и определенный недостаток, поскольку, если пользователь случайно разъединял кабель, вся сеть полностью выходила из строя. Однако во многих случаях преимущества технологии тонкого Ethernet перевешивали ее недостатки.

#### 2.4.2. Сеть Ethernet на основе витой пары

Современные технологические достижения сделали возможным создание такой сети Ethernet, где не требуется экранировать сигнальные провода внешним электромагнитным экраном, как в коаксиальном кабеле. Новую технологию называли *Ethernet на основе витой пары* (*twisted pair Ethernet*). Она позволяла использовать в качестве эфира обычные неэкранированные медные провода, наподобие тех, что применяются в телефонии<sup>5</sup>. Сеть на основе витой пары обладает двумя важными преимуществами, по сравнению с тонким Ethernet: она дешевле и защищена от случайных повреждений сети при отключении одной машины или повреждении сетевого кабеля, ведущего к компьютеру. В некоторых случаях технология витой пары позволяет создать сеть Ethernet на основе существующей в организации проводки. Кроме того, прокладка витой пары (ее называют *кабелем 5-й категории*) обходится дешевле и проще, чем коаксиального кабеля.

Официально технология на основе витой пары называется *10Base-T*. Первые сети с витой парой работали на скорости 10 Мбит/с, как и сети с толстым или тонким Ethernet. Подключение компьютеров к сети Ethernet на основе витой пары осуществляется через специальное устройство, которое называется *концентратором*, или *хабом* (*hub*), как показано на рис. 2.6. При этом используется восьмижильный (четырехпарный) кабель.

Концентратор — это электронное устройство, которое имитирует сигналы в кабеле Ethernet. Он представляет собой небольшую коробку, которая обычно размещается в монтажном шкафу или распределительной коробке. Длина соединительного провода между компьютером и концентратором не должна превышать 100 м. Для работы концентратора требуется источник питания. Кроме того, некоторые концентраторы позволяют обслуживающему персоналу следить за их работой и состоянием прямо по сети. С точки зрения платы сетевого интерфейса, подключение к концентратору аналогично подключению к трансиверу.

<sup>4</sup> Аббревиатура *BNC* расшифровывается двояко: *Bayonet Nut Connector*, или *миниатюрный байонетный соединитель*, и *British Naval Connector*, или *английский морской соединитель*. — Прим. ред.

<sup>5</sup> Термин “витая пара” возник не случайно; он взят из телефонии, где сигнальные провода скручивали для уменьшения перекрестных помех между соседними каналами связи.

Другими словами, концентратор обеспечивает такие же возможности в плане передачи данных, как и толстый или тонкий Ethernet. Просто с помощью концентратора реализуется альтернативная схема подключения компьютеров к сети.

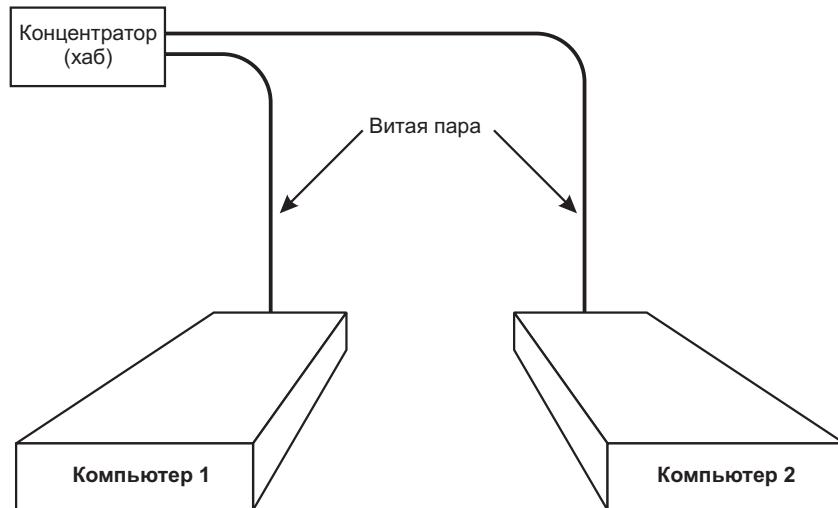


Рис. 2.6. Схема сети Ethernet на основе витой пары. Каждый компьютер подключается к концентратору с помощью четырехпарного кабеля

#### 2.4.3. Пропускная способность сети Ethernet

За время своего существования большая часть элементов оригинальной технологии Ethernet не претерпела глубоких изменений, несмотря на то, что схема подключения компьютеров к сети изменялась при переходе от оригинального толстого кабеля Ethernet к тонкому, а затем — к витой паре. Об этом свидетельствует и то, что первоначально скорость обмена данными по витой паре была такой же, как и у толстого кабеля Ethernet (10 Мбит/с). И хотя современные компьютеры без особых проблем могут генерировать такой поток данных, истинной скоростью работы сети нельзя считать ту, с которой два компьютера могут обмениваться данными между собой. Скорость работы сети должна оцениваться в зависимости от ее пропускной способности. Представьте себе, что сеть — это автомагистраль, соединяющая между собой несколько городов (компьютеров), а пакеты — движущиеся по ней автомобили. Так вот, высокая пропускная способность сети означает, что по шоссе может одновременно двигаться большой поток машин. Низкая пропускная способность говорит о том, что автомагистраль не может справиться с существующим потоком автотранспорта. Например, сеть Ethernet с пропускной способностью 10 Мбит/с может одновременно обслуживать либо несколько компьютеров, генерирующих большой поток данных, либо множество компьютеров, создающих небольшой трафик в сети.

В конце 1970-х годов, когда проводилась стандартизация технологии Ethernet, пропускная способность локальной сети, равная 10 Мбит/с, была более чем достаточна для обслуживания множества компьютеров, поскольку существующие на то время скорости работы центральных процессоров компьютеров и плат интерфейса не позволяли обмениваться данными с большими скоростями. Однако к середине 1990-х годов производительность микропроцессоров существенно выросла, что

отразилось на интенсивности использования сетей. В результате сеть Ethernet с пропускной способностью 10 Мбит/с уже не могла использоваться в качестве магистральной не то что в крупных, а даже в средних корпорациях, поскольку она не справлялась с резко возросшим потоком данных. Таким образом, узким местом в технологии Ethernet стала ее пропускная способность.

#### 2.4.4. Технология быстрого Ethernet

Чтобы увеличить пропускную способность сети Ethernet, инженеры разработали новую версию этого стандарта, в которой скорость обмена данными была увеличена на порядок. Официально новую технологию назвали *100Base-T*, но чаще всего ее стали называть *быстрым Ethernet (Fast Ethernet)*. Как следует из официального названия, в технологии быстрого Ethernet использовался кабель витой пары 5-й категории, как и в технологии 10Base-T. Однако благодаря особому режиму использования проводников кабеля, удалось повысить скорость обмена данными до 100 Мбит/с.

Чтобы понять, как удалось увеличить пропускную способность сети на порядок, необходимо иметь в виду два момента. Во-первых, несмотря на то, что скорость работы компьютеров существенно возросла, передавать непрерывный поток данных со скоростью 100 Мбит/с могут только очень немногие компьютерные системы. И, во-вторых, стандарт 100Base-T никак не повлиял на большую часть компонентов технологии Ethernet. В частности, величина максимального размера пакета осталась такой же, как и в технологии 10Base-T. Все это свидетельствует о том, что технология быстрого Ethernet не оптимизирована для достижения максимально возможной скорости обмена между двумя абонентами сети. Она рассчитана на охват большего количества рабочих станций и большой общий трафик в сети.

#### 2.4.5. Технология 10/100 Ethernet

Вскоре после появления технологии быстрого Ethernet производители стали выпускать комбинированные платы сетевого интерфейса, поддерживающие скорости передачи данных как 10, так и 100 Мбит/с. Новую технологию назвали *двухскоростным Ethernet (dual-speed Ethernet)*, или *10/100 Ethernet*. Кроме сетевых плат, ее стали использовать и в концентраторах. В этом нет ничего необычного, поскольку в сущности все сетевые платы стандарта 100Base-T передавали дополнительные сигналы, благодаря которым оборудование, установленное на другом конце провода могло определить тип сетевой платы, от которой пришел пакет. Более того, если на концах кабеля в разъемах RJ-45 задействованы все восемь проводов, такой кабель можно использовать для подключения как устройств 10Base-T, так и 100Base-T.

И хотя оборудование, поддерживающее стандарт 10/100 Ethernet, было немного дороже, чем оборудование стандарта 10Base-T, оно стало очень популярным. Двухскоростные сетевые платы оказались весьма кстати при модернизации сети и плавном переходе на 100-мегабитовую технологию. Например, если компьютер с двухскоростной сетевой платой стандарта 10/100 Ethernet подключить к концентратору 10Base-T, специальная схема в плате автоматически определит возможную скорость передачи данных и перейдет на работу в 10-мегабитовом режиме. Если же впоследствии компьютер будет подключен к концентратору 100Base-T, то оборудование опять же автоматически определит новую скорость передачи данных и перейдет в 100-мегабитовый режим. Следует заметить, что изменение скорости передачи данных происходит полностью автоматически; при этом не нужно перенастраивать какое бы то ни было программное или аппаратное обеспечение.

#### **2.4.6. Гигабитовый Ethernet**

К концу 1990-х годов, по мере распространения технологии 100Base-T, стало очевидным, что в отдельных случаях пропускной способности канала связи в 100 Мбит/с явно недостаточно. Поэтому технология Ethernet была доработана, в результате чего скорость передачи данных возросла до 1 Гбит/с (миллиардов бит в секунду). Ее назвали *1000Base-T*. Новая технология стала особенно привлекательной для создания корпоративных магистральных сетей, по которым проходит большой поток данных, поступающий от множества компьютеров. Однако при повышении скорости передачи данных снижается помехоустойчивость канала связи. Гигабитовый Ethernet оказался слишком восприимчивым к электромагнитным помехам. Поэтому кабель, без проблем работавший на скоростях 10 и даже 100 Мбит/с, не годился для технологии 1000Base-T.

Подобно быстрому Ethernet, технология гигабитового Ethernet оптимизирована под высокую пропускную способность сети. Поэтому были сохранены первоначальные форматы пакетов и их максимальные размеры. Таким образом, пакеты данных, проходящие по сетям 10Base-T, 100Base-T и 1000Base-T, совместимы друг с другом. Следовательно, можно сконцентрировать потоки данных, проходящие по десяти загруженным сетям 100Base-T, в один поток и передать его по одной сети 1000Base-T.

#### **2.4.7. Отличительные особенности технологии Ethernet**

При разработке технологии Ethernet ставилось несколько задач: она должна быть построена по принципу общей шины и поддерживать широковещательный режим передачи данных, обеспечивать негарантированную доставку пакетов (*best-effort*) и распределенное управление доступом. В сети Ethernet используется топология, которая называется *общей шиной*, поскольку все рабочие станции подключаются к одному совместно используемому каналу связи. Кроме того, сеть Ethernet построена по принципу *широковещательной передачи данных* (*broadcast technology*), т.е. абоненты сети получают все посланные в эфир пакеты. Это позволяет передать пакет сразу всем абонентам сети. О том, как переслать пакет от одной рабочей станции к другой или к их группе, речь пойдет чуть ниже. А пока достаточно понять, что на самом нижнем уровне в сети Ethernet находится оборудование, которое не различает адресатов в передаваемых пакетах. Другими словами, от концентратора пакеты поступают на все подключенные к нему сетевые платы, а плата уже “решает”, какие пакеты должен принять компьютер, в котором она установлена. В технологии Ethernet не предпринимаются специальные усилия для доставки пакетов. Поскольку сетевому оборудованию отправителя не передается информация о доставке пакета получателю, нельзя гарантировать, что пакет будет доставлен получателю без проблем. Типична, например, ситуация, когда компьютер получателя выключен. При этом все посланные ему пакеты теряются, а отправитель об этом даже не подозревает. Ниже будет описано, как в семействе протоколов TCP/IP решается проблема негарантированной доставки пакетов сетевым оборудованием.

В технологии Ethernet применяется распределенное управление доступом, так как, в отличие от некоторых других сетевых технологий, в нем не предусмотрена централизованная система предоставления доступа. В Ethernet применяется система доступа, называемая *множественным доступом с контролем несущей и обнаружением коллизий* (*Carrier Sense Multiple Access with Collision Detect*, или *CSMA/CD*). Название *множественный доступ с контролем несущей* говорит о том, что одновременный доступ к сети Ethernet имеет множество компьютеров. При этом каждая машина определяет, свободен ли эфир, по наличию несущей частоты в кабеле. Когда сетевая плата собирается передать пакет данных,

она проверяет, не передается ли по сети в этот момент кем-либо другой пакет (т.е. выполняет контроль несущей). Если несущая в кабеле не обнаружена, сетевая плата начинает передачу данных. Процесс передачи пакета ограничен во времени, поскольку его длина конечна и не может превышать заранее оговоренного значения, называемого максимальным размером пакета. Кроме того, время, прошедшее после предыдущей отправки пакета сетевой платой, не должно быть меньше заранее установленного значения. Это сделано для того, чтобы предотвратить монопольное использование сети одним компьютером и предоставить доступ к сети другим абонентам.

#### 2.4.8. Обнаружение коллизий и методика восстановления

Посланный сетевой платой при передаче данных сигнал не достигает всех абонентов сети одновременно. Он передается по медным проводам со скоростью, составляющей примерно 70% от скорости света. Поэтому возможна ситуация, когда две сетевые платы, находящиеся на некотором удалении друг от друга, обнаружат, что эфир свободен, и одновременно начнут передачу данных. При этом в кабеле возникают так называемые перекрестные помехи, в результате чего оба посланных сигнала сильно искажаются. Такое явление называется *коллизией* (*collision*).

В сетях Ethernet коллизии обрабатываются очень интересным способом. Каждая сетевая плата во время передачи данных контролирует эфир на предмет наличия в нем посторонней несущей и возникновения перекрестных помех. На техническом языке этот процесс называется *обнаружением коллизий* (*collision detection*, или *CD*). Таким образом, Ethernet — это сеть с множественным доступом, контролем несущей и обнаружением коллизий (CSMA/CD). При обнаружении коллизии сетевая плата останавливает передачу данных, ожидает момента, когда активность других абонентов сети спадет, после чего выполняет повторную попытку передачи данных. При этом должны соблюдаться определенные правила, иначе сеть будет перегружена помехами от других сетевых плат, безуспешно пытающихся передать данные, поскольку каждая передача будет приводить к коллизиям. Чтобы избежать подобных ситуаций, в технологии Ethernet используется стратегия *двойной экспоненциальной задержки* (*binary exponential backoff*). Суть ее состоит в том, что после возникновения первой коллизии отправитель повторяет попытку передачи данных через случайно выбранный интервал времени. Если вторая попытка передачи данных также привела к коллизии, отправитель увеличивает выбранный в первый раз интервал задержки в два раза и вновь пытается передать данные. В случае неудачи время задержки увеличивается в четыре раза и т.д. Смысл введения экспоненциальной задержки состоит в том, что после возникновения коллизии велика вероятность того, что несколько сетевых плат начнут повторную передачу данных одновременно, а это приведет к еще большим помехам в сети. При этом велика вероятность того, что несколько сетевых плат выберут близкие времена задержек. Поэтому вероятность возникновения повторных коллизий в эфире также остается высокой. Таким образом, благодаря удвоению случайно выбранного значения интервала, стратегия экспоненциальной задержки позволяет рассеять попытки повторной передачи данных на достаточно большом промежутке времени, что существенно снижает вероятность возникновения дальнейших коллизий.

#### 2.4.9. Адресация оборудования в сетях Ethernet

В сетях Ethernet принят 48-битовый механизм адресации. Каждой плате сетевого интерфейса при изготовлении назначается уникальное 48-битовое число, которое называется *адресом Ethernet* (*Ethernet address*). Для этого производители

оборудования приобретают блоки адресов Ethernet<sup>6</sup> и назначают из них уникальные адреса каждой вновь произведенной сетевой плате. Таким образом, не может существовать двух плат сетевого интерфейса с одинаковыми адресами Ethernet.

Как правило адрес Ethernet записывается в плату сетевого интерфейса в форме, удобной для обработки компьютером. Поскольку адреса Ethernet относятся к сетевому оборудованию, иногда их называют *аппаратными адресами (hardware addresses)*, *физическими адресами (physical addresses)*, *адресами доступа к среде передачи данных (media access (MAC) addresses)*, а также адресами второго уровня (*layer 2 addresses*). Обратите внимание на одну важную особенность физического адреса Ethernet.

*Физический адрес назначается плате сетевого интерфейса Ethernet. Поэтому при замене в компьютере этой платы по причине ее отказа или установки новой платы соответственно, изменяется и физический адрес этого компьютера.*

В дальнейшем при рассмотрении сетевых протоколов высокого уровня мы остановимся на том, как в их реализациях учитывается возможность изменения физического адреса Ethernet.

Плата сетевого интерфейса анализирует полученные пакеты и “решает”, какие из них предназначены для компьютера, в котором она установлена. Напомним, что в сетях Ethernet копии прошедших через концентратор пакетов рассылаются всем подключенным к нему сетевым платам, даже если пакет адресован только одному из компьютеров. Поэтому плата сетевого интерфейса для фильтрации пакетов использует поле адреса получателя. Она игнорирует пакеты, адресованные другим машинам и пропускает только те, которые предназначены “ее” компьютеру. Понятно, что если бы сетевая плата передавала все полученные пакеты центральному процессору компьютера на фильтрацию и дальнейшую обработку, то при большом трафике, он вряд ли бы успел обработать все полученные пакеты в реальном масштабе времени. Поэтому чтобы избежать перегрузки компьютера входящим потоком данных, был введен механизм адресации, а фильтрация пакетов перенесена на уровень сетевого оборудования. Перенос фильтрации пакетов на аппаратный уровень преследовал и еще одну цель — освободить от этой работы центральный процессор. В результате производительность подключенных к сети компьютеров не будет снижаться при увеличении общего трафика в сети.

Выше мы уже говорили о том, что 48-битовый адрес Ethernet идентифицирует компьютер получателя. При этом адреса бывают трех типов.

- *Физический адрес одной сетевой платы, или одноадресатный адрес (unicast address).*
- *Широковещательный сетевой адрес (broadcast address).*
- *Многоадресатный сетевой адрес (multicast address).*

В соответствии с принятым соглашением, широковещательный адрес, состоящий из 48 единиц, зарезервирован для одновременной рассылки пакетов всем компьютерам. С помощью многоадресатных адресов можно организовать усеченную форму широковещания в сети. При этом компьютеры, составляющие группу (она называется *многоадресатной*, или *multicast group*), необходимо настроить так, чтобы они реагировали на получение пакетов, посланных по указанному многоадресатному адресу. Для подключения к многоадресатной группе

<sup>6</sup> Распределением адресного пространства Ethernet занимается Институт инженеров по электротехнике и электронике (IEEE), который по мере необходимости выделяет производителям сетевого оборудования свободные блоки адресов.

компьютер должен сообщить ее адрес сетевой плате и перевести плату в режим многоадресатной работы. Преимущество использования многоадресатной группы заключается в том, что имеется возможность сделать выборочную широковещательную передачу только на нужные компьютеры. При этом посылаемый по многоадресатному адресу пакет будет принят только компьютерами, входящими в группу, другие же компьютеры сети его не получат.

Для поддержки режимов широковещательной и многоадресатной адресации, плата сетевого интерфейса Ethernet на аппаратном уровне должна распознавать не только свой физический адрес. Обычно сетевые платы принимают два вида пакетов: посланные по ее физическому (одноадресатному) адресу и те, что посланы по сетевому широковещательному адресу. Некоторые сетевые платы можно запрограммировать на распознавание многоадресатных и, даже, альтернативных физических адресов. При начальной загрузке компьютера операционная система инициализирует сетевую плату Ethernet и назначает ей набор адресов, которые она должна распознавать. После этого сетевая плата на аппаратном уровне проверяет в каждом пакете поле адреса получателя и передает пакет для дальнейшей обработки компьютеру только в том случае, если адрес в пакете совпал с одним из запрограммированных адресов платы.

#### 2.4.10. Формат фрейма Ethernet

Стандарт Ethernet следует считать средством обеспечения соединения между абонентами сети, относящимся к канальному уровню (link-level layer). Поэтому имеет смысл рассматривать передаваемые по сети данные как состоящие из *фреймов*<sup>7</sup> (*frame*), или кадров. Фреймы Ethernet имеют переменную длину, однако их размер не может быть меньше 64 октетов<sup>8</sup> и превышать 1518 октетов (с учетом преамбулы фрейма, передаваемых данных и кодов циклической проверки CRC). Как и в любой другой сети с коммутацией пакетов, в каждом фрейме Ethernet предусмотрено специальное поле для адреса получателя. Помимо адреса получателя, фрейм Ethernet содержит также физический адрес отправителя (рис. 2.7).

Адрес Преамбула получателя	Адрес отправителя	Тип фрейма	Данные	Поле CRC
8 октетов	6 октетов	6 октетов	2 октета	46–1500 октетов

Рис. 2.7. Формат фрейма (пакета) Ethernet. Перед передачей по сети фрейм дополняется специальной преамбулой (рисунок сделан без соблюдения масштаба полей)

Таким образом, кроме полей адреса отправителя и получателя, в каждый передаваемый по сети Ethernet фрейм добавляется преамбула, поле типа, поле данных и код циклического избыточного контроля (Cyclic Redundancy Check, или CRC). Преамбула состоит из 64-битовой последовательности нулей и единиц и предназначена для облегчения синхронизации принимающей сетевой платы. Поле CRC (его размер 32 бита) служит для обнаружения ошибок при передаче данных на приемном конце. При этом отправитель вычисляет код циклического избыточного контроля как функцию от данных, находящихся во

<sup>7</sup> Термин *фрейм* (кадр) взят из терминологии, использовавшейся для описания процесса передачи данных по последовательному каналу связи. При этом отправитель разделял данные на фрагменты (фреймы) и перед передачей фрейма помещал до и после него специальные коды.

<sup>8</sup> Формально для описания размера символа, зависящего от применяемого оборудования, используется термин *байт*. Однако в телекоммуникации используется другой термин — *октет*, который, в отличие от байта, не зависит от применяемого оборудования и всегда состоит из 8 бит.

фрейме, и помещает их в передаваемый пакет. Получатель снова вычисляет код CRC и сравнивает его с находящимся во фрейме. Если коды совпадают, принимающая сторона считает, что пакет получен без искажений.

Поле типа состоит из 16-битового целого числа, которое предназначено для идентификации типа данных, передаваемых во фрейме. С точки зрения построения процесса межсетевого взаимодействия поле типа очень важно, поскольку оно позволяет автоматически распознавать фреймы Ethernet. Другими словами, операционная система на основе поля типа полученного фрейма определяет программный модуль протокола, который должен обработать поступивший пакет данных. Одно из ключевых преимуществ автоматического распознавания фреймов заключается в том, что на одной и той же машине можно использовать несколько различных протоколов, а также передавать по физической сети пакеты данных разных протоколов без их взаимного влияния. Благодаря полю типа на одном и том же компьютере могут свободно сосуществовать приложения, использующие как семейство протоколов TCP/IP, так и любые другие протоколы, например те, что только разрабатываются и находятся на стадии тестирования. Таким образом, операционная система анализирует поле типа каждого полученного фрейма и определяет, как должно обрабатываться его содержимое. Ниже мы увидим, как автоматически опознаваемые фреймы Ethernet используются в семействе протоколов TCP/IP для выделения пакетов разных протоколов.

#### 2.4.11. Удлинение сегментов сети Ethernet с помощью повторителей

Выше мы уже говорили о том, что максимальная длина сегмента сети, построенной с использованием оригинального кабеля Ethernet, не должна превышать 500 м (для тонкого Ethernet — 185 м, для витой пары — 100 м). Однако

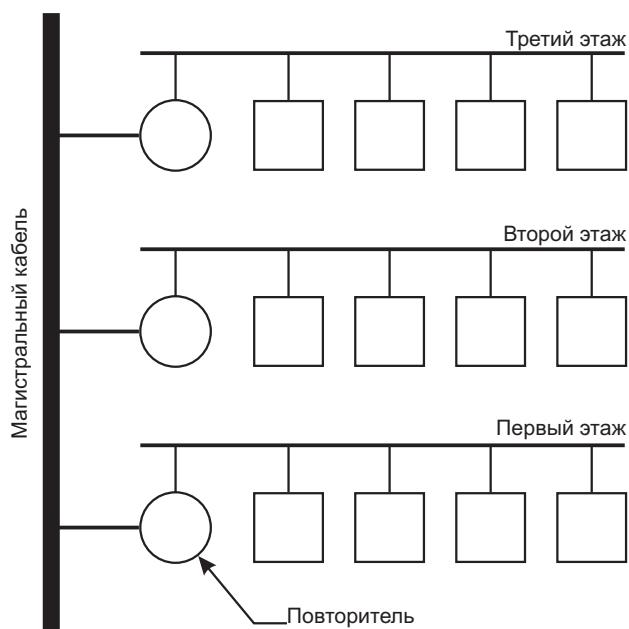


Рис. 2.8. Для подключения дополнительных кабелей Ethernet к магистральному каналу офиса используются повторители. Между двумя абонентами сети должно находиться не более двух повторителей

сегмент сети можно удлинить, использовав одно из двух устройств — *повторитель* (*repeater*) или *мост* (*bridge*). *Повторитель* (*repeater*) — это электронное устройство, которое обрабатывает аналоговый сигнал, проходящий по сетевому кабелю. Он чем-то напоминает концентратор, применяющийся для сетей Ethernet на основе витой пары. Повторитель ретранслирует электрические сигналы из одного кабеля в другой. В оригинальной схеме подключения к сети Ethernet повторитель помещался между двумя отрезками коаксиального кабеля, что позволяло удвоить общую длину сегмента. Для сохранения временных задержек на заданном в спецификации CSMA/CD уровне в стандарте Ethernet оговаривается максимальное количество используемых в сети повторителей: между двумя абонентами сети не должно быть больше двух повторителей. На рис. 2.8 показана типовая схема разводки сети для среднего офиса. Кабель, составляющий магистральную сеть офиса, прокладывается по зданию вертикально через все этажи. Для подключения к нему компьютеров, находящихся на каждом этаже здания, используется дополнительный кабель и повторитель. Таким образом, компьютеры подключаются непосредственно к кабелю, проложенному по этажу, а не к магистральному кабелю.

#### 2.4.12. Удлинение сегментов сети Ethernet с помощью мостов

Объединение двух сетей Ethernet с помощью моста более предпочтительно, чем с помощью повторителя или концентратора, поскольку мост ретранслирует полученные по сети пакеты, а не аналоговый сигнал, проходящий по кабелю. В частности, мост не ретранслирует помехи в кабеле, пакеты, содержащие ошибочные или искаженные фреймы. Прежде чем ретранслировать фрейм в другой сегмент, мост должен получить его без ошибок из первого сегмента. Кроме того, каждый сеанс связи между мостом и сетью Ethernet должен отвечать спецификации CSMA/CD. Следовательно, возникающие в одном сегменте коллизии и связанные с этим задержки распространения пакетов никак не влияют на пакеты, проходящие по другому сегменту. Поэтому с помощью мостов можно объединять практически неограниченное количество сегментов Ethernet. Однако не следует забывать приведенное ниже правило.

*Мосты не являются устройствами для межсетевого взаимодействия. Другими словами, сегменты, объединенные с помощью мостов, функционируют так же, как один большой сегмент Ethernet.*

Сети, в которых используются мосты, относятся к классу *сквозных* (*transparent*), или “прозрачных” для компьютера, поскольку для него не имеет значения, сколько мостов установлено на конкретном сегменте сети. Для обмена данными с другими компьютерами, подключенными через мост, используется такое же оборудование, формат фрейма и методика, как и по обычному локальному сегменту.

Большинство мостов могут не только ретранслировать фреймы между двумя физическими кабелями сети, но и принимать решения о том, какие из фреймов следует ретранслировать, а какие — нет. Подобные устройства называются *адаптивными* (*adaptive*), или *обучающимися* (*learning*) мостами. Адаптивный мост представляет собой простейший компьютер с двумя платами сетевого интерфейса, который работает по определенной программе. Эта программа ведет два списка адресов — по одному для каждой сетевой платы, или интерфейса. Как только первая сетевая плата моста получает фрейм, из него извлекается 48-битовый физический адрес отправителя, который заносится в список, относящийся к первому интерфейсу моста. Аналогично, когда вторая сетевая плата моста получает фрейм, адрес его отправителя заносится в список, относящийся

ко второму интерфейсу моста. В результате через некоторое время адаптивный мост сформирует два списка адресов машин, подключенных к первому и второму сегментам моста.

После занесения в список адреса отправителя адаптивный мост сверяет со своими списками адрес получателя фрейма и определяет, нужно ли ретранслировать фрейм в другой сегмент. Если оказывается, что адрес получателя находится в одном списке с адресом отправителя, мост не ретранслирует фрейм. Если же адрес отправителя не найден в этом списке (т.е. фрейм посыпается по широковещательному либо многоадресатному адресу, или мост еще не успел определить, в каком из сегментов находится машина получателя фрейма), фрейм ретранслируется в другой сегмент сети Ethernet.

Преимущества адаптивного моста очевидны. Адаптивный мост полностью автономен, поскольку сам выполняет поиск физических адресов, полученных из фреймов сетевого трафика. Другими словами, системному администратору не нужно вручную конфигурировать адаптивный мост и заносить в него список физических адресов сетевых плат. Кроме того, так как адаптивный мост не выполняет без особой надобности ретрансляцию фреймов между сегментами, он позволяет увеличить пропускную способность загруженных сетей за счет изоляции трафика по отдельным сегментам. Особенно хорошо адаптивные мосты работают в сетях, которые могут быть физически разделены на два сегмента. При этом в каждом из сегментов следует сосредоточить компьютеры, которые чаще всего взаимодействуют друг с другом. Например, очень хорошо поместить в один сегмент сети сервер и рабочие станции, которые чаще всего с ним взаимодействуют. Итак, подведем итог.

*Адаптивный мост используется для объединения двух сегментов Ethernet в один протяженный участок сети. Это устройство ретранслирует фреймы из одного сегмента сети в другой. При этом адаптивный мост анализирует адрес отправителя фрейма и составляет два списка машин, которые подключены к разным сегментам сети. Кроме того, анализируется адрес получателя фрейма, и если оказывается, что он находится в одном списке с отправителем, то ретрансляция фрейма в другой сегмент не выполняется.*

С точки зрения семейства протоколов TCP/IP, применение мостов в сетях Ethernet является еще одной формой физического подключения компьютеров. При этом не следует забывать одну важную вещь.

*Поскольку мосты и повторители позволяют объединить несколько физических кабелей сети в один так, что эти соединения оказываются совершенно "прозрачными" для машин, подключенных к сети Ethernet, полученную в результате системы кабелей следует считать единственным физическим участком сети.*

Мосты промышленного производства — гораздо более сложные и интеллектуальные устройства, чем те, что описаны выше. При включении питания они автоматически выполняют поиск других мостов и строят таким образом топологию сети. Для принятия решения о ретрансляции фреймов в них используется специальный алгоритм *распределенного связующего дерева* (*spanning-tree algorithm*). В частности, мост выполняет ретрансляцию широковещательных фреймов так, чтобы в каждый сегмент доставлялась только одна их копия. Без этого алгоритма невозможно обойтись в том случае, когда сегменты сети Ethernet соединены в кольцо с помощью мостов, поскольку тогда бы выполнялась ретрансляция широковещательных пакетов в обе стороны одновременно, что привело бы к печальным последствиям.

## 2.5. Волоконно-оптические сети FDDI

Волоконно-оптические сети *FDDI* (*Fiber Distributed Data Interface*, или *распределенный интерфейс передачи данных по оптоволоконному кабелю*), — еще одна популярная технология передачи данных со скоростью 100 Мбит/с (т.е. с той же скоростью, которую обеспечивает быстрый Ethernet). Однако в отличие от Ethernet и других сетевых технологий, в которых для передачи электрических сигналов используются медные кабели, в FDDI используется оптоволоконный кабель, а данные кодируются с помощью импульсов света<sup>9</sup>.

Оптоволоконный кабель имеет два преимущества перед медным проводом. Во-первых, поскольку электромагнитные помехи никак не влияют на оптическую среду передачи данных, оптоволоконный кабель можно прокладывать в непосредственной близости от мощных электрических устройств. Во-вторых, поскольку в оптоволоконном кабеле в качестве носителя используется свет, плотность потока данных в нем может быть существенно выше, чем в медном кабеле. А это означает, что по оптоволоконному кабелю можно передавать существенно больший трафик, чем по медному.

На первый взгляд может показаться, что при изгиба оптоволокно легко повредить и поэтому создать сеть на его основе существенно труднее, чем на основе медного кабеля. Однако на практике оказывается, что оптоволоконный кабель удивительно гибкий. Стеклянная жила оптоволоконного кабеля имеет очень маленький диаметр и покрыта сверху пластмассовой оболочкой, которая и защищает ее от повреждения. Конечно, оптоволоконный кабель нельзя сгибать под углом 90° (коаксиальный кабель, кстати, тоже), но зато его можно проложить по дуге, радиус которой составляет порядка 1–2 см. Поэтому прокладка оптоволоконного кабеля не вызывает особых трудностей.

### 2.5.1. Отличительные особенности сетей FDDI

Сети FDDI построены на основе кольцевой 100-Мбитовой технологии общего доступа с передачей маркера и возможностью самовосстановления. Сеть FDDI является сетью *общего доступа*, поскольку к одному оптоволоконному кабелю подключается множество компьютеров, которые поочередно участвуют в процессе пересылки пакетов. Сеть FDDI имеет *кольцевую топологию*, поскольку все компьютеры в сети объединяются в одно большое кольцо. При этом каждый компьютер кольца связан оптоволоконным кабелем с двумя соседними компьютерами. Сеть FDDI построена по технологии *кольца с передачей маркера* (*token passing ring*), или просто *маркерного кольца* (*token ring*), которое используется для управления доступом к среде передачи данных. Суть метода управления доступом состоит в том, что при отсутствии в сети трафика по кольцу от компьютера к компьютеру передается специальный фрейм, называемый *маркером* (*token*). Прежде чем отправить данные, рабочая станция должна дождаться получения маркера, передать пакет данных, а затем отправить маркер соседней станции. Таким образом, циркулирующий по кольцу маркер гарантирует равноправие доступа к сети всех рабочих станций. Дело в том, что, прежде чем отправить очередной пакет данных, рабочая станция должна снова дождаться маркера, а это позволяет другим машинам отправить свои пакеты данных.

Однако, пожалуй, самая интересная особенность технологии FDDI заключается в ее возможности обнаруживать и устранять неполадки в сети. Сеть FDDI называется *самовосстанавливающейся* (*self-healing*), поскольку оборудование может автоматически реагировать на нештатную ситуацию.

<sup>9</sup> Кстати, существует сетевая технология, аналогичная FDDI, в которой для передачи данных используются медные провода. Она называется *CDDI*, или *Copper Distributed Data Interface* (*распределенный интерфейс передачи данных по медному кабелю*)

## 2.5.2. Топология с двумя встречными кольцами

Для автоматического восстановления работоспособности сети при отказах оборудования в технологии FDDI используется два независимых кольца, подключенных к каждому компьютеру (рис. 2.9).

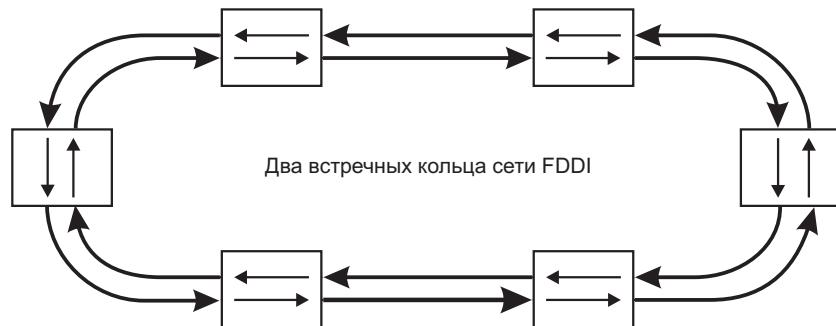


Рис. 2.9. Схема подключения шести компьютеров к сети FDDI. Стрелками показано направление передачи данных по оптоволоконным кабелям и подключененным к ним компьютерам

Кольца FDDI часто называют *встречными* (*counter rotating*), поскольку потоки данных протекают по каждому из колец в противоположных направлениях. Причина, по которой была выбрана схема подключения с двумя встречными кольцами, станет вам понятной чуть позже, когда мы будем рассматривать восстановление работоспособности сети FDDI после отказа оборудования или обрыва оптоволоконного кабеля.

В штатных условиях для работы сети FDDI оба кольца не нужны. Если нет отказов оборудования, сеть FDDI функционирует точно так же, как и любая сеть с передачей маркера. Плата сетевого интерфейса компьютера проверяет все фреймы, которые циркулируют по кольцу, и сравнивает адрес получателя каждого фрейма со своим физическим адресом. Если адреса совпадают, она передает полученный фрейм на обработку “своему” компьютеру и пересыпает фрейм далее по кольцу.

Прежде чем отправить фрейм, плата интерфейса ожидает получения маркера. Получив маркер, она временно приостанавливает пересылку чужих фреймов и отправляет свой. После отправки своего фрейма плата интерфейса отправляет маркер и возобновляет пересылку чужих фреймов. Получив маркер, плата интерфейса всегда отправляет только один свой фрейм, даже если в очереди на отправку их скопилось несколько.

Интересно рассмотреть, как поведет себя сеть FDDI в случае отказа одной из плат интерфейса или повреждения оптоволоконного кабеля. Как только плата сетевого интерфейса обнаруживает, что она не может отправить фрейм соседнему компьютеру, она пытается задействовать резервное кольцо для устранения проблемы. На рис. 2.10 показан поток данных, протекающий в кольцах FDDI, после отказа одной из плат интерфейса. Обратите внимание, что благодаря наличию двух колец соседние сетевые платы могут отключить отказавшее устройство от сети.

Таким образом, понятно, зачем в топологии FDDI предусмотрено второе кольцо и почему в нем данные текут в обратном направлении. Отказ сети может также произойти по причине повреждения или случайного разъединения оптоволоконного кабеля. Поэтому, если оптоволоконный кабель обоих колец проложен по одному физическому маршруту, очень велика вероятность повреждения обоих колец. При наличии двух встречных колец оборудование FDDI в случае отказа автоматически восстанавливает работоспособность сети, направляя потоки

**Печатай файл этой страницы отдельно**

Как и в других сетевых технологиях, в сети FDDI каждому компьютеру назначается уникальный физический адрес, и в каждом фрейме предусмотрено поле адреса получателя. Однако чтобы сделать технологию FDDI более гибкой и обеспечить стандартные средства для объединения двух колец, разработчики предусмотрели возможность использования в сети FDDI фреймов разных форматов. Например, длина поля адреса получателя может составлять либо 4, либо 12 символов. (Напомним, что *символ* — это единица измерения длины поля, равная четырем битам.) Кроме того, во фрейм также может быть включена информация о маршрутах. В поле для маршрутной информации отправитель может указать, что фрейм должен быть сначала послан в точку соединения колец, а затем — получателю, подключенному к этой точке.

Одно из преимуществ технологии FDDI — большой размер ее фрейма. Поскольку фрейм FDDI может содержать до 9000 символов длиной 4 бита, его максимальная длина может достигать 4500 октетов. При этом длина заголовка фрейма не превышает нескольких сотен октетов. Поэтому в одном фрейме может содержаться более 4 Кбайт данных. Использование фреймов значительного размера в приложениях, пересылающих большие потоки данных (как, например, при пересылке файлов), снижает непроизводительные расходы и, следовательно, повышает производительность сети в целом.

## 2.6. Асинхронный режим передачи (ATM)

*Асинхронный режим передачи (Asynchronous Transfer Mode, или ATM)*, находит применение в сетевой технологии, требующей установки соединения, которая может использоваться как в локальных, так и в глобальных сетях. Режим ATM создан специально для того, чтобы обеспечить максимально высокую скорость коммутации данных. Самое быстродействующее оборудование ATM может работать на гигабитовых скоростях<sup>10</sup>. Естественно, что работа на таких высоких скоростях требует применения сложного оборудования. Поэтому ATM значительно дороже других сетевых технологий.

Для достижения высоких скоростей передачи данных в сетях ATM используется специализированное программное и аппаратное обеспечение. Во-первых, сети ATM строятся на основе одного или нескольких высокоскоростных *коммутаторов (switches)*, к которым подключаются пользовательские компьютеры и другие коммутаторы ATM. Во-вторых, для подключения абонентов к сети ATM (включая компьютеры пользователей и другие коммутаторы) используется оптоволоконный кабель, поскольку оптоволоконный кабель обеспечивает большую скорость передачи данных, по сравнению с медным. Обычно скорость обмена данными между компьютерами пользователей и коммутатором ATM составляет 155 Мбит/с. В-третьих, на самом нижнем уровне в сетях ATM используются фреймы фиксированного размера, которые называются *ячейками (cells)*. Поскольку все ячейки имеют одинаковый размер, коммутатор ATM может их обрабатывать с очень высокой скоростью.

### 2.6.1. Размер ячейки ATM

Что удивительно, размер ячейки ATM составляет всего 53 октета. При этом длина заголовка составляет 5 октетов, за которым следуют 48 октетов данных. В следующих главах будет показано, что когда технология ATM используется

---

<sup>10</sup> Большинство современных компьютеров пока еще не могут генерировать или обрабатывать данные с такими скоростями. В сетях ATM гигабитовые скорости применяются для обработки потоков данных, поступающих от большого числа компьютеров.

для пересылки IP-трафика, размер ячейки, равный 53 октетам, оказывается мало подходящим. Поэтому в сетях ATM допускается применение пакетов больших размеров.

### 2.6.2. Сетевая технология, требующая установки соединения

Сети ATM отличаются от описанных выше сетей с коммутацией пакетов, поскольку в них используется процедура *установки соединения*. Прежде чем компьютер, подключенный к сети ATM, сможет отправлять ячейки, должно быть установлено соединение с получателем. Другими словами, компьютер должен выполнить ряд процедур с коммутатором и сообщить ему адрес получателя. Эти процедуры чем-то напоминают обычный телефонный звонок, когда набирается номер абонента<sup>11</sup>. Компьютер, который должен отправлять ячейки, сообщает коммутатору ATM адрес удаленного компьютера и ожидает, пока тот определит маршрут к получателю и установит с ним связь. Запрос на установку соединения может завершиться неудачей, если удаленный компьютер его отвергнет либо не ответит на него либо коммутаторы, находящиеся между отправителем и получателем, не смогут в данный момент времени проложить маршрут соединения.

После того как соединение с получателем успешно установлено, локальный коммутатор ATM назначает ему специальный идентификатор и передает его отправителю вместе с соответствующим сообщением. Отправитель использует в дальнейшем этот идентификатор для отправки или получения ячеек.

После завершения обмена данными отправитель снова посыпает запрос к коммутатору ATM на разрыв соединения. Получив запрос, коммутатор разрывает соединение между двумя компьютерами. Эта процедура аналогична тому, как после окончания разговора абонент вешает телефонную трубку. После разрыва соединения компьютеры не смогут обмениваться данными до тех пор, пока не будет установлено новое соединение. Следует отметить, что идентификаторы соединения могут использоваться повторно. Как только текущее соединение разрывается, коммутатор может назначить его идентификатор другому соединению.

## 2.7. Технологии создания глобальных сетей: ARPANET

Ниже будет показано, что технологии глобальных сетей оказали огромное влияние на системы адресации и маршрутизации, принятые в Internet. В заключительной части этой главы мы отобрали для описания несколько сетевых технологий, сыгравших заметную роль в истории развития Internet; они также используются в примерах, приведенных в последующих главах книги.

Одной из самых старых технологий построения глобальных сетей является ARPANET. Ее разработка финансировалась Управлением перспективного планирования научно-исследовательских работ (*Advanced Research Projects Agency*, или ARPA). Осенью 1968 года оно заключило контракт на разработку технологии ARPANET с тремя сотрудниками Кембриджского университета, шт. Массачусетс — Болтом (Bolt), Беранеком (Beranek) и Ньюманом (Newman). Первый участок сети ARPANET был запущен в опытную эксплуатацию в сентябре 1969 года.

Сеть ARPANET долгое время являлась тестовым полигоном для исследования сетей с коммутацией пакетов. Однако кроме исследовательских, ARPANET служила и чисто практическим целям. Ученые нескольких университетов, а также сотрудники некоторых военных и государственных исследовательских институтов регулярно ее использовали для обмена файлами и сообщениями электронной

---

<sup>11</sup> Поскольку технология ATM была разработана для пересылки как голосовых, так и обычных данных, существует четкая взаимосвязь между сетью ATM и телефонной сетью.

почты, а также для работы на удаленных компьютерах. В 1975 году управление сетью было выведено из-под контроля ARPA и поручено *управлению связи* Министерства обороны США (*Defense Communication Agency*, или *DCA*). Управление DCA сделало ARPANET частью собственной сети, которую назвало *Defense Data Network (DDN)*. Сеть DDN разрабатывалась в рамках программы объединения нескольких разрозненных сетей и создания на их основе части глобальной коммуникационной системы для Министерства обороны США.

В 1983 году Министерство обороны разделило ARPANET на две связанные сети. При этом за сетью ARPANET были сохранены ее исследовательские функции, а для военных целей была сформирована новая сеть, которую назвали *MILNET*. Однако MILNET использовалась исключительно для пересылки открытых данных, поскольку в то время не ставился вопрос о ее защите от несанкционированного доступа. И хотя была достигнута договоренность об обмене трафиком между ARPANET и MILNET при нормальных условиях, управление сетями было построено так, что можно было в любой момент их разъединить<sup>12</sup>. Поскольку и в ARPANET и в MILNET использовалось одинаковое сетевое оборудование, описанные ниже технические детали применимы к обеим этим сетям. Более того, эти сетевые технологии свободно продавались на рынке и были использованы некоторыми крупными корпорациями для создания собственных сетей с коммутацией пакетов.

Поскольку сеть ARPANET была давно запущена в работу и ежедневно использовалась большим количеством разработчиков, создававших Internet, она имела огромное влияние на их работу. Поэтому сложилось мнение об ARPANET, как об очень надежном глобальном магистральном канале, на основе которого может быть создана Internet. Влияние единого, центрального глобального магистрального канала было настолько сильным, что оно отразилось на разработке стандартов ряда протоколов Internet, которые мы рассмотрим чуть ниже. Все это препятствовало в дальнейшем расширению магистрального канала Internet за счет подключения дополнительных магистральных сетей.

Физически сеть ARPANET состояла приблизительно из 50 миникомпьютеров типа C30 и C300, выпущенных фирмой BBN Corporation. Они назывались *узлами коммутации пакетов*<sup>13</sup> (*Packet Switching Nodes*, или *PSN*) и были разбросаны по всей территории материковой части США и Западной Европы. Сеть MILNET состояла приблизительно из 160 PSN, причем 34 из них были расположены в Европе, а 18 — в Тихом Океане и в Азиатско-Тихоокеанском регионе. Устройства PSN располагались в местах, которые отвечали за работу сети. Как правило, в каждом таком месте был установлен один PSN, который был задействован только для коммутации пакетов. Его нельзя было использовать для решения вычислительных задач общего плана. Устройства PSN считались частью ARPANET и являлись собственностью *Сетевого операционного центра* (*Network Operations Center*, или *NOC*), который размещался в корпорации BBN в Кембридже, шт. Массачуссетс. Операционный центр управлял устройствами PSN всей ARPANET.

Для формирования сети устройства PSN соединялись друг с другом двухточечными (point-to-point) выделенными каналами связи, арендованными у национальных телефонных компаний. Например, с помощью выделенных линий соединялись между собой устройства PSN сети APRANET, находившиеся в университетах

<sup>12</sup> Пожалуй, один из самых известных случаев разъединения сетей произошел в ноябре 1988 года, когда Internet была атакована известной программой-червем Morrisa, которая с огромной скоростью стала размножать себя по узлам сети.

<sup>13</sup> Первоначально узлы коммутации пакетов назывались *интерфейсными процессорами сообщений* (*Interface Message Processors*, или *IMP*). В некоторых публикациях эти термины до сих пор используются как синонимы.

Пердью, Карнеги Меллон и Висконсин. Первоначально большинство выделенных линий связи, составлявших ARPANET, работали на скорости всего 56 Кбит/с. В 1968 году такая скорость считалась достаточно высокой для обеспечения работы магистрального канала, однако по нынешним меркам ее можно назвать крайне низкой. Напомним, что под скоростью работы сети мы понимаем количество прошедших по сети данных в единицу времени (т.е. пропускную способность), а не скорость доставки отдельных пакетов. По мере подключения новых компьютеров к ARPANET, мощность ее магистрального канала соответствующим образом увеличивалась, чтобы он мог справиться с текущей нагрузкой. Например, за последний год существования сети ARPANET, скорость многих каналов связи, протянувшихся по всей территории США, достигла 1 Мбит/с.

При разработке военных проектов во главу угла всегда ставилась надежность системы и отсутствие в ней узких мест. При создании ARPANET агентство ARPA решило не отходить от этих правил, поэтому оно потребовало, чтобы каждое устройство PSN соединялось с другими подобными устройствами с помощью как минимум двух выделенных каналов связи. При этом программное обеспечение должно быть настроено так, чтобы при возникновении нештатной ситуации система автоматически выбирала альтернативный маршрут. Поэтому сеть ARPANET оставалась в рабочем состоянии, даже если один из каналов передачи данных выходил из строя.

Кроме поддержки выделенных каналов связи, каждое устройство PSN сети ARPANET должно было иметь до 22 портов, к которым подключались компьютеры пользователей, или *хосты* (*hosts*). Первоначально каждый компьютер, имевший доступ к ARPANET, подключался напрямую к одному из портов устройства PSN. Как правило, подключение хоста к узлу коммутации пакетов осуществлялось с помощью специальной платы интерфейса, которая вставлялась в один из свободных разъемов шины ввода-вывода компьютера.

В оригинальном аппаратном обеспечении порта PSN использовался сложный протокол для передачи данных по сети ARPANET. Его называли 1822 — по номеру технического отчета, в котором он был описан. Протокол 1822 позволял хосту отправить по сети ARPANET пакет получателю, который идентифицировался адресом конкретного PSN и номером его порта. Сама процедура отправки пакета была достаточно сложной отчасти потому, что в протоколе 1822 оговаривался надежный и управляемый механизм доставки. Для предотвращения monopolизации сети конкретным хостом в протоколе 1822 оговаривалось максимальное количество пакетов, которое он может отослать за один раз. Чтобы гарантировать доставку каждого пакета получателю, посыпая пакет, отправитель, согласно протоколу 1822, должен был ожидать от устройства PSN сигнала готовности к следующему сообщению *RFNM* (*Ready For Next Message*). Таким образом, сигнал RFNM использовался для подтверждения доставки предыдущего пакета. В протоколе 1822 была также оговорена специальная схема резервирования буферов, согласно которой, прежде чем посыпать пакет устройству PSN, отправитель должен был зарезервировать для него буфер на узле коммутации пакетов.

И хотя здесь не были рассмотрены многие технические детали работы ARPANET, суть вам уже понятна — технология ARPANET в сущности была не более чем механизмом передачи. Когда компьютер, подключенный к одному из портов PSN, отправлял пакет другому порту, данные доставлялись точно в таком же виде, как они и были посланы. Другими словами, в процессе пересылки к данным не добавлялась никакая служебная информация. Поэтому в посланных по сети ARPANET пакетах не предусматривалось специальное поле, в котором бы указывался их тип. Таким образом, в отличие от других сетевых технологий, сеть ARPANET нельзя использовать для доставки автоматически опознаваемых пакетов. Итак, можно подвести некоторые итоги.

*В сетях, подобных ARPANET или ATM, не предусматривается использование автоматически опознаваемых фреймов. Согласование форматов и содержимого пересылаемых по таким сетям пакетов всецело возлагается на подключенные к ним компьютеры.*

К сожалению, протокол 1822 так и не стал промышленным стандартом. Причина заключалась в том, что платы интерфейса под этот протокол выпускались всего несколькими фирмами. Поэтому по мере роста ARPANET становилось все сложнее подключать к ней новые компьютеры. Для решения возникшей проблемы управление ARPA со временем пересмотрело интерфейс PSN и стало использовать в нем международный стандарт X.25<sup>14</sup>. В реализации первой версии узла коммутации пакетов на основе X.25 использовался только протокол передачи данных этого стандарта, который назывался HDLC/LAPB. Однако в более поздних версиях PSN при подключении к этим устройствам можно было использовать все протоколы стандарта X.25. Таким образом, со временем сеть ARPANET стала похожа на сеть X.25.

Однако это сходство было чисто внешним, поскольку внутри ARPANET использовался собственный набор протоколов, который скрыт от пользователя за ширмой стандарта X.25. Например, существовал специальный протокол, позволяющий одному устройству PSN запросить состояние другого устройства; еще один протокол был предназначен для обмена пакетами между устройствами PSN; кроме того, был предусмотрен протокол для обмена между устройствами PSN информацией об их состоянии и оптимальном маршруте следования пакетов.

Поскольку первоначально ARPANET разрабатывалась как единая независимая сеть, предназначенная для использования в научно-исследовательских целях, при создании набора протоколов и схемы адресации для нее никто всерьез не задумывался о возможном в будущем расширении. Однако уже к середине 1970-х годов стало ясно, что наземные сети не смогут решить всех проблем обмена информацией, поэтому ARPA стало финансировать исследования в области спутниковой связи и передачи пакетов данных по радиоканалу. Накопленный в результате исследований опыт работы с различными сетевыми технологиями и был положен в основу принципов межсетевого взаимодействия.

### 2.7.1. Система адресации ARPANET

Подробности реализации механизма адресации в ARPANET сегодня вряд ли кого-то заинтересуют. Однако на его примере можно показать альтернативный метод формирования физических адресов в глобальных сетях. В отличие от адресных схем с *прямой адресацией* (*flat address*), использующихся в локальных сетях, в системе адресации глобальных сетей должна быть предусмотрена возможность включения в сам адрес информации, которая бы способствовала эффективной маршрутизации пакетов во время доставки их получателю. В технологии ARPANET каждому узлу коммутации пакетов назначается уникальный адрес, представляющий собой целое число. Обозначим его через  $P$ . Каждый порт узла коммутации пронумеровывается, и ему назначается целое число из диапазона  $[0, N-1]$ , где  $N$  — общее количество портов в устройстве PSN. Таким образом, теоретически адрес абонента в сети ARPANET должен состоять из двух небольших целых чисел ( $P, N$ ). На практике оборудование оперировало с одним

<sup>14</sup> Стандарт X.25 был принят в свое время *Международным консультативным комитетом по телеграфии и телефонии* (International Consultative Committee for Telegraphy and Telephony, или CCITT), который впоследствии был реорганизован в *Телекоммуникационный сектор* (*Telecommunication Section*) при *Международном телекоммуникационном союзе* (*International Telecommunication Union*, или ITU).

большим целым числом, часть битов которого в двоичном представлении отводилось под номер порта узла коммутации пакетов ( $N$ ), а оставшаяся часть — под адрес устройства PSN ( $P$ ).

## 2.8. Сетевая программа Национального научного фонда

Понимая, что в ближайшем будущем очень важным моментом в научных исследованиях будет процесс обмена данными, *Национальный научный фонд (NSF)* в 1987 году основал *отделение сетевых и коммуникационных исследований и инфраструктуры (Division of Network and Communications Research and Infrastructure)*. В его задачи входило обеспечение современными сетевыми коммуникационными средствами ученых и инженеров США. И хотя отделение фонда NSF финансировало основные исследовательские программы в области сетевых коммуникаций, сферой его основных интересов было расширение Internet.

В предложенной фондом NSF программе расширения Internet вводилось понятие трехуровневой иерархической системы. На верхнем уровне иерархии находился глобальный магистральный канал США. Средний уровень составляли ряд региональных сетей, каждая из которых развертывалась на небольшой географической территории. К нижнему уровню иерархии относились так называемые сети доступа, охватывающие небольшие организации и университетские городки. В модели NSF к национальному магистральному каналу подключаются сети среднего уровня, а к ним — локальные сети доступа. Рабочее место каждого исследователя подключается к локальной сети доступа. С помощью единственного сетевого подключения исследователь может обмениваться информацией как со своими сослуживцами, так и с коллегами по работе, находящимися за много километров от него. В первом случае данные передаются по локальной сети от одного компьютера к другому. Во втором случае данные по локальной сети поступают на ближайший узел маршрутизации и от него попадают в сеть среднего уровня. При необходимости выполняется их маршрутизация в магистральный канал.

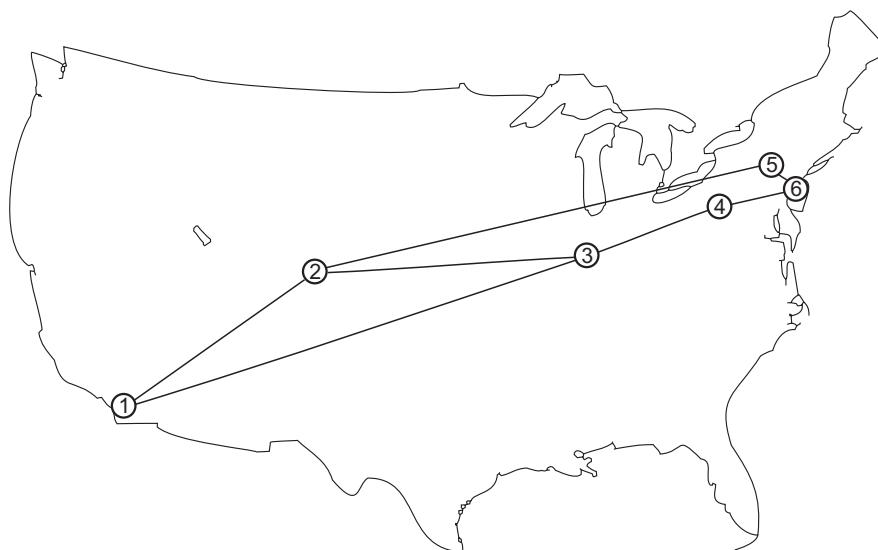
### 2.8.1. Первый магистральный канал NSFNET

В предыдущем разделе была рассмотрена сетевая программа, финансируемая фондом NSF. Теперь пришло время поговорить о магистральном канале NSFNET, поскольку он имеет интересную историю и для его создания была применена очень интересная технология. Развертывание магистрального канала выполнялось в четыре этапа. В то время как сеть ARPANET исчерпала свои возможности, пропускная способность и протяженность канала NSFNET все время увеличивались. В конечном итоге он стал основным магистральным каналом зарождающейся Internet. Первый вариант сетевой магистрали был запущен в работу в очень сжатые сроки и рассматривался как временная мера, позволяющая ученым получить доступ к суперкомпьютерам Национального научного фонда. Поэтому первый магистральный канал был сформирован из шести микрокомпьютеров LSI-11 фирмы Digital Equipment Corporation, размещенных в существующих суперкомпьютерных центрах NSF. Сетевая магистраль протянулась по континентальной части США от Принстона (шт. Нью-Джерси) до Сан-Диего (шт. Калифорния), как показано на рис. 2.11. При этом использовались выделенные каналы связи с пропускной способностью 56 Кбит/с.

На установленном в каждом из центров микрокомпьютере LSI-11 была запущена специальная программа, разработанная Дэйвом Миллсом (Dave Mills), которую любовно прозвали “летящий мячик” (*fuzz-ball*<sup>15</sup>). Для доступа к компьютерам, установленным в суперкомпьютерном центре, использовалась обычная

<sup>15</sup> Происхождение названия “fuzz-ball” выяснить так и не удалось.

сеть Ethernet. Для связи с миникомпьютерами, установленными в других центрах, использовались традиционные протоколы канального уровня, которые обслуживали выделенные каналы связи с последовательной передачей данных. Программа микрокомпьютера обрабатывала таблицы, содержащие адреса возможных получателей пакетов данного центра, и направляла полученные из вне пакеты соответствующему компьютеру локальной сети.



*Рис. 2.11. Схема первого магистрального канала NSFNET, объединившего шесть суперкомпьютерных центров Национального научного фонда США: (1) Сан-Диего (шт. Калифорния); (2) Боулдер-Сити (шт. Колорадо); (3) Чемпейн (шт. Иллинойс); (4) Питтсбург (шт. Пенсильвания); (5) Айтака, (шт. Нью-Йорк); (6) Принстон (шт. Нью-Джерси)*

Основной узел, через который осуществлялась связь первого магистрального канала NSFNET с остальной частью Internet, находился в университете Карнеги Меллона, поскольку там же располагалось устройство PSN сети ARPANET. Поэтому, когда пользователю, подключенному к сети NSFNET, нужно было обращаться к компьютеру сети ARPANET, пакеты посыпались микрокомпьютеру LSI-11, находящемуся в университете Карнеги Меллона, откуда они попадали по локальной сети Ethernet на устройство PSN сети ARPANET. Кроме того, программа “летящий мячик” распознавала пакеты, посланные по локальной сети Ethernet абонентам NSFNET, расположенным в других суперкомпьютерных центрах, и отправляла пакеты по магистральной сети на соответствующий узел.

### **2.8.2. Второй магистральный канал NSFNET (1988-1989)**

После запуска первого магистрального канала NSFNET возможностями компьютерной связи были потрясены тысячи людей. Однако его пропускная способность и скорость коммутации пакетов оставались на очень низком уровне, что не позволяло обеспечить всех желающих соответствующим уровнем услуг. Уже через несколько месяцев после начала работы магистральный канал оказался перегруженным, а его создатели стали искать быстрые пути решения возникших неотложных проблем. В это время NSF напряженно работало над проектом второго магистрального канала.

В 1987 году фонд NSF объявил, что ждет предложений от заинтересованных групп, которые бы хотели создать новый, более скоростной магистральный канал. Предложения были приняты к рассмотрению в августе 1987 года и получили должную оценку в конце года; 24 ноября 1987 года фонд NSF объявил, что выбрано предложение,несенное группой, в состав которой входили следующие фирмы: MERIT Inc., IBM Corporation и MCI Incorporated. Из этих трех фирм только первая нуждается в представлении. Она поддерживала работоспособность глобальной сети шт. Мичиган, которая была создана местным университетом города Эн Або (Ann Arbor). Группа предложила создать второй магистральный канал, разместить операционный центр и центр управления сетью в Эн Або и ввести систему в строй к будущему лету. Поскольку фонд NSF выделил средства на создание нескольких новых сетей среднего уровня, то первоначальный проект второго магистрального канала пришлось пересмотреть и включить в него дополнительные сетевые центры. Каждый такой дополнительный центр должен был обеспечивать подключение к магистральному каналу одной из вновь созданных сетей NSF среднего уровня.

Логично предположить, что разделение труда между тремя фирмами должно было складываться следующим образом. Фирма MERIT отвечала за разработку проекта и развертывание магистрального канала, а также поддерживала работоспособность сетевого операционного центра. Фирма IBM обеспечивала проект техникой и людскими ресурсами; фирма MERIT разрабатывала, настраивала и тестировала оборудование и программное обеспечение. Фирма MCI, национальный поставщик услуг связи, должна была обеспечить проект коммуникационными ресурсами, выделив часть полосы оптоволоконного канала своей телефонной сети. Конечно же, на практике все три фирмы тесно сотрудничали между собой при выполнении исследовательских проектов, а представители фирм IBM и MCI принимали участие в руководстве проектом.

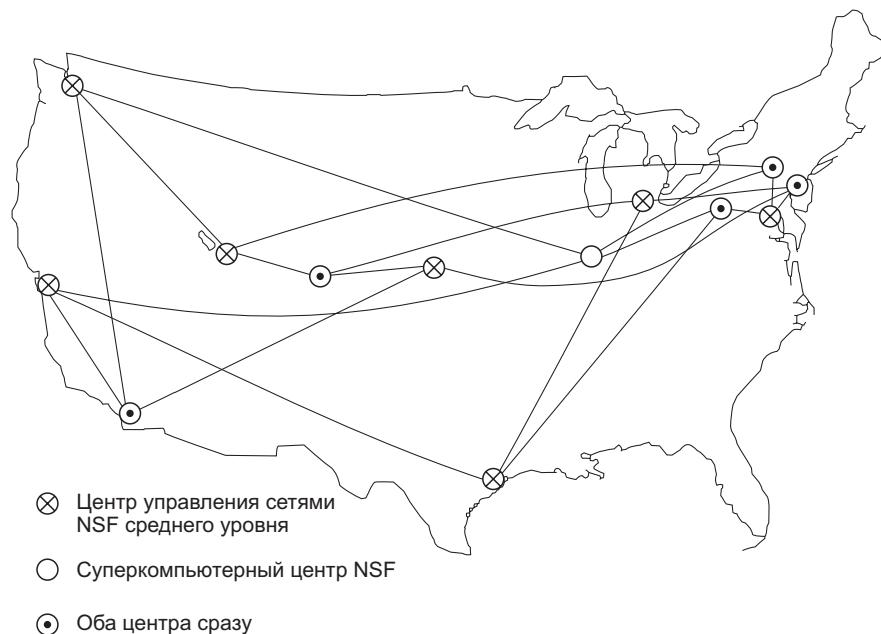


Рис. 2.12. Схема второго магистрального канала NSFNET после запуска в эксплуатацию летом 1988 (до лета 1989)

К середине лета 1988 года все необходимое оборудование было развернуто, и второй магистральный канал сети NSFNET был запущен в эксплуатацию. Вскоре после этого первый магистральный канал был отключен от сети и демонтирован. На рис. 2.12 показана структурная схема второго магистрального канала NSFNET после введения его в эксплуатацию летом 1988 года.

Технология, выбранная для создания второго магистрального канала NSFNET, была довольно необычной. По существу это была глобальная сеть, основу которой составляли маршрутизаторы пакетов, соединенные между собой каналами связи. Как и в первом магистральном канале, в каждом сетевом центре был установлен маршрутизатор пакетов, который подключался как к локальной сети Ethernet центра, так и к каналу связи с другим центром.

### 2.8.3. Магистральный канал NSFNET 1989-1990 гг.

Наблюдая на протяжении года за потоками данных, циркулирующими во втором магистральном канале NSFNET, операционный центр решил изменить конфигурацию сети. Были удалены ненужные связи между центрами и добавлены несколько дополнительных связей. Кроме того, была расширена пропускная способность магистральной сети до 1,544 Мбит/с за счет использования каналов DS1. На рис. 2.13 показана пересмотренная схема сети, обеспечивающая резервные связи между всеми центрами.

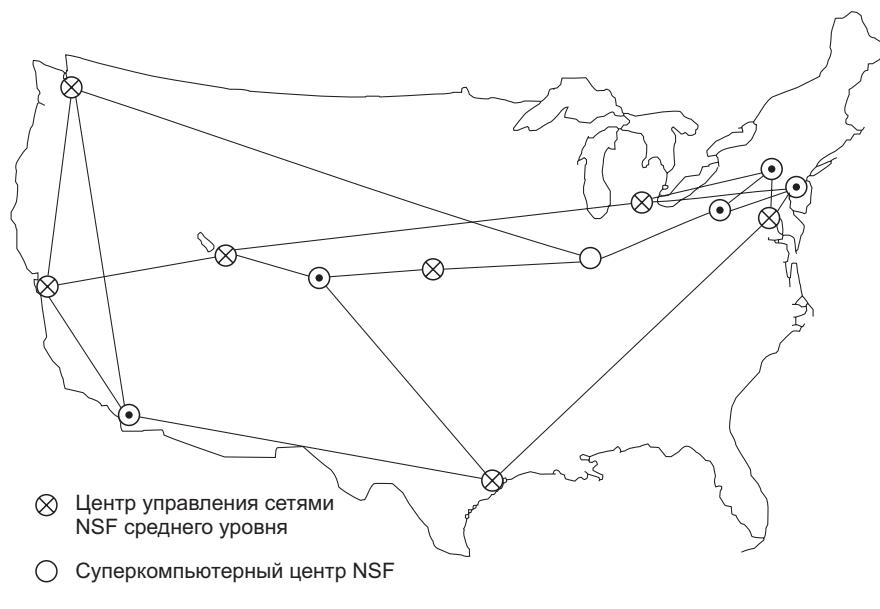


Рис. 2.13. Схема модернизированного магистрального канала NSFNET (лето 1989 — 1990 гг.)

## 2.9. Сеть ANSNET

К 1991 году фонд NSF и другие государственные учреждения США поняли, что масштабы Internet вышли далеко за отведенные ей на этапе разработки рамки университетской и научной сети. К Internet стало подключаться множество

организаций, разбросанных по всему Земному шару. В результате быстро выросло количество коммерческих пользователей глобальной сети, которые не имели никакого отношения к научным и исследовательским кругам. Трафик в магистральном канале NSFNET вырос почти до миллиарда пакетов в день, и его пропускной способности в 1,5 Мбит/с на отдельных участках стало уже не хватать. Было очевидно, что нужен другой, более высокоскоростной магистральный канал. Поэтому правительство США начало проводить политику приватизации и коммерческого использования Internet. Фонд NSF принял решение передать магистральную сеть на попечение закрытой акционерной компании и оплачивать доступ к ней для государственных научных и исследовательских организаций.

В ответ на новую государственную политику, в декабре 1991 года фирмы IBM, MERIT и MCI учредили некоммерческую организацию и назвали ее *Advanced Networks and Services (ANS)*. ANS предложила создать новый высокоскоростной магистральный канал Internet. Однако в отличие от всех предыдущих магистральных каналов и сетей его формирующих, которые были собственностью государства, владельцем новой сетевой магистрали Internet должна была стать организация ANS. К 1993 году ANS ввела в строй новый магистральный канал, который заменил NSFNET. Его назвали *ANSNET*. Он состоял из каналов передачи данных с пропускной способностью 45 Мбит/с<sup>16</sup>. Таким образом, мощность нового магистрального канала Internet была расширена примерно в 30 раз по сравнению с предыдущей сетевой магистралью NSFNET. На рис. 2.14 показаны основные магистрали ANSNET, а также ряд новых сетевых центров, подключенных к ней в 1994 году. Обратите внимание, что к каждой из точек присутствия может быть подключено несколько сетевых центров, обслуживающих конечных пользователей.

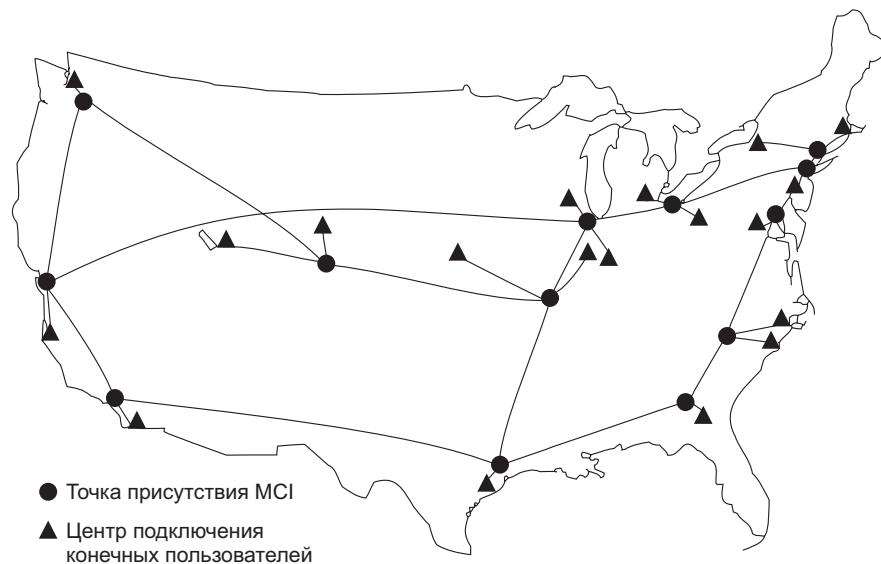


Рис. 2.14. Схема сети ANSNET, которая с 1993 года являлась магистральным каналом Internet в США. Каждый из каналов передачи данных работал на скорости 45 Мбит/с

<sup>16</sup> В телекоммуникации для обозначения каналов передачи данных, работающих на скорости 45 Мбит/с используется специальное обозначение DS3. Его часто путают с другим термином T3, который означает тип кодирования данных, передаваемых по каналу DS3.

## **2.10. Сверхвысокоскоростной магистральный канал (vBNS)**

В 1995 году фонд NSF заключил контракт с фирмой MCI на создание магистрального канала Internet с пропускной способностью 155 Мбит/с (канал OC3), который должен был прийти на смену ANSNET. Его называли *сверхвысокоскоростным магистральным каналом сетевых служб* (*very high speed Backbone Network Service*, или *vBNS*). Реализация подобного проекта потребовала применения новых высокоскоростных маршрутизаторов пакетов, что позволило существенно увеличить пропускную способность каналов передачи данных.

### **2.10.1. Коммерческие магистральные каналы Internet**

С 1995 года Internet начинает широко использоваться в коммерческих целях. При этом доля государственного финансирования новых сетевых проектов стала постепенно снижаться. И хотя сеть vBNS по прежнему продолжает функционировать, в настоящее время она используется, преимущественно в исследовательских целях. Вместо нее несколько коммерческих компаний на собственные средства создали ряд больших магистральных каналов, по которым проходит основной поток данных Internet. Например, крупные национальные телекоммуникационные корпорации типа *AT&T* и *MCI* создали собственные магистральные сети с высокой пропускной способностью, по которым проходит поток данных Internet, поступающий от их клиентов. Как будет описано ниже, коммерческие магистральные каналы соединены между собой на условиях *партнерских соглашений* (*peering arrangements*). Это позволяет клиентам разных коммерческих компаний напрямую обмениваться между собой пакетами данных.

## **2.11. Другие технологии, использующие протокол TCP/IP**

Одно из основных преимуществ протокола TCP/IP заключается в том, что он может работать на основе самого разнообразного сетевого оборудования. Выше были рассмотрены самые распространенные технологии и принципы построения локальных и глобальных сетей. В этом разделе кратко будут описаны другие технологии, которые подтверждают, что

*секрет популярности семейства протоколов TCP/IP в большей степени заключается в их способности работать практически с любой аппаратурной технологией передачи данных.*

### **2.11.1. Технология X25NET и туннелирование**

В 1980 году фонд NSF учредил организацию *Computer Science NETwork (CSNET)*, в задачи которой входило предоставление услуг Internet промышленным предприятиям и небольшим учебным заведениям. Для подключения клиентов к сети CSNET с выходом в Internet использовалось несколько технологий. Одна из них называлась *X25NET*. Эта технология первоначально было разработана в университете Пердью и предназначалась для передачи данных по открытым *сетям общего пользования* (*Public Data Networks*, или *PDN*) с помощью семейства протоколов TCP/IP. Причины создания подобной технологии были чисто экономические. В те времена стоимость прямого выделенного канала в Internet была очень большой. Поэтому национальные телекоммуникационные компании стали предлагать услуги по коммутации пакетов. Таким образом, технология X25NET позволила небольшим организациям, имеющим доступ к сетям с коммутацией пакетов общего пользования, обмениваться данными с Internet.

Те из читателей, кто знаком с основами работы сетей с коммутацией пакетов общего пользования, могут удивиться применению технологии X25NET, поскольку известно, что в этих сетях используется исключительно семейство протоколов CCITT X.25, точно так же, как в Internet — семейство протоколов TCP/IP. В отличие от большинства технологий с коммутацией пакетов, в протоколах X.25 используется принцип установки соединения, наподобие технологии ATM. Они были разработаны для предоставления отдельным прикладным программам услуг по установке логических соединений. Таким образом, идея использования технологии ATM для переноса трафика TCP/IP была опробована много лет тому назад на сетях с коммутацией пакетов общего пользования и семействе протоколов X.25.

Выше уже отмечалось, что для переноса трафика Internet может использоваться несколько низкоуровневых технологий. Одна из них — X25NET — позволяет продемонстрировать, как для этой цели можно использовать средства высокого уровня. Применимый при этом метод доступа иногда называют *туннелированием* (*tunneling*). Он означает, что с точки зрения семейства протоколов TCP/IP, любая сложная сетевая система с собственным набором протоколов может рассматриваться как средство доставки, по аналогии с сетевым оборудованием. Чтобы отправить трафик TCP/IP по *туннелю* X.25, компьютер сначала должен установить соединение X.25, а затем переслать пакеты TCP/IP так же, как обычные данные. В результате система X.25 по открытому соединению доставляет пакеты TCP/IP на другой узел коммутации X.25, где выполняется их обработка и при необходимости пересылка на третий узел коммутации X.25, и т.д. до тех пор, пока пакеты не достигнут получателя. Поскольку при туннелировании IP-пакеты считаются обычными данными, которые должны быть без искажения доставлены по назначению, существование автоматически опознаваемых фреймов невозможно по определению. Поэтому процесс туннелирования возможен только в случае, если обоим участникам открытого соединения X.25 заранее известно, что они будут обмениваться IP-пакетами (т.е. при этом будут согласованы форматы передачи данных и способы декодирования типа каждого пакета).

Ориентированный на установку соединения интерфейс наделяет протокол X.25 специфическими свойствами. В отличие от сетей, не требующих установки соединения, в системах с установкой соединения используется абстрактное понятие *виртуального канала* (*virtual circuit*, или *VC*). Перед отправкой данных коммутаторы в сети должны установить виртуальный канал (или проложить маршрут) между отправителем и получателем. Выше уже шла речь о том, что протоколы Internet оптимизированы для работы в системах доставки пакетов, не требующих установки соединения. А это означает, что для передачи IP-трафика по сетям, требующим установки соединения, необходимы дополнительные меры.

Теоретически для создания туннеля в сети достаточно установить одно соединение. Т.е. после того, как два компьютера создадут между собой виртуальный канал, они могут обмениваться трафиком TCP/IP. Однако на практике использование единственного соединения приводит к неэффективной работе систем передачи данных. Причина заключается во внутренней структуре протоколов, которые используются в этих системах. Например, фактором, сдерживающим производительность в семействе протоколов X.25, является ограничение на количество пакетов, которые могут быть посланы по установленному каналу связи до получения сигнала подтверждения приема. Поэтому для повышения производительности сети X.25 при передаче IP-трафика нужно одновременно открывать несколько соединений. Этот фактор учли разработчики технологии X25NET. При подключении к узлу получателя, отправитель открывает несколько виртуальных каналов и распределяет трафик между ними. После получения пакетов,

пришедших по всем виртуальным каналам, выполняется их объединение и восстановление данных.

Для туннелирования по сети высокого уровня (наподобие X.25) требуется установить соответствие адресов, используемых в Internet и в этой сети. В качестве примера рассмотрим систему адресации, применяемую в сетях X.25. Она описывается в специальном стандарте, который называется *X.121*. Физический адрес состоит из 14 цифр, 10 из которых назначаются поставщиком сетевых услуг стандарта X.25. Так, один из крупных поставщиков сетевых услуг, назначает первые 10 цифр в соответствии с телефонным кодом региона, в котором расположены сетевые узлы X.25. Такая система адресации становится понятна, если вспомнить, что ее разрабатывала та же организация, которая создавала международные стандарты на телефонию. Однако не существует абсолютно никакой математической зависимости между тем, что описанной системой адресации X.25 и адресами хостов, используемыми в протоколе TCP/IP. Поэтому компьютер, выполняющий туннелирование трафика TCP/IP по сети X.25, должен вести таблицу соответствия адресов Internet и X.25. Более детально о проблеме соответствия адресов речь пойдет в главе 5, “Преобразование IP-адресов в физические адреса (ARP)”. Там будут описаны и другие методы установки соответствия адресов, кроме использования статических таблиц. В главе 18, “Протокол TCP/IP в сетях ATM”, показано, что подобная проблема возникает также в сетях ATM, однако для ее решения используется совершенно другой подход.

Поскольку сети общего пользования X.25 функционируют независимо от Internet, на первый план выходит организация точки взаимодействия между этими двумя сетями. В сетях ARPA и CSNET для этой цели используются выделенные компьютеры, которые обеспечивают обмен пакетами между сетями X.25 и ARPANET. Самое первое из таких устройств называлось *VAN-шлюз*. Оно поддерживало спецификацию X.25 и могло пересыпать каждую получаемую дейтаграмму по установленному виртуальному каналу ее получателю.

Важность технологии X25NET заключается в том, что на ее примере была продемонстрирована гибкость семейства протоколов TCP/IP и возможность их адаптации к любому способу транспортировки данных. В частности, было показано, что туннелирование позволяет применить для организации межсетевого взаимодействия широкий спектр сложных сетевых технологий.

### 2.11.2. Сети с двухточечным подключением

Выше уже шла речь о том, что любая глобальная сеть состоит из ряда выделенных коммутаторов пакетов, которые соединяются друг с другом каналами передачи данных, арендованными у телефонной компании. Первоначально телефонные компании создавали подобные каналы связи в расчете на то, что по ним будут передаваться оцифрованные голосовые данные. И только со временем их использование для передачи сетевого трафика вышло на первый план. Поэтому скорости передачи данных в этих системах не кратны  $10^k$ , где  $k$  — целое число. Они выражаются в единицах, кратных примерно 64 Кбит/с, или пропускной способности одного голосового канала связи. Голосовой аналоговый сигнал оцифровывается с частотой 8000 раз в секунду, при этом ширина выборки составляет 8 бит. Для кодирования оцифрованных голосовых выборок используется метод *импульсно-кодовой модуляции* (*Pulse Code Modulation*, или *PCM*). В табл. 2.2 приведены часто используемые скорости передачи данных в цифровых телефонных каналах Северной Америки и Европы. Эти же скорости используются для передачи сетевого трафика по выделенным каналам связи, арендованным у телефонных компаний. Обратите внимание, что пропускная способность каналов связи кратна числу голосовых каналов связи.

---

**Таблица 2.2. Скорости передачи данных в цифровой телефонии**

---

<i>Название стандарта</i>	<i>Скорость передачи данных (Мбит/с)</i>	<i>Число голосовых каналов</i>	<i>Где используется</i>
-	0,064	1	
T1	1,544	24	Северная Америка
T2	6,312	96	Северная Америка
T3	44,736	672	Северная Америка
E1	2,048	30	Европа
E2	8,448	120	Европа
E3	34,368	480	Европа

---

Кроме перечисленных в табл. 2.2, существуют каналы связи с более высокой пропускной способностью. Дополнительно к стандартам высокоскоростной передачи данных по медным проводам были разработаны стандарты передачи данных по оптоволоконным линиям связи с той же пропускной способностью. В табл. 2.3 приведены несколько примеров высокоскоростных стандартов передачи данных. Естественно, что стоимость аренды таких каналов существенно дороже, чем каналов с более низкой пропускной способностью. Для передачи данных с высокими скоростями на большие расстояния используется оптоволоконный кабель.

---

**Таблица 2.3. Примеры стандартов высокоскоростной передачи данных**

---

<i>Название стандарта</i>	<i>Аналогичный стандарт для оптоволоконной линии</i>	<i>Скорость передачи данных (Мбит/с)</i>	<i>Число голосовых каналов</i>
STS-1	OC-1	51,840	810
STS-3	OC-3	155,520	2430
STS-12	OC-12	622,080	9720
STS-24	OC-24	1244,160	19440
STS-48	OC-48	2488,320	38880

---

С точки зрения протокола TCP/IP любая система передачи данных, в которую входят только два компьютера называется *сетью с двухточечным подключением* (*point-to-point network*). Таким образом, типичным примером такой сети являются два компьютера, соединенные между собой выделенным каналом связи. Естественно, что такую систему можно называть сетью лишь с большой натяжкой. Однако для сохранения логики изложения мы будем рассматривать любое соединение между двумя компьютерами как сеть. А теперь остается лишь отметить, что сеть с двухточечным подключением имеет существенное отличие от обычной сети. Поскольку соединение существует только между двумя компьютерами, в такой сети не используется адресация на уровне сетевого оборудования. Поэтому ниже, при обсуждении привязки IP-адресов, мы будем делать исключение для сетей с двухточечным подключением.

### **2.11.3. Работа с сетью по коммутируемым телефонным каналам**

Еще одной особенностью семейства протоколов TCP/IP является возможность их использования для работы с сетью по коммутируемым телефонным каналам (т.е. по обычным телефонным проводам). Впервые этот вид услуг был предоставлен клиентам сети CSNET. Абоненты CSNET, которые пользовались Internet лишь время от времени, не могли себе позволить по финансовым соображениям подключиться к глобальной сети по выделенному каналу связи. Для таких клиентов в CSNET и был предусмотрен доступ к сети по коммутируемым телефонным каналам. Система работала следующим образом. Как только возникала необходимость доступа к сети, клиентское программное обеспечение с помощью модема, подключенного к обычной телефонной линии, устанавливало соединение с концентратором сети CSNET. В ответ на телефонный звонок модем, подключенный к концентратору, соединялся с клиентским модемом. В результате чего образовывался временный канал передачи данных. После прохождения стандартной процедуры опознавания абонента, концентратор начинал пересыпать пакеты данных по этому каналу между абонентом и Internet. Поскольку на набор номера модемом клиента и установку соединения с модемом концентратора уходило некоторое время, возникала задержка между получением запроса и отправкой первого пакета по коммутируемой сети. Хотя для автоматизированных систем наподобие электронной почты подобная задержка не имела особого значения.

Доступ к Internet через коммутируемое соединение — еще один пример сети с двухточечным соединением. С точки зрения семейства протоколов TCP/IP установка коммутируемого соединения эквивалентна прокладыванию кабеля. Как только модем на другом конце провода ответит на поступивший телефонный звонок и установит соединение с вызывающим модемом, между двумя компьютерами, к которым подключены эти модемы, устанавливается прямое соединение. Такое соединение может существовать столько времени, сколько нужно.

### **2.11.4. Сетевые технологии с передачей маркера**

Технология с передачей маркера (token ring) используется не только в сетях FDDI. Готовые решения на основе этой технологии продаются на рынке уже лет двадцать. Например, популярная технология создания локальных сетей с передачей маркера была создана фирмой IBM. Ранние версии сетей от IBM работали на скорости 4 Мбит/с. В более поздних версиях скорость передачи данных была увеличена до 16 Мбит/с. Подобно другим сетям с передачей маркера, в сети от IBM все компьютеры подключаются к одному кольцу. Прежде чем передать пакет, сетевая плата должна дождаться получения маркера, а после отправки пакета — сразу же передать маркер далее по кольцу.

В старой технологии сети с передачей маркера, разработанной фирмой Proteon Corporation, использовалась оригинальная система адресации. В главе 5, “Преобразование IP-адресов в физические адреса (ARP)”, на ее основе будет продемонстрирован один из возможных способов обработки физических адресов в семействе протоколов TCP/IP. Отличие этой технологии от других сетевых технологий заключается в том, что при создании сети пользователь может самостоятельно назначить физический адрес каждой сетевой плате. Подобные сети называются *proNET*. В отличие от сетей Ethernet, в которых уникальный адрес назначается каждой плате интерфейса при изготовлении, установка аппаратного адреса в сетевых адаптерах *proNET* выполняется с помощью восьми специально предусмотренных переключателей. Перед помещением платы в компьютер пользователь должен с помощью этих переключателей назначить ей уникальный адрес. Поскольку с помощью восьми переключателей можно установить в двоичной форме адрес, значение которого выбирается из диапазона чисел от 0 до 255

включительно, максимальное количество компьютеров в одной сети proNET не может превышать 254 (нулевой адрес не используется, а адрес 255 применяется для широковещательной передачи). При создании сети proNET сетевой администратор должен проконтролировать уникальность физических адресов каждой сетевой платы. Обычно всем платам назначаются последовательные адреса, начиная с 1.

Технология, позволяющая пользователю назначать аппаратные адреса сетевым платам, имеет как преимущества, так и недостатки. Одной из основных проблем является человеческий фактор. В самом деле, сетевой администратор может по ошибке назначить один и тот же адрес нескольким платам. К преимуществам данной технологии можно отнести удобство сопровождения сети. Даже если одна из сетевых плат выйдет из строя, ее можно легко заменить и сохранить при этом старый физический адрес.

### 2.11.5. Беспроводные сетевые технологии

Один из самых интересных экспериментов ARPA в области коммутации пакетов завершился созданием технологии *пакетной радиосвязи* (*packet radio*), в которой для передачи пакетов данных используются радиосигналы. Эта технология стала широко применяться для военных целей, поскольку одним из ее основных преимуществ является мобильность станций передачи данных. Для поиска других станций, установки с ними двухточечного канала связи и передачи пакетов данных по этому каналу используется специальное оборудование и программное обеспечение. Поскольку станции передачи данных могут изменять свое географическое положение и выходить за пределы области радиовидимости, система пакетной радиосвязи должна постоянно отслеживать возможность радиосвязи с ними и автоматически изменять соответствующим образом маршрутизацию пакетов при изменении топологии радиосети. Для демонстрации возможности обмена данными по протоколу TCP/IP между удаленной станцией и Internet была создана и запущена в эксплуатацию экспериментальная сеть пакетной радиосвязи.

В последние годы на рынке стало появляться оборудование для создания беспроводных локальных сетей. Для передачи данных между компьютерами в пределах одного здания по беспроводным локальным сетям применяется несколько методов, таких как установка очередности передачи или скачкообразное изменение несущей частоты. Передающее оборудование для беспроводных сетей имеет небольшие габариты и вес. Его можно легко подключить к портативному компьютеру, создав по-настоящему мобильную систему, которая тем не менее все время остается подключенной к локальной сети предприятия.

В современных коммуникационных системах используется технология беспроводной широкополосной передачи данных, которая первоначально разрабатывалась как альтернатива кабельному телевидению. Речь идет о *многоканальной многоочечной распределенной системе* (*Multichannel Multipoint Distribution System*, или *MMDS*). Скорость передачи данных в такой системе соизмерима с той, что обеспечивает популярная технология *DSL* (*Digital Subscriber Line*, или *цифровая абонентская линия*), используемая для высокоскоростной передачи данных по обычным медным телефонным проводам.

Для передачи данных используется также технология сотовой радиосвязи, которая разрабатывалась для сетей голосовой мобильной связи. Основным преимуществом систем сотовой связи является то, что при передаче данных станция может двигаться с большой скоростью. Поскольку при разработке этой технологии ставилась задача поддержания устойчивой голосовой связи с абонентом, который движется в автомобиле, оборудование сотовой связи может легко передавать поток пакетов данных на движущийся узел.

## 2.12. Резюме

В этой главе был проведен обзор аппаратного обеспечения нескольких сетевых технологий, которые используются для создания сетей на основе семейства протоколов TCP/IP. Были рассмотрены как дешевые технологии построения локальных сетей, типа Ethernet и FDDI, так и дорогостоящие глобальные магистральные сетевые системы, использующие арендованные выделенные каналы связи для передачи данных. Кроме того, мы показали, что семейство протоколов TCP/IP может использоваться для передачи пакетов данных совместно с другими технологиями, такими как сети общего пользования. При этом применяется метод туннелирования. Несмотря на то, что мы не углублялись в детали работы технологий (поскольку они не так важны), вы должны были уловить основную идею, сформулированную ниже.

*Семейство протоколов TCP/IP является универсальной сетевой технологией, которую можно использовать для передачи данных практически по любому типу оборудования и сетевым системам высокого уровня.*

## Материал для дальнейшего изучения

В первых компьютерных системах связи применялись двухточечные выделенные каналы связи. При этом в качестве устройства передачи данных часто использовались обычные последовательные порты, описанные в книге Мак-Намара (McNamara) [84]. В статье Меткалфа (Metcalf) и Боггса (Boggs) [86] описан прототип сети Ethernet со скоростью передачи данных 3 Мбит/с. Оригинальная технология Ethernet со скоростью передачи данных 10 Мбит/с описана в спецификации [43]. О стандарте IEEE 802.3 речь идет в отчете Нельсона [93]. Исторические перспективы развития технологии Ethernet описаны в статье Шоха (Shoch), Делала (Dalal) и Ределя (Redell) [118]. В отчете Абрамсона (Abramson) [1] можно ознакомиться с результатами исследования сети ALOHA, а в статье Коттона (Cotton) [36] сделан обзор технологий для создания локальных сетей.

Кольцевая сетевая технология с передачей маркера впервые предложена в статье Фармера (Farmer) и Ньюхолла (Newhall) [50]. Некоторые выводы по этой технологии сделаны в работах Миллера (Miller) и Томпсона (Thompson) [87], а также Эндрюса (Andrews) и Шульца (Shultz) [3]. Альтернативная технология — сеть с передачей сегментов (slotted ring network) — предложена в работе Пирса (Pierce) [103]. Сравнение этих двух технологий сделано в статье Розенталя (Rosenthal) [112].

За дополнительной информацией о сети ARPANET обратитесь к книге Церфа (Cerf) [18] и отчету BBN [6]. Идеи технологии X25NET обобщены в работе Камера (Comer) и Корба (Korb) [29]. Доступ к сети по коммутируемым каналам связи описан в статье Ланцилло (Lanzillo) и Партриджа (Partridge) [82]. Режим асинхронной передачи ATM и способы его использования в глобальных сетях описан в работе ДеПрудера (De Prycker) [42]. Обзор большого количества гигабитовых сетевых технологий, включая ATM и описание внутренней структуры высокоскоростных коммутаторов, приведен в книге Партриджа [98].

## Упражнения

- 2.1. Какие сетевые технологии используются в сети вашего предприятия?
- 2.2. Каков максимальный размер пакета, который может быть послан по высокоскоростной сети, например, Hyperchannel корпорации Network System?

- 2.3.** Если в вашей локальной сети используется концентратор Ethernet, узнайте у системного администратора, сколько машин можно к нему подключить. Если же ваша сеть построена на основе нескольких концентраторов (например, они могут располагаться на каждом этаже здания), определите способ их подключения друг к другу.
- 2.4.** Каковы преимущества и недостатки технологии туннельной передачи данных?
- 2.5.** Обратитесь к стандарту Ethernet и определите размер преамбулы и величину минимальной паузы при передаче пакетов. Какова максимальная скорость передачи данных по сети Ethernet в стационарном режиме?
- 2.6.** Какие параметры спутникового канала связи наиболее и наименее предпочтительны?
- 2.7.** Определите минимально возможное время передачи файла размером 5 Мбайт по сети с пропускной способностью 28,8 Кбит/с, 1,54 Мбит/с, 10 Мбит/с, 100 Мбит/с и 2,4 Гбит/с.
- 2.8.** Будет ли достаточно скорости центрального процессора, жесткого диска и системной шины современного компьютера для передачи данных из файла, находящегося на диске, со скоростью 2 Гбит/с?



# 3

## Основы и структура межсетевого взаимодействия

### 3.1. Введение

В предыдущих главах были рассмотрены механизмы передачи данных по сетям, которые лежат в основе любой компьютерной коммуникационной системы. В этой главе мы сосредоточимся на способах объединения разрозненных сетевых технологий в одну согласованную систему. Цель заключается в том, чтобы создать систему, обеспечивающую универсальные средства взаимодействия и не зависящую от использованного в ней сетевого оборудования. Поэтому сначала мы опишем абстрактную модель высокого уровня, которая лежит в основе всех проектных решений. В последующих главах будет показано, как на основе этой абстракции создаются требуемые уровни приложений, которые обеспечивают универсальный и независимый от физического транспортного уровня механизм передачи данных. Кроме того, в следующих главах будет рассмотрено использование созданной нами коммуникационной системы в прикладных программах.

### 3.2. Взаимодействие на уровне приложений

При проектировании независимых от сетевого аппаратного обеспечения систем передачи данных используется два различных подхода. Один из них заключается в том, что для скрытия неоднородностей в системе и особенностей низкоуровневых сетевых технологий используются специальные программы. В другом случае эти программы встраиваются прямо в операционную систему. Раньше разнородные сети объединялись в единообразную систему с помощью специальных прикладных программ, которые назывались *шлюзами уровня приложений* (*application gateways*). В подобных системах на каждом подключенном к сети компьютере запускалась специальная прикладная программа, которая обрабатывала особенности сетевого оборудования и обеспечивала сетевое соединение для этого компьютера. Кроме того, эта программа взаимодействовала с аналогичными программами, запущенными на других компьютерах сети. Например, некоторые системы электронной почты состоят из набора прикладных программ, запущенных на компьютерах сети. Каждая из этих программ настраивается на передачу сообщений соседнему компьютеру. В результате путь, по которому проходит сообщение от отправителя до получателя, может быть очень длинным (включая узлы, принадлежащие другим сетям). Однако на доставку сообщений это никак не влияет, поскольку почтовые системы всех компьютеров могут взаимодействовать друг с другом.

Использование специальных прикладных программ для скрытия деталей низкоуровневого сетевого взаимодействия может на первый взгляд показаться вполне естественным. Однако подобный подход сдерживает дальнейшее развитие системы и делает процесс обмена данными чрезвычайно громоздким. В самом деле, чтобы расширить функциональные возможности системы, необходимо для каждого компьютера создать новую прикладную программу. Добавление нового сетевого оборудования требует модификации существующих программ или создания новых программ для всех возможных приложений. Таким образом, каждая прикладная программа компьютера должна самостоятельно обеспечивать для себя сетевое соединение, что неминуемо влечет за собой дублирование кодов этих программ.

Разбирающиеся в сетях пользователи понимают, что с ростом числа взаимодействующих сетей до нескольких сотен или тысяч уже никто не сможет создать необходимый на все случаи жизни набор прикладных программ. Более того, описанный выше принцип пошагового обмена данными на уровне приложений требует корректной работы программ на всех компьютерах, участвующих в этом процессе. Если что-то случится с программой на одном из промежуточных звеньев цепи, ни отправитель, ни получатель уведомлены не будут и никак не смогут повлиять на устранение возникшей проблемы. Следовательно, системы, использующие промежуточные программы для реализации механизма взаимодействия, не могут гарантировать надежный обмен данными.

### 3.3. Взаимодействие на сетевом уровне

Кроме систем, взаимодействующих на уровне приложений, существует и альтернативный вариант — системы, взаимодействующие между собой на сетевом уровне. Сетевой механизм взаимодействия обеспечивает доставку небольших пакетов данных от отправителя прямо к получателю без использования промежуточных прикладных программ. Пересылка небольших фрагментов данных вместо огромных и громоздких файлов обладает рядом преимуществ. Во-первых, в нем непосредственно участвует низкоуровневое сетевое аппаратное обеспечение, что делает его чрезвычайно эффективным. Во-вторых, взаимодействие на сетевом уровне позволяет отделить логику приложения от механизма пересылки данных. В результате компьютеры на промежуточных узлах могут оперировать трафиком, не вникая, для каких сетевых приложений он предназначен. В-третьих, использование взаимодействия на сетевом уровне позволяет создать универсальную систему, предназначенную для выполнения практически любых видов компьютерной связи. В-четвертых, структура системы позволяет легко управлять сетью. Например, при появлении новых сетевых технологий требуется только внести соответствующие изменения в программное обеспечение сетевого уровня, не затрагивая при этом самих прикладных программ.

Идея реализации универсальной коммуникационной системы, взаимодействующей на сетевом уровне, вытекает из абстрактной концептуальной модели *объединенной сети* (*internetworking*). Понятие объединенной сети (*internetwork*, или *internet*), — чрезвычайно емкое. Оно позволяет выполнять процесс взаимодействия без учета подробностей реализации конкретной сетевой технологии, и скрыть от пользователя все низкоуровневые моменты. Более того, это понятие определяет принципы разработки программного обеспечения и проясняет методы обработки физических адресов и маршрутизации. Поэтому после рассмотрения основных причин принятия модели межсетевого взаимодействия, будут подробно описаны ее особенности.

Прежде всего сформулируем два основных постулата, которые следует иметь в виду при разработке коммуникационных систем.

- Не существует одной универсальной сетевой технологии, которая удовлетворяла бы всем требованиям.
- Идея универсального межсетевого взаимодействия как нельзя лучше соответствует пожеланиям пользователей.

Первый постулат основан на экономических, а также чисто технических соображениях. Недорогие сетевые технологии, применяемые для создания высокоскоростных локальных сетей, могут работать только на небольших расстояниях. В то же время стоимость глобальных коммуникационных технологий, применяемых при создании протяженных сетей, очень высока. По этой причине они крайне редко используются в локальных сетях. Поскольку ни одна сетевая технология не может удовлетворить всем требованиям, мы рассмотрим несколько основных низкоуровневых сетевых технологий.

Второй постулат очевиден. Пользователям хотелось бы организовать взаимодействие между двумя любыми узлами сети. В частности, хотелось бы иметь такую коммуникационную систему, которая не была бы связана рамками физических сетей.

Итак, наша конечная цель — создать унифицированную и согласованную объединенную сетевую систему, которая бы поддерживала ряд универсальных сетевых служб. В сетях такой системы могут использоваться низкоуровневые технологии передачи данных, наподобие тех, что были описаны в главе 2, “Обзор основных сетевых технологий”. Поэтому очевидно, что нужно разработать программное обеспечение и поместить его в иерархической модели между физическим уровнем передачи данных, зависящим от применяемой аппаратной технологии, и уровнем приложений. Это программное обеспечение как раз и будет той ширмой, за которой спрячутся низкоуровневые особенности применяемого сетевого оборудования. Кроме того, с его помощью можно будет связать набор разнородных сетей в одну большую сеть. Описанный принцип межсетевого взаимодействия называется *объединенной сетью* (*internetwork*, или *internet*).

Идея создания объединенной сети соответствует стандартному подходу, применяемому при проектировании сложных систем. Разработчики придумывают высокоуровневую компьютерную систему, для реализации которой используются существующие компьютерные технологии. При этом создаются необходимые уровни программного обеспечения, которые эмулируют с требуемой эффективностью возможности высокоуровневой системы. В следующем разделе описывается первый этап процесса проектирования, на котором выполняется более конкретная постановка задачи.

### **3.4. Особенности объединенной сети**

Безусловно, сама идея создания набора универсальных служб очень важна. Однако на ней не оканчивается список идей, которые было бы неплохо реализовать в объединенной сети, поскольку универсальным службам можно найти множество применений. При постановке задачи преследовалась цель скрыть от пользователя особенности низкоуровневой структуры объединенной сети. Другими словами, мы не хотим заставлять конечного пользователя или прикладную программу учитывать низкоуровневые особенности аппаратного обеспечения при использовании объединенной сети. Кроме того, нежелательно, чтобы взаимодействие зависело от топологии объединенной сети. В частности, добавление новой сети к системе не должно быть связано с ее подключением к центральному узлу коммутации или с прокладкой физических каналов связи ко всем существующим сетям. Идея заключается в том, чтобы можно было передавать данные между компьютерами отправителя и получателя по промежуточным сетям, даже

если последние напрямую не соединены с сетями, в которых находятся взаимодействующие абоненты. Также желательно, чтобы всем компьютерам объединенной сети назначался универсальный набор идентификационных параметров, которые можно считать их *именами*, или *адресами*.

В концепцию унифицированной объединенной сети также следует включить идею независимости пользовательского интерфейса от сети. Другими словами, нужно сделать так, чтобы набор действий, выполняемых при установке соединения или передаче данных, всегда оставался однотипным и не зависел от типа используемой сетевой технологии и компьютера получателя. Само собой разумеется, что при создании или использовании прикладных программ, взаимодействующих друг с другом, пользователь не должен вникать в тонкости топологии связанной сети.

### 3.5. Структура объединенной сети

В предыдущих главах были рассмотрены способы подключения компьютеров к самостоятельным сетям. Теперь возникает закономерный вопрос: “Как соединить между собой эти сети, чтобы они составляли единую сеть?” Ответ на него состоит из двух частей. Физически две сети можно соединить только при помощи компьютера, который подключен к обеим из них. Однако физическое соединение не обеспечивает межсетевое взаимодействие, идея которого была высказана выше. Причина в том, что подобное соединение не может гарантировать взаимодействие компьютеров, находящихся в разных сетях. Для того чтобы объединенная сеть заработала, необходимо выделить специальный компьютер, который будет передавать пакеты из одной сети в другую. Компьютеры, связывающие две сети и выполняющие пересылку пакетов между ними, называются *межсетевыми шлюзами* (*internet gateways*) или *маршрутизаторами* (*internet routers*).

В качестве примера рассмотрим объединение двух физических сетей, изображенных на рис. 3.1. Как видно из рисунка, маршрутизатор *R* подключен как к сети 1, так и к сети 2. Поскольку *R* является маршрутизатором, он должен перехватывать все пакеты, посланные из сети 1 компьютеру, находящемуся в сети 2, и выполнять передачу этих пакетов между сетями. Аналогично, маршрутизатор *R* должен перехватывать пакеты, посланные из сети 2 компьютеру находящемуся в сети 1, и выполнять передачу этих пакетов между сетями.

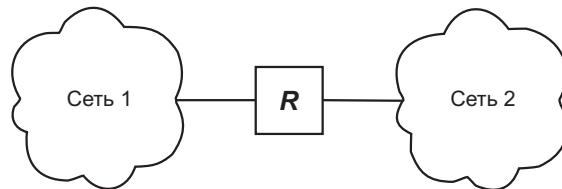


Рис. 3.1. Пример объединения двух физических сетей с помощью маршрутизатора *R* (IP-шлюза)

На рис. 3.1 физические сети обозначены в виде облака; этим подчеркивается, что не имеет значения, какое сетевое оборудование в них используется. Эти сети могут быть как локальными, так и глобальными, причем количество компьютеров в них может быть любым.

<sup>1</sup> Раньше в литературе часто можно было встретить термин *IP-шлюз* (*IP gateway*). Поскольку этот термин прижился у производителей сетевого оборудования, мы иногда будем им пользоваться как синонимом.

### 3.6. Объединение сетей с помощью IP-маршрутизаторов

Хотя на рис. 3.1 продемонстрирован основной принцип объединения сетей, все же показанную конфигурацию сетей можно считать очень упрощенной. На самом деле сложные объединенные системы состоят из большого количества сетей и маршрутизаторов. При этом каждый маршрутизатор должен обладать информацией о топологии объединенной системы, находящейся за пределами той сети, к которой он подключен. Давайте рассмотрим в качестве примера объединение трех сетей с помощью двух маршрутизаторов (рис. 3.2).

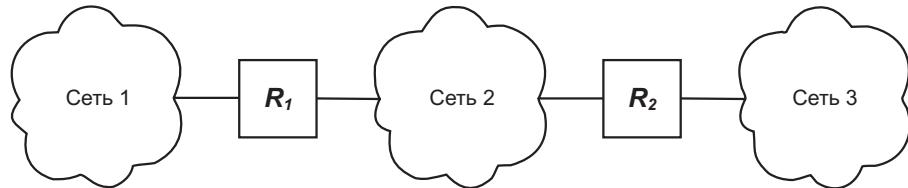


Рис. 3.2. Пример объединения трех сетей с помощью двух маршрутизаторов

В этом примере маршрутизатор  $R_1$  должен передавать из сети 1 в сеть 2 все пакеты, предназначенные для компьютеров, которые подключены как к сети 2, так и к сети 3. А теперь представьте себе большую объединенную сеть, состоящую из множества сетей. Очевидно, что в последнем случае принятие решения маршрутизатором о том, куда направлять пакеты, становится более сложной задачей.

На первый взгляд маршрутизаторы кажутся очень простыми устройствами, но от этого их значение не становится меньше, поскольку они обеспечивают средства для объединения сетей, а не просто отдельных компьютеров. Таким образом, только что мы установили важный принцип создания объединенной сети.

*При создании объединенной сети на основе семейства протоколов TCP/IP, взаимодействие между сетями обеспечивает специальные компьютеры, называемые IP-маршрутизаторами, или IP-шлюзами.*

Может показаться, что устройства, направляющие пакеты к получателям, должны быть мощными компьютерами с достаточным объемом внутренней и внешней памяти для хранения информации о каждой подключенной к объединенной сети машине. На самом деле маршрутизаторы, используемые в сетях TCP/IP, как правило, являются небольшими компьютерами. Часто они имеют жесткий диск небольшого размера и весьма скромный объем оперативной памяти. Это возможно благодаря тому, что маршрутизация пакетов выполняется в соответствии с приведенным ниже правилом.

*При перенаправлении пакетов маршрутизаторы используют не адрес получателя пакета, а адрес сети, в котором расположена машина получателя.*

Таким образом, если при перенаправлении пакетов исходить из адреса сети, то количество информации, которое должен хранить маршрутизатор, будет пропорционально количеству сетей в объединении, а не количеству подключенных к ним машин.

Поскольку маршрутизаторы играют основную роль в межсетевом взаимодействии, мы еще вернемся к ним в следующих главах и подробнее опишем принцип их работы и методы, с помощью которых они получают маршрутную информацию. А пока можно считать, что существует реальная возможность хранить в каждом маршрутизаторе объединенной сети информацию о маршрутах

следования пакетов ко всем ее сетям. Кроме того, мы будем полагать, что средства взаимодействия между физическими сетями обеспечиваются только с помощью маршрутизаторов.

### 3.7. Преимущества взаимодействия на сетевом уровне

Напомним, что семейство протоколов TCP/IP было разработано для обеспечения универсального взаимодействия между компьютерами, не зависящего от типов конкретных сетей, к которым эти компьютеры подключены. Поэтому с точки зрения пользователя такая объединенная сеть выглядит как единая виртуальная сеть, к которой подключаются все компьютеры. При этом детали конкретного физического подключения не имеют значения. Поэтому, чтобы облегчить конечному пользователю понимание механизма взаимодействия, объединенная сеть должна представляться в виде единой сети, а не совокупности отдельных сетей, как показано на рис. 3.3 (а). Для реализации такого подхода маршрутизаторов, связывающих физические сети, недостаточно. На каждом компьютере объединенной сети должно быть установлено также специальное программное обеспечение, с помощью которого прикладные программы смогут использовать объединенную сеть так, как если бы это была одна физическая сеть.

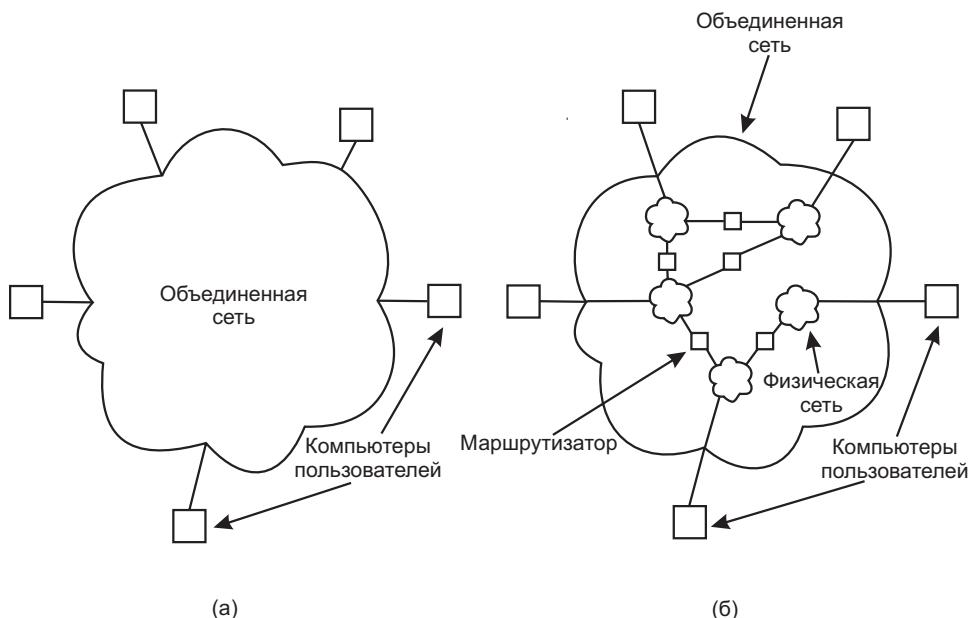


Рис. 3.3. Объединенная сеть TCP/IP с точки зрения конечного пользователя — предполагается, что каждый компьютер подключен к одной большой сети (а); схема подключения физических сетей с помощью маршрутизаторов (б)

Таким образом, преимущества взаимодействия на сетевом уровне понятны. Поскольку для прикладных программ, взаимодействующих между собой по объединенной сети, не имеет значения какова внутренняя структура механизма организации соединений, они могут запускаться без каких бы то ни было изменений на любом компьютере. Кроме того, так как детали физического подключения каждого компьютера к сети обрабатывает специальное межсетевое программное обеспечение, то при изменении топологии сети (добавлении или удалении каналов связи), изменения вносятся только в сетевые программы. Фактически становится

возможным оптимизировать внутреннюю структуру объединенной сети за счет изменения топологии физических каналов связи даже без остановки выполнения прикладных программ.

Второе преимущество взаимодействия на сетевом уровне заключается в его удобстве. Пользователи не должны знать структуру сетевых подключений и какие потоки данных по ним передаются. При написании прикладных программ, взаимодействующих по сети, физическую структуру сетевых подключений можно не учитывать. По сути, сетевые администраторы могут свободно изменять внутреннюю структуру объединенной сети, не изменяя прикладное программное обеспечение на большинстве компьютеров, подключенных к ней. Естественно, что при перемещении компьютера в другую физическую сеть придется изменить настройки его сетевого программного обеспечения.

Как видно из рис. 3.3 (б) между некоторыми парами сетей может не быть прямого соединения, обеспечиваемого маршрутизаторами. Поэтому для взаимодействия двух компьютеров, расположенных в таких сетях, нужно передавать пакеты через несколько маршрутизаторов и несколько промежуточных сетей. Таким образом, взаимодействующие сети можно сравнить с большой автомагистралью между городами. Каждая сеть берет на себя обязательство передавать транзитные потоки данных по своим каналам связи в обмен на право посыпать в объединенную сеть собственный трафик. При этом рядовые пользователи могут и не подозревать, что по их локальной сети проходит дополнительный трафик.

### 3.8. Все сети равноправны

В главе 2, “Обзор основных сетевых технологий”, был приведен краткий обзор сетевого оборудования, которое может использоваться для создания взаимодействующих между собой сетей TCP/IP, и продемонстрировано большое разнообразие существующих сетевых технологий. Выше отмечалось, что объединенная сеть состоит из набора связанных между собой равноправных сетей. На данный момент важно уяснить основной принцип: с точки зрения объединенной сети любая коммуникационная система, способная передавать пакеты, рассматривается как одна сеть, независимо от ее пропускной способности, задержек при передаче пакетов, максимального размера пакета и географической протяженности. В частности, на рис. 3.3 (б) физические сети обозначены небольшими облакками одинакового размера, поскольку, несмотря на их отличия, с точки зрения семейства протоколов TCP/IP они считаются равноправными. Следовательно можно сделать важный вывод.

*С точки зрения семейства протоколов TCP/IP все сети являются равноправными. Одной сетью считается и локальная сеть Ethernet, и глобальный магистральный канал и двухточечное соединение между двумя компьютерами.*

Читатель, не разбирающийся в структуре межсетевых взаимодействий, может не понять такого упрощенного подхода к сетям. По существу в семействе протоколов TCP/IP определено абстрактное понятие “сети”, которое не зависит от физических особенностей конкретной сети. Ниже будет показано, что подобная абстракция делает протокол TCP/IP чрезвычайно мощным средством взаимодействия.

### 3.9. Вопросы, оставшиеся без ответа

В проведенном выше кратком обзоре основных сетевых технологий, использующихся для создания объединенной сети, многие вопросы остались без ответа. Например, вас может заинтересовать точная форма сетевых адресов, назначаемых

компьютерам при подключении их к объединенной сети, а также то, как они относятся с аппаратными адресами технологий Ethernet, FDDI или ATM, описанных в главе 2. В следующих трех главах мы постараемся ответить на эти вопросы. В них будет описан формат IP-адресов и показано, как программно выполняется их преобразование в физические адреса. Кроме того, интересно рассмотреть вопрос о том, каков будет точный формат пакета по мере его прохождения по объединенной сети, а также что произойдет, если пакеты будут поступать на некоторый компьютер или маршрутизатор быстрее, чем он их сможет обработать. Ответ на эти вопросы вы найдете в главе 7, “Протокол IP: доставка дейтаграмм без установки соединения”. И наконец, вас может заинтересовать, как на одном компьютере может одновременно выполняться несколько прикладных программ, посылающих пакеты разным получателям и получающих от них ответы. При этом не происходит никакой путаницы с пакетами, отправляемыми и получаемыми другими приложениями. Кроме того, один из интересных вопросов заключается в том, как маршрутизаторы узнают о том, кому какие пакеты нужно передавать. Обо всем этом вы узнаете в следующих главах.

Хотя на данный момент это может быть и не так очевидно, выбранный нами способ изложения позволит вам освоить структуру и принципы применения межсетевого программного обеспечения и протоколов. Мы рассмотрим отдельно каждую составную часть, изучим материал в общем и опишем технические детали. Начали мы с описания физического уровня, на основе которого строится любой процесс межсетевого взаимодействия. В каждой последующей главе будет рассмотрена одна из составляющих межсетевого программного обеспечения. В результате у вас должна сложиться целостная картина того, как все эти части собрать воедино.

### **3.10. Резюме**

Объединенная сеть — это нечто большее, чем набор нескольких физических сетей, соединенных с помощью компьютеров. Для реализации механизма межсетевого взаимодействия необходимо, чтобы все связанные между собой системы подчинялись определенным правилам, позволяющим каждому компьютеру обмениваться данными со всеми остальными компьютерами. В частности, объединенная сеть должна позволять двум любым компьютерам взаимодействовать между собой даже в том случае, если между сетями, к которым они подключены, нет прямого канала связи. Такое возможно только в случае, когда всем компьютерам назначается набор универсальных идентификационных характеристик, и все они выполняют одинаковый набор процедур по перемещению данных к конечному получателю.

В объединенной сети взаимодействие между отдельными сетями выполняется за счет использования специальных компьютеров, физически подключаемых к двум или более сетям, которые называются IP-маршрутизаторами, или IP-шлюзами. Маршрутизаторы пересыпают пакеты из одной сети в другую.

### **Материал для дальнейшего изучения**

Описанная в этой главе модель межсетевого взаимодействия основывается на работах Церфа (Cerf) и Кеина (Cain) [16] и Церфа и Канна (Kahn) [17]. В них речь идет об объединенной сети, состоящей из набора отдельных сетей, связанных между собой маршрутизаторами. Кроме того, авторы в общих чертах описывают межсетевой протокол, очень похожий на тот, который впоследствии был положен в основу семейства протоколов TCP/IP. Более подробную информацию о структуре объединенной сети Internet можно найти в работах Постела (Postel)

[104]; Постела, Саншайна (Sunshine) и Чена (Chen) [105]; Хайндена (Hinden), Хэверти (Haverty) и Шельтцера (Sheltzer) [57]. В статье Шоха (Shoch) [117] представлены результаты исследований в области назначения межсетевых имен и адресов. Объединенная сеть, разработанная в исследовательском центре Пало Альто (Palo Alto Research Center, PARC) корпорации Xerox, являющаяся альтернативой рассматриваемым нами сетям TCP/IP, описана в статье Боггса (Boggs) и др. [12]. Реализация межсетевых взаимодействий в системе UNIX System V описана в работе Черитона (Cheriton) [19].

## Упражнения

- 3.1. Какова должна быть мощность процессоров, используемых в маршрутизаторах объединенной сети Internet? Были ли вы удивлены размерами и скоростью работы старых маршрутизаторов? Почему?
- 3.2. Оцените приблизительное количество сетей, входящих в объединенную сеть вашего предприятия. Какое количество маршрутизаторов при этом используется?
- 3.3. Проанализируйте структуру объединенной сети, представленной на рис. 3.3 (б). Какие из маршрутизаторов являются ключевыми и почему?
- 3.4. Изменение маршрутной информации — непростая задача, поскольку невозможно поменять эту информацию сразу во всех маршрутизаторах. Придумайте алгоритм, гарантирующий внесение изменений либо на всех маршрутизаторах сразу, либо ни на одном.
- 3.5. В объединенной сети маршрутизаторы периодически обмениваются информацией, находящейся в их таблицах маршрутизации. В результате появляется возможность подключения к сети нового маршрутизатора, который сразу же начнет пересыпал пакеты в нужном направлении. Придумайте алгоритмы обмена маршрутной информацией.
- 3.6. Сравните структуру объединенной сети TCP/IP с той, которая была разработана фирмой Xerox Corporation.



# 4

## Классовая адресация

### 4.1. Введение

В предыдущей главе было дано определение объединенной сети TCP/IP как единой виртуальной сети, состоящей из множества физических сетей, связанных между собой с помощью маршрутизаторов. В этой главе речь пойдет об адресации — важной составляющей любой системы, с помощью которой программам поддержки протокола TCP/IP удается скрыть особенности физической реализации конкретной сети и представить объединенную сеть как единую унифицированную систему.

### 4.2. Универсальная система идентификации

Говорят, что коммуникационная система предоставляет *универсальные услуги связи*, если она позволяет любому своему узлу взаимодействовать с любым другим узлом сети. Поэтому, чтобы сделать коммуникационную систему универсальной, необходимо определить приемлемый для всех метод идентификации каждого подключенного к ней компьютера.

Чаще всего узел сети характеризуют с точки зрения его *имени, адреса и маршрута*. В работе [117] Шоч (Shoch) предложил считать, что имя узла должно характеризовать *сам объект*, его адрес должен определять *положение объекта в сети*, а маршрут — *способ достижения объекта*<sup>1</sup>. Хотя приведенные выше определения даны на интуитивном уровне, они могут ввести вас в заблуждение. На самом деле имена, адреса и маршруты являются, в порядке понижения уровня иерархии, основными характеристиками системы идентификации узла. Вообще говоря, для идентификации компьютеров люди предпочитают пользоваться удобопроизносимыми именами, тогда как программное обеспечение эффективнее работает с компактной формой идентификации, которую называют *адресом*. Для универсальной идентификации машин в протоколе TCP/IP можно пользоваться либо именами, либо адресами. Однако было принято решение стандартизовать компактную двоичную форму записи адреса, которая позволяет повысить эффективность выполнения ряда компьютерных операций, таких как выбор маршрута следования пакетов. Поэтому далее будут рассматриваться только двоичные адреса, а вопросы их соответствия удобопроизносимым именам и методы использования адресов для маршрутизации пакетов мы оставим на потом.

---

<sup>1</sup> Данные, указывающие на то, где можно найти объект, называют *указателем ресурса*.

### 4.3. Оригинальная классовая система адресации

Итак, объединенную сеть следует рассматривать как одну большую сеть, подобно любой другой физической сети. Конечно, между ними есть разница, которая заключается в том, что объединенная сеть является виртуальной структурой, придуманной инженерами и реализованной целиком и полностью в виде программного обеспечения. В результате разработчики были вольны в выборе форматов пакетов, их размеров и методов доставки, системы адресации и т.п., поскольку их не ограничивали рамки применяемого оборудования. Для определения адресов разработчики протокола TCP/IP выбрали систему, принятую для обозначения сетевых физических адресов. Каждому узлу объединенной сети назначается 32-разрядный двоичный адрес, называемый *межсетевым адресом* (*internet address*), или *IP-адресом*. Особенность межсетевых адресов заключается в том, что они выбираются с учетом методов выполнения эффективной маршрутизации. В частности, в IP-адресе кодируется идентификационная информация о сети, к которой подключен узел, и уникальный номер узла в этой сети. Таким образом, можно сделать некоторые выводы.

*Каждому узлу объединенной сети TCP/IP назначается уникальный 32-разрядный адрес, который используется во всех сеансах связи с этим узлом.*

Для того чтобы вы уловили основную идею системы адресации протокола TCP/IP, мы для начала изложим материал в упрощенном виде, а затем постепенно будем его углублять. Итак, в простейшем случае каждому узлу объединенной сети назначается 32-разрядный универсальный идентификатор, который является его адресом. Часть IP-адреса (префикс) отводится под идентификатор сети. Таким образом, у всех машин некоторой сети префиксы IP-адресов будут одинаковыми.

Теоретически каждый адрес состоит из пары значений (*netid*, *hostid*), где часть *netid* идентифицирует сеть, а часть *hostid* — узел в этой сети. На практике же в рамках объединенной сети нет единого соглашения о том, какая часть IP-адреса выделяется для идентификации сети, а какая — узла, поскольку разработчики не указали в стандарте единых требований по этому поводу. В оригинальной системе адресации, которая теперь называется *классовой* (*classful*), IP-адрес должен быть представлен в одном из первых трех форматов<sup>2</sup>, показанных на рис. 4.1.



Рис. 4.1. Пять форматов IP-адресов, используемых в оригинальной классовой системе адресации. Первые три класса — А, В и С — можно идентифицировать по первым трем битам IP-адреса

<sup>2</sup> Четвертый формат зарезервирован для многоадресатной передачи в объединенной сети и будет описан чуть позже. А пока что мы ограничимся описанием первых трех форматов IP-адреса, определяющего объект в сети.

В классовой системе адресации IP-адрес является *автоматически опознаваемым*, поскольку его тип (или границу между префиксом и суффиксом) можно легко определить по значению адреса без обращения ко внешним источникам информации. В частности, класс адреса можно определить по значению трех старших битов. Обратите внимание, что для определения трех основных классов адреса достаточно проанализировать только два старших бита. Система адресации класса *A* используется в тех сетях, количество узлов в которых превышает  $2^{16}$  (или 65536). При этом 7 битов отводится под поле *netid*, а 24 бита — под поле *hostid*. Система адресации класса *B* используется для сетей средних размеров, количество узлов в которых превышает  $2^8$  (или 256), но меньше  $2^{16}$ . При этом под поле *netid* отводится 14 битов, а под поле *hostid* — 16 битов. Система адресации класса *C* применяется для сетей, имеющих менее  $2^8$  узлов. При этом под поле *netid* отводится 21 бит, а под поле *hostid* — 8 битов. Обратите внимание, что изначально система IP-адресов была задумана так, чтобы можно было быстро извлечь из двоичного адреса поля *netid* и *hostid*. Подобная эффективность особенно важна для тех маршрутизаторов, которые используют поле адреса *netid* для принятия решения о том, куда следует направлять пакеты. К способам поиска эффективных методов маршрутизации мы еще вернемся после рассмотрения изменений и дополнений, которые были внесены в оригинальную систему адресации.

#### 4.4. Адреса определяют сетевые соединения

Для того чтобы облегчить понимание материала, мы говорили, что один IP-адрес соответствует одному узлу сети. Однако строго говоря, такое утверждение не совсем верно. Давайте рассмотрим маршрутизатор, подключенный к двум физическим сетям. Поскольку IP-адрес определяет как сеть, так и узел в ней, может возникнуть вопрос: можем ли мы назначить маршрутизатору один IP-адрес, и если да, то какой?. Ответ на этот вопрос отрицательный. В случае если обычный компьютер имеет несколько физических соединений с сетью, он называется *многоадресным узлом (multi-homed hosts)*. Естественно, что многоадресному узлу, коим является и маршрутизатор, нужно назначить несколько IP-адресов. Причем каждый из адресов должен соответствовать одному из сетевых подключений компьютера. Итак, рассмотрение многоадресных узлов навело нас на важную мысль.

*Поскольку IP-адрес идентифицирует и сеть, и узел в этой сети, то можно сказать, что он соответствует не определенному компьютеру в сети, а его сетевому подключению.*

Таким образом, если маршрутизатор подключен к  $n$  сетям, то ему должно быть назначено  $n$  IP-адресов, по одному для каждого сетевого подключения.

#### 4.5. Сетевые и направленные широковещательные адреса

Мы уже говорили о преимуществах кодирования информации о сети в IP-адресе. Речь шла о том, что подобная структура адреса позволяет задействовать наиболее эффективные механизмы маршрутизации. Еще одно преимущество заключается в том, что IP-адрес может соответствовать как отдельному компьютеру сети, так и сетевому подключению этого компьютера. В соответствии с принятым соглашением, значение поля *hostid* равное нулю никогда не назначается отдельному узлу сети. Другими словами, IP-адрес с нулевым полем *hostid* используется для обозначения самой сети. Поэтому из всего сказанного выше можно сделать такой вывод.

*IP-адреса могут идентифицировать как отдельные узлы, так и сами сети. По принятому соглашению, IP-адрес, у которого все биты поля *hostid* равны нулю зарезервирован и используется для обозначения самой сети.*

Еще одним существенным преимуществом системы IP-адресов является то, что в ней предусмотрена возможность *направленной широковещательной адресации* (*directed broadcast address*) всех узлов сети. Согласно принятым стандартам, адрес, у которого значения всех битов поля *hostid* равны единице, зарезервирован и используется для направленной широковещательной передачи<sup>3</sup>. При отправке пакета по широковещательному адресу его копия передается по объединенной сети к сети получателя. Промежуточные маршрутизаторы на пути следования пакета обрабатывают только часть *netid* IP-адреса. При этом значение поля *hostid* никак не влияет на доставку пакета к сети получателя. Как только пакет достигнет маршрутизатора, подключенного к нужной сети, маршрутизатор выделяет из IP-адреса значение поля *hostid* и определяет метод доставки пакета. Если все биты этого поля равны единице, маршрутизатор выполняет широковещательную передачу пакета всем узлам указанной сети.

В большинстве сетевых технологий (таких как, например, Ethernet) широковещательная передача пакетов выполняется с той же эффективностью, что и передача обычных пакетов. В некоторых сетях широковещательная передача пакетов выполняется программным путем, в результате чего немного возрастают задержки в передаче, по сравнению с обычной передачей. Однако оборудование некоторых сетей вовсе не поддерживает режим широковещательной передачи. Поэтому использование направленного широковещательного IP-адреса отнюдь не гарантирует, что в сети получателя поддерживается широковещательный режим передачи или что пакеты будут доставлены с должной эффективностью всем узлам сети. Подводя итоги, можно сказать,

*IP-адрес может использоваться для указания режима направленной широковещательной передачи, в котором пакет посыпается сразу всем компьютерам конкретной сети. В случае, если это возможно, широковещательный IP-адрес преобразовывается в соответствующий аппаратный широковещательный адрес. Согласно принятому соглашению, направленный широковещательный адрес должен содержать в поле *netid* корректный идентификатор сети, а в поле *hostid* — все единицы.*

## 4.6. Ограниченнная широковещательная передача

В предыдущем разделе были рассмотрены так называемые *направленные* (*directed*) широковещательные адреса. Название “направленные” говорит о том, что в каждом адресе присутствует как корректный идентификатор сети, так и идентификатор узла, определяющий режим широковещательной передачи в этой сети. Направленные широковещательные адреса могут однозначно интерпретироваться в любой точке объединенной сети, поскольку из них можно выделить уникальный идентификатор сети получателя, а также определить, что заказан режим широковещательной передачи. Таким образом, направленные широковещательные адреса могут использоваться для реализации мощного (и поэтому иногда весьма опасного) механизма, позволяющего удаленному компьютеру

---

<sup>3</sup> К сожалению, в первых версиях программ, реализующих протокол TCP/IP для операционной системы Berkeley UNIX, по ошибке для широковещательной передачи использовались IP-адреса, у которых поле *hostid* равнялось нулю. Поскольку эта ошибка прижилась, во многих реализациях протокола TCP/IP предусматривалась возможность использования нулевого значения поля *hostid* для направленной широковещательной передачи.

послать единственный пакет, который затем будет автоматически разослан всем компьютерам указанной сети.

С точки зрения системы адресации основным недостатком направленной широковещательной передачи является то, что в адресе необходимо указывать идентификатор сети. Поэтому существует еще одна форма широковещательного адреса, который называют *ограниченным широковещательным адресом* (*limited broadcast address*), или *широковещательным адресом локальной сети* (*local network broadcast address*). Она предназначена для организации режима широковещательной передачи в локальной сети независимо от назначенного ей префикса IP-адреса. Локальный широковещательный адрес состоит из 32 единиц (поэтому иногда его называют широковещательным адресом, состоящим из всех единиц). Локальный широковещательный адрес используется в процедурах инициализации узла сети для определения его IP-адреса, или префикса IP-адреса локальной сети. После определения корректного IP-адреса узла или префикса локальной сети, узел должен переходить на использование направленного режима широковещательной передачи.

Как правило, в семействе протоколов TCP/IP режим широковещательной передачи данных ограничивается минимально возможным набором машин. В главе 10, “Бесклассовая адресация и подсети (CIDR)”, посвященной адресации подсетей, будет рассказано о том, как это правило влияет на несколько сетей, у которых общий префикс адреса.

## 4.7. Интерпретация нуля как значения “это”

Выше уже шла речь о том, что поле, состоящее из всех единиц, может быть интерпретировано как значение “для всех”, например “для всех узлов” сети. Вообще говоря, в межсетевом программном обеспечении поля, состоящие из одних нулей, интерпретируются как значение “это”. Такая интерпретация повсеместно встречается в литературе. Таким образом, IP-адрес, поле *hostid* которого состоит из одних нулей, интерпретируется как “этот” узел, а если поле *netid* равно нулю, то его значение интерпретируется как “эта” сеть. Естественно, подобная интерпретация адреса уместна только в том случае, если она может быть сделана однозначно. Например, если компьютер получает пакет, у которого в IP-адресе получателя поле *netid* состоит из одних нулей, а значение поля *hostid* соответствует адресу получателя, то он интерпретирует значение поля *netid* как “эта сеть” (т.е. сеть, по которой был получен пакет).

Нулевое значение поля *netid* используется в том случае, когда компьютер должен обменяться данными по локальной сети, но префикс IP-адреса этой сети еще не известен. Как уже было сказано, другие компьютеры локальной сети будут интерпретировать нулевое значение поля *netid* IP-адреса получателя пакета как “эта сеть”. В большинстве случаев в ответных пакетах будет содержаться реальный префикс сети, поэтому отправитель исходного пакета сможет сохранить его для дальнейшего использования. В главах 9, “Протокол IP: обработка ошибок и управляющие сообщения (ICMP)”, и 23, “Начальная загрузка и автоконфигурация (BOOTP, DHCP)”, будут более подробно описаны механизмы, с помощью которых компьютер локальной сети может определить идентификатор своей сети.

## 4.8. Механизмы адресации подсетей и суперсетей

В описанных механизмах адресации требовалось, чтобы каждой физической сети был назначен уникальный префикс IP-адреса. И хотя это требование было частью первоначального плана, оно не прижилось. В 1980-х годах, когда локальные сети получили широкое распределение, стало очевидным, что при

соблюдении требований уникальности префиксов IP-адресов для каждой физической сети все доступное адресное пространство быстро исчерпается. Поэтому для экономии сетевых префиксов механизм адресации был несколько изменен. В результате стало возможным назначать нескольким физическим сетям один и тот же префикс IP-адреса. Подобную систему адресации назвали *адресацией подсетей* (*subnet addressing*).

В 1990-х года было придумано еще один механизм расширения адресации, построенный без учета классовой иерархии адресов. Он позволял произвольным образом разбивать IP-адрес на префикс и суффикс. Такую систему адресации назвали *бесклассовой* (*classless*), а сами адресуемые сети — *супersetами* (*supernet*). Новая система адресации позволяла более эффективно использовать выделенное адресное пространство.

Расширения системы адресации для подсетей и супersetов будут описаны в главе 10, “Бесклассовая адресация и подсети (CIDR)”. А пока достаточно иметь в виду, что классовая система адресации, описанная в этой главе, была расширена и в чистом виде используется не так широко, как раньше.

## 4.9. Многоадресатная передача

Кроме *одноадресатной передачи* (*unicast delivery*), когда пакеты доставляются на конкретный компьютер, и *широковещательной передачи* (*broadcast delivery*), когда пакеты доставляются на все компьютеры конкретной сети, в системе IP-адресации предусмотрен специальный вид многоточечной доставки, который называется *многоадресатной передачей* (*multicasting delivery*). В последнем случае пакеты доставляются на заранее определенный набор компьютеров. Многоадресатная IP-адресация с большим успехом применяется в тех сетях, где оборудование позволяет реализовать режим многоадресатной доставки. Подробнее о многоадресатной системе адресации речь пойдет в главе 17, “Режим многоадресатной передачи в объединенной сети”. А пока следует уяснить, что для многоадресатной передачи используются адреса класса *D*.

## 4.10. Недостатки IP-адресации

Выше уже говорилось о том, что в IP-адресе “защита” информация о самой сети. Однако помещение информации о сети в IP-адрес имеет ряд недостатков. Один из самых очевидных недостатков заключается в том, что IP-адрес относится к сетевому соединению, а не к конкретному компьютеру.

*Поэтому, при перемещении компьютера из одной сети в другую, необходимо изменить его IP-адрес.*

Чтобы понять, насколько это неудобно, представьте себе абстрактного мобильного пользователя с портативным компьютером, который должен подключаться к Internet в разных географических точках в зависимости от того, куда его забросила судьба. Понятно, что в такой ситуации портативному компьютеру нельзя назначить фиксированный IP-адрес, поскольку сам адрес должен идентифицировать сеть, к которой этот компьютер подключен. В главе 19, “Протокол мобильной связи с IP-сетями”, будет показано, как система IP-адресации позволяет решить сложную проблему *мобильности* IP-адреса.

Еще одним недостатком классовой системы адресации является ограниченное количество компьютеров в сети класса *C*. Как только количество машин в такой сети превосходит 255, необходимо переходить на адреса класса *B*. И хотя эта проблема на первый взгляд кажется несущественной, изменение адресов в сети

может привести к значительным потерям времени и сложности в локализации возможных проблем. Поскольку в большинстве сетевого программного обеспечения не предусмотрена возможность назначения одной и той же физической сети нескольких IP-префиксов, сетевые администраторы не могут спланировать и осуществить постепенный переход к новой системе адресации. Вместо этого они должны менять адреса сразу на всех машинах (со всеми вытекающими последствиями).

Один из существенных недостатков системы межсетевой адресации станет понятен только после ознакомления с принципами маршрутизации. Однако учитывая важность, мы вкратце его здесь опишем. Выше уже шла речь о том, что маршрутизация в объединенной сети выполняется на основе IP-адресов. Для принятия решения о том, куда направлять конкретный пакет, маршрутизатор использует поле *netid* сетевого адреса. Давайте рассмотрим случай, когда компьютер имеет два физических подключения к объединенной сети. Выше мы уже говорили, что такому компьютеру нужно назначить два IP-адреса. Таким образом будет справедливым приведенное ниже утверждение.

*Поскольку при маршрутизации используется поле идентификатора сети IP-адреса, маршрут, по которому будут доставляться пакеты к компьютеру с несколькими IP-адресами, зависит от того, какой адрес получателя указан.*

Из этого утверждения следуют довольно неожиданные выводы. С точки зрения пользователя каждый компьютер в сети логично рассматривать как единое целое и использовать для его идентификации одно имя. Поэтому пользователей часто удивляет тот факт, что некоторым компьютерам назначается несколько имен и что маршрут следования пакетов к этому компьютеру может меняться в зависимости от того, по какому из его имен (точнее, адресов) будут посланы пакеты.

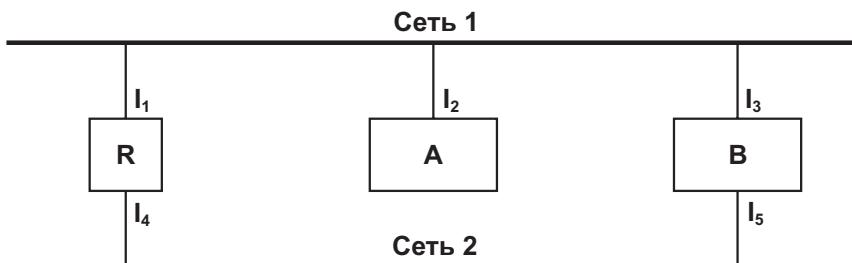


Рис. 4.2. Пример простой объединенной сети с многоадресным узлом B, демонстрирующий недостатки системы IP-адресации. Если интерфейс I<sub>3</sub> перестает функционировать, для обмена пакетами с узлом B через маршрутизатор R следует использовать IP-адрес I<sub>5</sub>.

Еще один неожиданный вывод заключается в том, что иногда недостаточно знать только один из IP-адресов узла сети для обмена с ним данными. Бывают случаи, когда невозможно доставить пакеты по одному из IP-адресов. Давайте в качестве примера рассмотрим простую объединенную сеть, показанную на рис. 4.2. Как видно из рисунка, к сети 1 подключены два компьютера — A и B, — которые могут напрямую взаимодействовать между собой. Поэтому для обращения к компьютеру B, пользователь компьютера A должен использовать IP-адрес I<sub>3</sub>. Однако существует и альтернативный маршрут от компьютера A к компьютеру B — через маршрутизатор R. Он используется том случае, когда пользователь компьютера A посыпает пакеты по IP-адресу I<sub>5</sub> (IP-адрес компьютера B в сети 2). А теперь предположим, что подключение компьютера B к сети 1 перестало

функционировать, но сам компьютер  $B$  находится в рабочем состоянии. Такое может случиться при повреждении кабеля, с помощью которого компьютер  $B$  подключен к сети 1, или выходе из строя соответствующей сетевой платы. Таким образом, пакеты, посланные с компьютера  $A$  по адресу  $I_3$  не смогут достичь компьютера  $B$ , хотя, если их отправить по адресу  $I_5$ , они достигнут получателя. Описанная только что проблема соответствия имен и адресов будет рассмотрена в следующих главах при обсуждении маршрутизации и привязки имен.

## 4.11. Точечная десятичная форма записи

Для удобства обращения, в технической документации или прикладных программах, IP-адрес указывают в виде четырех десятичных чисел, разделенных точками. При этом каждое число представляет один октет IP-адреса<sup>4</sup>. Таким образом, 32-разрядный IP-адрес 10000000 00001010 00000010 00011110 можно записать как 128.10.2.30.

В этой книге при выполнении операций с IP-адресами мы будем пользоваться точечной десятичной формой записи. Эта же форма записи используется практически во всех программах семейства протоколов TCP/IP, отображающих значение IP-адреса или требующих от пользователя его ввода. Например, в команде netstat системы UNIX при отображении или вводе IP-адреса пользователем чаще всего применяется точечная десятичная форма записи. (Для тех, кто не знает: эта команда используется для отображения маршрутной информации, сведений о сетевых соединениях и прикладных программах, таких как telnet или ftp.) Поэтому, при использовании классовой системы адресации важно понять, как выражаются IP-адреса разных классов в точечной десятичной форме записи. В табл. 4.1 приведены диапазоны изменения адресов для каждого из классов.

**Таблица 4.1. Диапазоны изменения IP-адресов разных классов, выраженных в точечной десятичной форме записи**

Класс	Минимальный адрес	Максимальный адрес
A	1.0.0.0	126.0.0.0
B	128.1.0.0	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	255.255.255.254

*Примечание.* Некоторые значения адресов зарезервированы для специального применения.

## 4.12. Адрес петли обратной связи

Проанализировав содержимое табл. 4.1, можно заметить, что классы сетей охватывают не все возможные диапазоны адресов. В частности префикс сети 127.0.0.0, который по принятой выше классификации должен относиться к сети класса A, зарезервирован для создания так называемой *петли обратной связи* (*loopback*). Она предназначена для тестирования работоспособности протокола TCP/IP и реализации механизма межпроцессного взаимодействия на локальном

<sup>4</sup> В точечной десятичной форме записи октеты иногда называют тетрадами.

компьютере. Если прикладная программа помещает в поле адреса получателя пакета адрес петли обратной связи, сетевое программное обеспечение обрабатывает пакет так, будто он только что получен по сети. При этом никакие данные в сеть не посылаются. В литературе четко сказано, что пакеты, предназначенные для сети с префиксом 127, не должны передаваться по какой бы то ни было физической сети. Более того, никакие узлы сети или маршрутизаторы не должны распространять информацию о маршрутизации или достижимости сети с адресом 127, поскольку он не является префиксом сетевого адреса.

### 4.13. Соглашение по использованию специальных адресов

На практике в системе IP-адресации используется несколько специальных адресов, содержащих в полях *netid* и *hostid* различные комбинации нулей (символ “это”) и единиц (символ “для всех”), как показано на рис. 4.3.



Примечания.

<sup>1)</sup>Разрешено использовать только в процессе инициализации сетевого программного обеспечения.  
Не является допустимым адресом получателя

<sup>2)</sup>Не является допустимым адресом отправителя

<sup>3)</sup>Пакеты с подобным адресом не должны появляться в сети

Рис. 4.3. Специальные формы IP-адресов, включающие допустимые комбинации нулей (символ “это”) и единиц (символ “для всех”). Длина сетевого префикса направлений широковещательных адресов зависит от класса используемой сети

Как отмечено в примечаниях к рис. 4.3, использование нулевого префикса сети разрешено только при выполнении инициализации сетевого программного обеспечения, которое обычно происходит при начальной загрузке компьютера. Он рассматривается как временная мера, позволяющая компьютерам в локальной сети обменяться начальной информацией. Как только компьютер определит свой IP-адрес и корректный идентификатор сети, он больше не должен использовать нулевой префикс сетевого адреса.

### 4.14. Управление адресацией в Internet

В пределах объединенной сети на основе семейства протоколов TCP/IP должно соблюдаться требование уникальности префиксов сетевых адресов. В тех организациях, где технология TCP/IP используется для создания закрытой внутренней сети (т.е. сеть, которая не имеет выхода в глобальную сеть Internet), префиксы адресов могут назначаться администрацией сети самостоятельно без согласования

с какими бы то ни было организациями. Однако если сеть предприятия планируется подключить к Internet, то в ней нужно использовать уникальные префиксы сетевых адресов, которые не должны совпадать с идентификаторами сетей других организаций. Поэтому, чтобы гарантировать уникальность префиксов IP-адресов по всей глобальной сети Internet, распределение сетевого адресного пространства должно выполняться централизованно. Первоначально контроль над назначением адресов сетей и установка единых правил их использования осуществлялся *Агентство по выделению уникальных параметров протоколов Internet* (*Internet Assigned Number Authority*, или *IANA*). С момента основания Internet и до осени 1998 г. назначением адресов ведал один человек — Джон Постел (*Jon Postel*). Он же руководил работой агентства IANA. После безвременной смерти Джона в конце 1998 г. для управления процессом назначения сетевых адресов была создана некоммерческая организация *по назначению имен и адресов в Internet* (*Internet Corporation For Assigned Names and Numbers*, или *ICANN*). Эта организация устанавливает единые правила использования и назначает имена, адреса и другие параметры протоколов.

В оригинальной классовой системе адресации, центральные органы Internet назначали адреса в зависимости от размеров сети. Адреса класса *C* выделялись сетям, к которым было подключено небольшое количество компьютеров (меньше 255), а адреса класса *B* резервировались для сетей с большим количеством компьютеров. И, наконец, чтобы получить адреса класса *A* к сети должно было быть подключено больше чем 65535 компьютеров. Адресное пространство выделялось неравномерно, поскольку большинство регистрируемых сетей было небольшого размера. Сетей среднего размера было совсем немного, а огромные сетевые системы вообще можно было пересчитать по пальцам.

Большинство организаций никогда не взаимодействовали с центральными органами Internet напрямую. Для подключения своих сетей к глобальной сети Internet они обычно заключали договора с локальными *поставщиками услуг*, или *провайдерами Internet* (*Internet Service Provider*, или *ISP*). Кроме обеспечения поддержки соединения между организацией заказчика и остальной частью Internet, провайдеры также назначали корректные префиксы адресов для всех сетей заказчика. На самом деле многие из мелких провайдеров являлись клиентами более крупных провайдеров. Поэтому все запросы на получение префиксов IP-адресов, поступившие от своих клиентов, они пересылали крупным провайдерам. Таким образом, на прямой контакт с ICANN выходили только очень крупные ISP.

Обратите внимание, что центральные органы Internet назначали только сетевую часть IP-адреса. После получения префикса сетевого адреса организация сама должна выбирать способы назначения уникального суффикса IP-адреса каждому компьютеру сети без вступления в контакт с центральными органами. Напомним, что в обязанности центральных органов входит назначение IP-адресов только тем сетям, которые должны быть подключены к глобальной сети Internet.

## 4.15. Зарезервированные префиксы IP-адресов

Как уже было сказано если сеть предприятия не имеет выхода во внешний мир, назначение уникальных адресов для его внутренних сетей TCP/IP является прерогативой администрации самого предприятия. Более того, в подобной ситуации каждая из групп может выбирать диапазоны адресов по своему усмотрению. Например, во внутренней сети фирмы IBM Corporation используются адреса, начинающиеся с 9.0.0.0, а в сети фирмы AT&T — с 12.0.0.0. Таким образом, если организация располагает двумя локальными сетями, в которых планируется использовать протокол TCP/IP без выхода в Internet, системный администратор вполне может назначить для них префиксы адресов 9.0.0.0 и 12.0.0.0.

Однако опыт подсказывает, что неразумно создавать закрытые объединенные сети, в которых используются такие же адреса, как и в глобальной сети Internet. Причина проста: большинство сетей рано или поздно все-таки подключаются к глобальной сети Internet, в результате чего возникают конфликты по адресам с другими сетями. Поэтому, чтобы избежать такой ситуации, инженерная группа IETF зарезервировала несколько префиксов IP-адресов и рекомендовала их использовать исключительно в закрытых сетях. Поскольку в набор зарезервированных префиксов включены как классовые, так и бесклассовые адреса, более подробно они будут рассматриваться в главе 10, “Бесклассовая адресация и подсети (CIDR)”.

#### 4.16. Пример

Чтобы прояснить систему IP-адресации, давайте рассмотрим в качестве примера топологию двух сетей, существовавших в середине 1980-х годов на факультете вычислительной техники университета Пердью, и покажем, как они были подключены к Internet. На рис. 4.4 показаны адреса сетей и схема их взаимодействия с помощью маршрутизаторов.

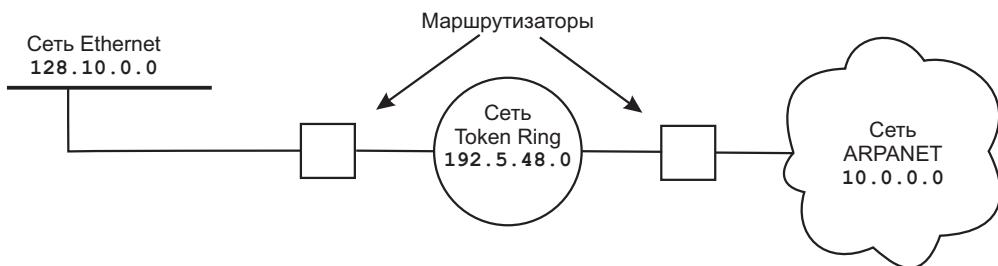


Рис. 4.4. Схема подключения двух сетей к магистральному каналу Internet. Каждой сети был назначен уникальный префикс IP-адреса

В приведенном выше примере задействованы три сети, которым назначены следующие адреса:

- ARPANET — 10.0.0.0;
- Ethernet — 128.10.0.0;
- Token ring — 192.5.48.0.

Согласно табл. 4.1, эти сети относятся к классам A, B и C, соответственно. На рис. 4.5 показана структурная схема двух факультетских сетей с именами подключенных к ним компьютеров и IP-адресами, назначенными для каждого сетевого соединения.

Как видно из рисунка, к объединенной сети подключено четыре компьютера: Arthur, Merlin, Guenevere и Lancelot. Два компьютера объединенной сети выполняют функции маршрутизатора пакетов. Это машина Taliesyn, которая связывает сети ARPANET и Token Ring, и машина Glatisant, связывающая сети Token Ring и Ethernet. Машина Merlin подключена как к сети Ethernet, так и к сети Token Ring, поэтому она может напрямую взаимодействовать с абонентами этих сетей. И хотя любой многоадресный узел, типа Merlin, можно настроить так, чтобы он выполнял маршрутизацию пакетов между двумя сетями, чаще всего для этих целей используются специальные выделенные компьютеры, или маршрутизаторы. Причина заключается в том, что выполнение функций маршрутизации требует привлечения дополнительных ресурсов как центрального

процессора, так и оперативной памяти. Особенно это относится к сетям с большим трафиком. Поэтому, как показано на рис. 4.5, функции маршрутизации трафика между сетями Ethernet и Token Ring выполняет отдельный компьютер Glatisant. (Реальный поток данных между этими двумя сетями существенно выше, чем предполагается в рассматриваемом нами примере. Причина в том, что для упрощения на рисунке показано только несколько машин, подключенных к объединенной сети, а на самом деле их гораздо больше.)

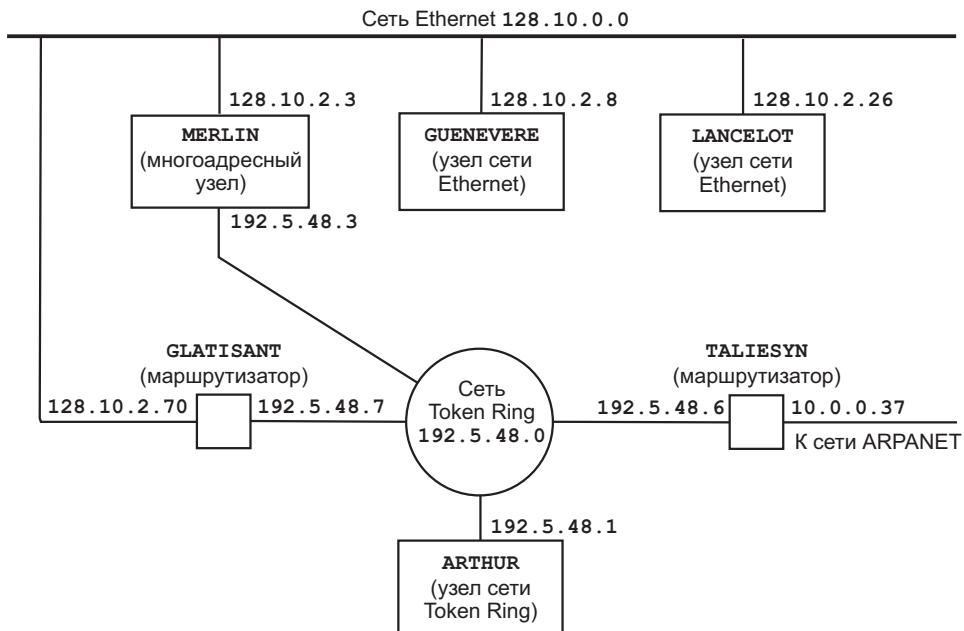


Рис. 4.5. Пример назначения IP-адресов для маршрутизаторов и компьютеров, подключенных к сетям, изображенным на рис. 4.4.

Как показано на рис. 4.5, каждому сетевому подключению назначен свой IP-адрес. Так, машине Lancelot назначен адрес 128.10.2.26, поскольку она имеет единственное подключение к сети Ethernet. Машине Merlin назначен IP-адрес 128.10.2.3 для интерфейса сети Ethernet и адрес 192.5.48.3 — для интерфейса сети Token Ring. Обратите внимание, что в рассматриваемом примере при назначении адресов для многоадресного узла сети для удобства сохранено значение его младшего байта адреса. Однако для маршрутизаторов Glatisant и Taliesyn адреса назначены несколько иначе. Например, для интерфейсов машины Taliesyn выбраны адреса 10.0.0.37 и 192.5.48.6, т.е. две строки цифр, абсолютно не связанных между собой. С точки зрения протокола IP не имеет значения, связаны как-то между собой цифры адреса компьютера, выраженные в точечной десятичной форме записи, или нет. Однако назначение компьютерам IP-адресов, цифры которых как-то связаны между собой или имеют какое-либо мнемоническое значение, облегчает жизнь обслуживающим сеть специалистам, системным администраторам и менеджерам. Таким образом, если для компьютера с несколькими интерфейсами назначить IP-адреса так, чтобы цифры в последнем октете совпадали, то обслуживающему персоналу легче их запомнить или идентифицировать интерфейс в конкретной сети.

## 4.17. Порядок следования байтов в сети

Чтобы работоспособность объединенной сети не зависела от используемых в ней компьютеров и сетевого оборудования, нужно определить единый вид представления данных с точки зрения программного обеспечения. Представьте себе, что произойдет, если, к примеру, программа, работающая на одном компьютере, отправит 32-разрядное целое число на другой компьютер. Сетевое оборудование, работающее на физическом уровне, передает последовательность бит с одной машины на другую без изменения порядка их следования. Однако не во всех типах компьютерных систем 32-х разрядные целые числа хранятся в одном и том же формате. В некоторых — младший по значимости байт целого числа хранится по меньшему адресу. Такой порядок следования байтов называется *прямым* (*Little Endian*). В других системах по меньшему адресу хранится старший по значимости байт целого числа. Подобный порядок следования байтов называется *обратным* (*Big Endian*). Кроме этих двух, существуют компьютерные системы, в которых целые числа составляются из групп 16-битовых слов. При этом по меньшему адресу хранится младшее по значимости слово, байты которого представлены местами. Таким образом, прямое копирование байтов с одного компьютера на другой без изменения порядка их следования может привести к искажению значения целых чисел.

Стандартизация порядка следования байтов в целых числах особенно важна в объединенной сети, поскольку такая информация, как адрес получателя или длина пакета, передается в виде бинарных данных. Естественно, что эти данные должны быть правильно интерпретированы как отправителем, так и получателем. В семействе протоколов TCP/IP описанная выше проблема решается за счет введения *стандартного* порядка следования байтов в сети, который должны использовать все компьютеры при помещении в пакет полей, содержащих двоичные данные. Перед отправкой пакета по сети, каждый компьютер или маршрутизатор должен преобразовать двоичные данные из машинного формата в стандартный. Аналогично, перед обработкой полученного пакета компьютер должен выполнить обратное преобразование. Естественно, что преобразование форматов содержащихся в пакете пользовательских данных, не выполняется, поскольку для протокола TCP/IP они представляют собой “черный ящик”. Прикладные программы могут по своему усмотрению форматировать и преобразовывать данные. В большинстве случаев для представления в пакете двоичных данных в сетевых прикладных программах выбирается стандартный порядок следования байтов, как и в протоколе TCP/IP. То есть эта данная проблема решается программистами уже на этапе написания приложений. Поэтому запускающий программы пользователь никогда не сталкивается с несовместимостью порядка следования байтов в полученных данных.

Для объединенной сети принят обратный (*Big Endian*) порядок следования байтов, т.е. самый старший байт передается первым. Таким образом, если попытаться представить себе пакет данных, который побайтно передается от одной машины к другой, то старший значащий байт целого двоичного числа, содержащегося в этом пакете, будет располагаться ближе к началу пакета, а самый младший значащий байт — ближе к его концу. По поводу формата представления данных в свое время было много дискуссий и до сих пор принятый стандарт периодически подвергается нападкам со стороны специалистов. В частности, ведутся споры по поводу того, что обратный порядок следования байтов уже давно устарел, поскольку практически во всех современных компьютерах для представления двоичных чисел используется прямой порядок следования байтов. Проблема состоит в том, что много лет тому назад, когда принимался этот стандарт, в большинстве компьютеров использовался как раз обратный порядок

следования байтов. Тем не менее каждый согласится с тем, что ключевым в этом споре является сам факт принятия стандарта, а каков этот стандарт — уже второстепенно. Поэтому пока стоит смириться с таким положением дел.

## 4.18. Резюме

В семействе протоколов TCP/IP в качестве адресов используются универсальные 32-х разрядные идентификаторы машин. Они называются межсетевыми адресами или IP-адресами и состоят из двух частей: префикса, идентифицирующего сеть, к которой подключен компьютер, и суффикса, уникального для каждого компьютера этой сети. Созданная ранее система IP-адресации называется классовой, поскольку каждый префикс сети относится к одному из трех основных классов. Для определения класса используются первые несколько битов IP-адреса. Количество возможных адресов в разных классах неодинаково. В первоначальной системе IP-адресации предусмотрено существование 127 сетей класса A, в каждой из которых может быть до 16 млн компьютеров. Сетей класса B может быть порядка 16 тыс., к каждой из которых можно подключить порядка 65 тыс. компьютеров. Сетей класса C может быть порядка 2 млн, к каждой из которых можно подключить до 254 компьютеров (первый и последний адреса зарезервированы для служебных нужд). Чтобы облегчить человеку восприятие IP-адресов, они выражаются точечной десятичной записью, т.е. в виде четырех десятичных чисел (октетов), разделенных точками.

Поскольку в IP-адресе закодирована информация как о сети, так и о конкретном узле этой сети, процесс маршрутизации пакетов выполняется очень эффективно. Одной из важных особенностей IP-адреса является то, что он определяет не конкретный компьютер, а интерфейс этого компьютера, с помощью которого он подключен к сети. Таким образом, если какой-либо компьютер подключен к нескольким сетям, ему будет назначено соответствующее количество IP-адресов. Одним из преимуществ системы IP-адресации является возможность посыпать пакеты одному узлу сети, группе компьютеров некоторой сети (многоадресатная доставка) или всем компьютерам сети (широковещательный режим передачи). Недостатком системы IP-адресации является возможность назначения одномуфизическому компьютеру нескольких адресов. В результате для выполнения гарантированной доставки пакета недостаточно знать какой-либо один из адресов машины, поскольку возможны ситуации, когда один из интерфейсов будет недоступен из-за обрыва сетевого кабеля или отказа оборудования.

Для обмена двоичными данными по сети между компьютерами с различными аппаратными платформами, в семействе протоколов TCP/IP принят стандартный порядок следования байтов в целых числах. С помощью двоичных целых чисел в полях пакета передается различная служебная информация, которая должна быть правильно интерпретирована всеми узлами сети. Перед отправкой пакета по сети, каждый компьютер или маршрутизатор должен преобразовать двоичные данные из машинного формата в стандартный. Аналогично, перед обработкой полученного пакета компьютер должен выполнить обратное преобразование.

## Материал для дальнейшего изучения

Рассмотренная в этой главе система IP-адресации была разработана Рейнольдсом (Reynolds) и Постелом (Postel) [RFC 1700]. За дополнительной информацией можно обратиться к работе Стала (Stahl), Романо (Romano) и Реккера (Recker) [RFC 1117].

Оригинальная система IP-адресации была усовершенствована после нескольких лет эксплуатации. Подробнее об этом речь пойдет в следующих главах.

В главе 10, “Бесклассовая адресация и подсети (CIDR)”, описывается одно из важных дополнений — так называемая бесклассовая система адресации. Она позволяет произвольным образом разбивать IP-адрес на префикс и суффикс. Кроме того, в главе 10 рассматривается важнейшая часть стандарта IP-адресации, позволяющая адресовать *подсети*. Благодаря этому можно использовать один префикс сетевого адреса в нескольких физических сетях. В главе 17, “Режим многоадресатной передачи в объединенной сети”, исследование методов IP-адресации будет продолжено. Там речь пойдет о том, как с помощью адресов класса D можно осуществлять *многоадресатную* передачу данных.

Порядок следования битов и байтов описывается в статье Коена (Cohen) [27]. Там же впервые вводятся термины *Big Endian* и *Little Endian*.

## Упражнения

- 4.1.** Вычислите точное максимально возможное количество сетей классов A, B и C. Определите точное максимальное количество узлов в этих сетях. При подсчетах не забудьте учесть широковещательный адрес, а также адреса классов D и E.
- 4.2.** Список назначенных адресов машин, представленный в форме, удобной для обработки на компьютере, иногда называют *таблицей хостов*. Если в вашем сетевом центре используется эта таблица, определите, сколько в ней находится адресов классов A, B и C.
- 4.3.** Сколько машин подключено к каждой из локальных сетей вашего предприятия? Существуют ли у вас такие локальные сети, для которых недостаточно применения адресной схемы класса C?
- 4.4.** В чем принципиальное отличие системы IP-адресации от системы нумерации телефонов?
- 4.5.** Очевидно, что один центральный орган в Internet не в состоянии достаточно оперативно обрабатывать заявки на регистрацию сетевых адресов. Попытайтесь предложить методику, позволяющую центральному органу перераспределить свои обязанности между несколькими группами, но так, чтобы гарантировалась уникальность адресов, назначаемых группами.
- 4.6.** Отличается ли стандартный порядок следования байтов в сети от того, который используется в вашем компьютере?
- 4.7.** Какое количество IP-адресов необходимо, чтобы уникальным образом идентифицировать каждый дом в вашей стране? А в мире? Будет ли при этом достаточно существующего пространства IP-адресов?



# 5

## Преобразование IP-адресов в физические адреса (ARP)

### 5.1. Введение

В предыдущей главе была описана система адресации протокола TCP/IP, согласно которой каждому узлу сети назначается уникальный 32-х разрядный адрес. Было сказано, что функционирование объединенной сети напоминает одну большую виртуальную сеть, поскольку при отправке и получении пакетов используются заранее назначенные адреса узлов сети. Были также рассмотрены аппаратные сетевые технологии и отмечено, что два компьютера, подключенные к одной физической сети, могут взаимодействовать между собой *только при условии, что им известны физические адреса друг друга*. При этом за кадром остался вопрос о том, как узел сети или маршрутизатор при отправке пакета по физической сети преобразует IP-адрес в корректный физический адрес. Этот вопрос и будет рассмотрен в данной главе на примере двух самых распространенных физических систем адресации.

### 5.2. Необходимость преобразования адресов

Давайте рассмотрим две машины *A* и *B*, подключенные к одной физической сети. Каждой из них назначен уникальный IP-адрес —  $I_A$  и  $I_B$  — и, кроме того, платы сетевого интерфейса этих машин имеют также свои физические адреса  $P_A$  и  $P_B$ . Наша цель — создать низкоуровневое программное обеспечение, которое позволит программам высокого уровня, игнорируя физические адреса, взаимодействовать между собой с помощью IP-адресов. Однако не стоит забывать о том, что в конечном счете взаимодействие между двумя машинами осуществляется с помощью физической сети. При этом может использоваться только та система физической адресации узлов, которая поддерживается конкретным сетевым оборудованием. Предположим, что машина *A* собирается отправить пакет машине *B* по физической сети, к которой они обе подключены. Однако машине *A* известен только IP-адрес ( $I_B$ ) машины *B*. При этом возникает вопрос, как машина *A* сможет определить физический адрес ( $P_B$ ) машины *B*?

Следует заметить, что преобразование адресов должно выполняться на каждом участке по пути следования пакета от отправителя до конечного получателя. В частности, возможно возникновение одной из двух ситуаций. Во-первых, на последнем этапе доставки пакета, его нужно отправить конечному получателю по той же физической сети. В этом случае компьютер, отправляющий пакет, должен преобразовать IP-адрес конечного получателя в физический адрес. Во-вторых, на промежуточных участках по пути следования пакет должен отправляться на очередной узел

маршрутизации. Поэтому промежуточный отправитель должен преобразовать IP-адрес следующего узла маршрутизации в физический адрес.

Описанная выше задача конвертирования адресов высокого уровня в физические адреса называется *проблемой преобразования адресов* (*address resolution problem*) и решается несколькими способами. В некоторых семействах протоколов ведется специальная таблица соответствия адресов высокого уровня физическим адресам. Эта таблица хранится на каждом узле сети. В других семействах протоколов проблема решается путем кодирования аппаратного адреса и создания на его основе сетевого адреса высокого уровня. Следует заметить, что при использовании только одного из приведенных выше методов редко удается создать эффективную систему адресации высокого уровня. Поэтому ниже будут рассмотрены два способа преобразования адресов, используемые в семействе протоколов TCP/IP, а также указано, когда наиболее уместно использовать каждый из них.

### 5.3. Два типа физических адресов

Существует два типа физических адресов. Один из них был рассмотрен в главе 2, “Обзор основных сетевых технологий”, на примере технологии Ethernet. По сути он представляет собой двоичное целое число высокой разрядности, которое назначается в качестве физического адреса при изготовлении устройства и в дальнейшем не может быть изменено. Второй тип физического адреса был также рассмотрен в главе 2 на примере сетей proNET. Это — небольшое целое число, которое назначается в качестве физического адреса сетевым администратором при установке сетевой платы в компьютер и может быть при необходимости изменено. Таким образом, проблема преобразования адресов для сетей наподобие proNET решается очень просто, чего нельзя сказать для сетей типа Ethernet. Начнем с рассмотрения легкой задачи.

### 5.4. Преобразование адресов методом прямого отображения

Рассмотрим сеть с передачей маркера типа proNET. В главе 2, “Обзор основных сетевых технологий”, уже говорилось о том, что в качестве физических адресов в ней используются небольшие целые числа и что пользователь должен выбрать значение физического адреса непосредственно перед установкой платы сетевого интерфейса в компьютер. Ключевой момент, облегчающий процесс преобразования адресов в сети данного типа заключается в том, что в процессе инсталляции сетевому адаптеру нужно назначить как IP-адрес, так и физический адрес. Следовательно, ничто не мешает выбрать эти адреса так, чтобы их части совпадали. Обычно IP-адреса назначаются так, что значение их поля `hostid` увеличивается на единицу при переходе от одной машины к другой, например 1, 2, 3. Поэтому при установке платы сетевого интерфейса можно выбрать ее физический адрес так, чтобы он соответствовал значению поля `hostid` IP-адреса. Например, системный администратор вполне может назначить физический адрес 3 компьютеру, IP-адрес которого равен 192.5.48.3, поскольку последний относится к классу C и его поле `hostid` равно 3.

Таким образом, для сетей типа proNET задача определения физического адреса по IP-адресу является очень простой. Достаточно из IP-адреса извлечь значение поля `hostid`. Эта операция на всех компьютерных платформах выполняется очень быстро, поскольку для ее реализации требуется всего несколько машинных команд. Описанный выше процесс преобразования адреса эффективен еще и потому, что для его выполнения не требуется обращаться ко внешним источникам

данных. И наконец, при подключении новых компьютеров к сети не нужно изменять старые установки или перекомпилировать код.

Выражаясь математическим языком, при назначении системы нумерации, которая бы позволяла эффективно преобразовывать адреса, нужно выбрать функцию  $f$ , с помощью которой по IP-адресу будет вычисляться физический адрес. В зависимости от используемого оборудования разработчики могут также выбирать ту или иную систему физических адресов. Таким образом, преобразование адреса  $I_A$  сводится к вычислению функции

$$P_A = f(I_A)$$

Наша задача сделать так, чтобы функция  $f$  вычислялась с максимальной эффективностью. Если набор физических адресов ограничен, то эффективное преобразование адресов можно организовать иначе, чем в приведенном примере. Например, при использовании протокола IP в сети, требующей установки соединения, наподобие ATM, отсутствует возможность выбора физических адресов. В таких сетях на одном или нескольких компьютерах, выполняющих роль серверов, хранятся пары адресов. Каждая пара представляет собой IP-адрес и соответствующий ему физический адрес. Обычно для ускорения поиска серверы хранят пары адресов в таблице, находящейся в оперативной памяти. В подобных случаях для повышения эффективности поиска соответствия в таблице программы могут использовать заранее оговоренную хеш-функцию. В конце главы, в упр. 5.1 предложен еще один подход к решению описанной проблемы.

## 5.5. Преобразование адресов методом динамической привязки

Чтобы понять, почему преобразование адреса в некоторых сетях трудно осуществить, давайте в качестве примера рассмотрим технологию Ethernet. Как говорилось в главе 2, “Обзор основных сетевых технологий”, каждой плате сетевого интерфейса Ethernet назначается уникальный 48-битовый адрес на заводе-изготовителе. Поэтому физический адрес компьютера может измениться, если плата сетевого интерфейса выйдет из строя и будет заменена. Более того, поскольку адрес Ethernet состоит из 48 битов, нет никаких шансов его закодировать в 32-х разрядный IP-адрес<sup>1</sup>.

Разработчики семейства протоколов TCP/IP подошли творчески к решению проблемы преобразования адресов в сетях, поддерживающих возможность широковещательной передачи данных, наподобие Ethernet. Придуманный ими метод позволяет добавлять к сети новые узлы или маршрутизаторы без необходимости перекомпиляции исходного кода сетевого программного обеспечения и без поддержки централизованной базы данных. Чтобы избежать необходимости поддерживать таблицу соответствия адресов, разработчики предпочли использовать низкоуровневый протокол динамической привязки адресов, который назвали *протоколом преобразования адресов* (*Address Resolution Protocol*, или ARP). С его помощью удалось реализовать довольно эффективный и несложный механизм конвертирования адресов.

Как показано на рис. 5.1, идея, положенная в основу механизма динамической привязки протокола ARP, очень проста. Как только узлу сети  $A$  требуется преобразовать IP-адрес  $I_B$ , он посылает в режиме широковещания специальный пакет, который предписывает узлу с IP-адресом  $I_B$  сообщить свой физический

<sup>1</sup> Поскольку метод прямого отображения, рассмотренный в предыдущем разделе является более удобным и эффективным, чем метод динамической привязки, новое поколение протоколов IPv6 создано с учетом возможности кодирования 48-битового аппаратного адреса в IP-адресе.

адрес ( $P_B$ ). Данный пакет будет получен всеми узлами сети, включая  $B$ . Однако ответ, содержащий физический адрес, посыпает только узел  $B$ , поскольку только ему назначен IP-адрес  $I_B$ . Получив ответный пакет, узел  $A$  использует присланный ему физический адрес для непосредственной отправки пакета узлу  $B$ . Таким образом, можно подвести некоторые итоги.

*Протокол преобразования адресов ARP позволяет узлу сети узнать физический адрес узла-получателя, подключенного к той же физической сети, используя при этом только IP-адрес получателя.*

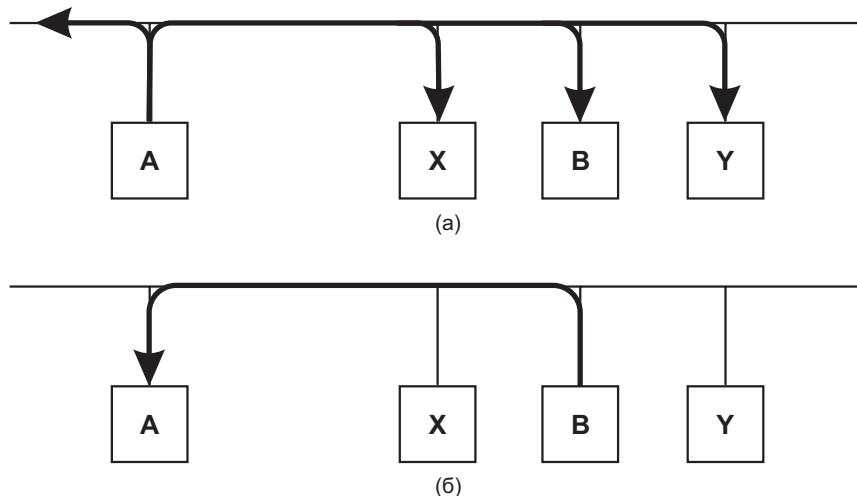


Рис. 5.1. Схема работы протокола ARP. Чтобы определить физический адрес ( $P_B$ ) узла  $B$ , узел  $A$  отправляет широковещательный запрос, содержащий IP-адрес ( $I_B$ ) узла  $B$ , ко всем машинам сети (а). В ответ на запрос узел  $B$  отправляет узлу  $A$  пару значений ( $I_B$ ,  $P_B$ ) (б)

## 5.6. Кэширование запросов ARP

На первый взгляд может показаться весьма странным, что, прежде чем машина  $A$  сможет отправлять пакеты машине  $B$ , она должна послать широковещательный запрос всем машинам сети, который достигает в том числе и машины  $B$ . Еще более странным кажется то, что машина  $A$  сначала посылает широковещательный запрос, в котором, выражаясь человеческим языком, содержится вопрос: “Как мне связаться с машиной  $B$ ?", вместо того, чтобы сразу послать пакет машине  $B$  в широковещательном режиме. Однако для этого существует весомая причина. Все дело в том, что широковещательный режим — слишком дорогой и ресурсоемкий, чтобы использовать его для простого обмена пакетами между двумя машинами. В самом деле, пакет, посланный в этом режиме, получают все машины сети, и каждая из них (кроме одной) должна впустую потратить время на его обработку.

Чтобы уменьшить накладные расходы, программы, использующие ARP, должны кэшировать ответы на несколько последних сделанных запросов, содержащих информацию об IP-адресах и соответствующих им физических адресах. Другими словами, как только программа получает ответ на посланный ARP-запрос, она сохраняет IP-адрес и соответствующий ему физический адрес в локальной памяти компьютера для дальнейшего использования. Перед отправкой

нового ARP-запроса программа всегда сначала просматривает кэш-память. Если нужное соответствие адресов найдено, широковещательный ARP-запрос в сеть не посыпается. Таким образом, если двум компьютерам в сети нужно обменяться информацией, они вначале шлют друг другу широковещательные ARP-запросы. После получения ответа все последующие передачи пакетов выполняются уже напрямую, без предварительных ARP-запросов. Опыт эксплуатации сетей показывает, что, поскольку в большинстве сеансов обмена информацией пересыпается более одного пакета данных, использование кэш-памяти даже небольшого размера уже имеет смысл.

## 5.7. Тайм-аут кэширования запросов ARP

Рассмотренный в предыдущем разделе механизм кэширования запросов ARP представляет собой один из примеров применения технологии систем с *неустойчивым состоянием* (*soft state*). Эта технология широко используется в сетевом программном обеспечении. Ее название говорит о том, что возможны ситуации, когда информация, содержащаяся у клиента, устаревает, а он об этом даже ничего не подозревает, поскольку никакие уведомления он не получает. Возвращаясь к ARP, давайте рассмотрим две машины, *A* и *B*, подключенные к сети Ethernet. Предположим, что машина *A* посылает ARP-запрос, на который машина *B* отвечает и после этого выходит из строя. При этом машина *A* не получит уведомления о том, что с машиной *B* что-то случилось. Более того, поскольку в кэш-памяти ARP-запросов машины *A* находится адресная привязка для машины *B*, машина *A* будет продолжать направлять пакеты машине *B*. В аппаратном обеспечении Ethernet не предусмотрены средства для контроля за состоянием машины *B*, поскольку в технологии Ethernet отсутствует механизм гарантированной доставки пакетов. Поэтому машина *A* не располагает средствами, позволяющими установить момент, когда информация, хранящаяся в ее кэш-памяти ARP-запросов, не соответствует действительности.

Таким образом, в системах с неустойчивым состоянием ответственность за достоверность информации возложена на владельцев этой информации. На практике это означает, что в сетевых программах, реализующих подобные системы, используются специальные таймеры, определяющие интервал времени, в течение которого информация считается достоверной. По истечении установленного времени какая бы то ни было информация о состоянии системы удаляется из кэш-памяти. Например, после того, как информация об адресной привязке помещена в кэш-память ARP-запросов, протокол ARP требует, чтобы программа установила значение таймера, равное, как правило, 20 минутам. По истечении установленного времени информация из кэш-памяти должна быть удалена. После этого события могут развиваться по одному из двух сценариев. Во-первых, если никакие пакеты больше не будут посыпаться получателю, проблема решается сама собой. Во-вторых, если получателю нужно отправить пакет, а в кэш-памяти заданная адресная привязка отсутствует, программа выполняет обычную процедуру посылки широковещательного ARP-запроса и обработки его ответа. Если к этому моменту машина получателя возобновила работу, информация об адресной привязке снова помещается в кэш-память ARP-запросов отправителя. Если же нет, отправитель отмечает тот факт, что машина получателя недоступна.

Использование в протоколе ARP алгоритмов с неустойчивым состоянием имеет как преимущества, так и недостатки. Основное преимущество заключается в независимости подобных систем. Во-первых, каждый компьютер может независимо от других компьютеров сети определять момент, когда нужно обновить информацию в кэш-памяти ARP-запросов. Во-вторых, отправителю для определения факта неправильной адресной привязки не нужно связываться с машиной

получателя или какой бы то ни было другой машиной. Достаточно отправить ARP-запрос, и если предполагаемый получатель на него не ответит, значит, его компьютер по какой-либо причине отключен от сети. В-третьих, предложенная методика обеспечивает надежный механизм передачи сообщений независимо от типа используемого сетевого оборудования. Основной недостаток применения алгоритмов с неустойчивым состоянием заключается в возникающих при этом задержках. В самом деле, если величина тайм-аута равняется  $N$  секунд, то отправитель узнает о том, что с получателем что-то случилось только по прошествии  $N$  секунд.

## 5.8. Усовершенствования протокола ARP

В протокол ARP было внесено несколько усовершенствований. Во-первых, очевидно, что, если машина  $A$  посыпает в сеть ARP-запрос для определения физического адреса машины  $B$ , чтобы отправить ей пакет данных, высока вероятность того, что впоследствии машине  $B$  потребуется отправить пакет машине  $A$ . Поэтому, чтобы упредить ARP-запрос от машины  $B$  в отношении физического адреса машины  $A$  и таким образом минимизировать лишний трафик в сети, посыпая ARP-запрос, машина  $A$  должна поместить в него свою адресную привязку (IP-адрес машины  $A$  и соответствующий ему физический адрес). При получении запроса машина  $B$  извлекает из него адресную привязку машины  $A$  и помещает ее в свою кэш-память, после чего посыпает ответ машине  $A$ . Во-вторых, поскольку машина  $A$  посыпает в сеть исходный ARP-запрос в широковещательном режиме, его принимают все машины сети. А это означает, что они могут извлечь из него адресную привязку машины  $A$  и поместить ее в свою кэш-память. В-третьих, при замене платы сетевого интерфейса (например, в случае выхода ее из строя) меняется физический адрес компьютера. А это означает, что необходимо предусмотреть механизм оповещения компьютеров локальной сети, в кэш-памяти ARP-запросов которых могла остаться старая адресная привязка. Компьютер может уведомить других абонентов сети о своем новом адресе, послав широковещательный ARP-запрос в момент начальной загрузки.

Приведенные выше соображения можно подытожить следующим образом.

*В каждый ARP-запрос отправитель включает свою адресную привязку (IP-адрес и соответствующий ему физический адрес). Получатели помещают эту привязку в свою кэш-память перед обработкой полученного ARP-пакета.*

## 5.9. Взаимосвязь ARP с другими протоколами

В протоколе ARP реализован один из возможных механизмов сопоставления IP-адресов физическим адресам. Выше уже отмечалось, что подобный механизм нужен не для всех сетевых технологий. Секрет в том, что без ARP можно обойтись, если сделать так, чтобы сетевое оборудование смогло само распознавать IP-адреса. Таким образом, ARP попросту является высокогорневым механизмом адресации, который использует низкогорневый механизм адресации конкретного сетевого оборудования. Эту идею можно подытожить следующим образом.

*ARP является низкогорневым сетевым протоколом, который позволяет скрыть особенности используемого сетевого оборудования благодаря назначению абонентам сети уникальных IP-адресов. ARP следует рассматривать как неотъемлемую составную часть физической сетевой системы. Он не входит в семейство протоколов TCP/IP.*

## 5.10. Реализация протокола ARP

Протокол ARP состоит из двух частей: первая выполняет преобразование IP-адреса в физический адрес при отправке пакета, а вторая отвечает на запросы, поступающие от других машин сети. Преобразование адресов при отправке пакетов реализовать довольно просто. Однако существуют некоторые моменты, усложняющие его реализацию. Определив IP-адрес получателя, программа сетевого протокола выполняет поиск в кэш-памяти ARP-запросов его привязки к физическому адресу. Найдя привязку, программа извлекает из кэш-памяти физический адрес получателя и помещает его в соответствующее поле фрейма, после чего фрейм отправляется получателю. Если же соответствующая привязка в кэш-памяти не обнаружена, программа должна отправить в сеть широковещательный ARP-запрос и подождать получения на него ответа.

Определение физического адреса машины с помощью широковещательного ARP-запроса может оказаться непростым делом. Например, нужный компьютер может быть отключен от сети либо сильно перегружен, поэтому он не сможет во время принять и обработать ARP-запрос. В результате отправитель может либо вообще не получить ответ на свой запрос, либо получить его с большой задержкой. Поскольку в технологии Ethernet доставка пакетов не гарантируется, первоначальный ARP-запрос также может быть утерян. В последнем случае отправитель должен как минимум еще раз повторить ARP-запрос. При этом он хранит в памяти исходный пакет, который должен быть отправлен получателю после преобразования адреса<sup>2</sup>. Кроме того, управляющая программа компьютера должна принять решение о том, нужно ли выполнять другие прикладные программы в тот момент, когда компьютер выполняет ARP-запрос. В большинстве случаев в период ожидания ответа на посланный ARP-запрос операционная система передает управление другой прикладной программе. Поэтому в сетевом программном обеспечении нужно предусмотреть обработку ситуации, когда по инициативе другой прикладной программы генерируется еще один ARP-запрос на преобразование того же самого IP-адреса. При этом в сеть не должны посыпаться повторные запросы на получение привязки для одного и того же IP-адреса.

Давайте рассмотрим случай, когда машина A получила адресную привязку для машины B, после чего плата сетевого интерфейса машины B вышла из строя и была заменена. И хотя физический адрес машины B изменился, содержимое кэш-памяти ARP-запросов осталось прежним. В подобном случае машина A будет посылать в сеть пакеты по несуществующему физическому адресу, что делает невозможным их доставку получателю. На примере описанного выше случая мы показали, почему так важно, чтобы с точки зрения программного обеспечения протокола ARP таблица привязок рассматривалась как кэш-память, элементы которой должны быть удалены после истечения заданного интервала времени. Естественно, что значение таймера для конкретного элемента кэш-памяти должно быть переустановлено после получения ответа на посланный ARP-запрос, в котором содержится нужная адресная привязка. Однако следует заметить, что значение таймера не переустанавливается в том случае, если элемент кэш-таблицы используется для отправки пакета.

Вторая часть кода, реализующего протокол ARP, предназначена для обработки пакетов с ARP-запросами, поступившими от других компьютеров сети. После получения ARP-пакета программа должна сперва извлечь из него IP-адрес отправителя и соответствующий ему физический адрес. После этого она должна проанализировать содержимое локальной кэш-памяти и выяснить, находится ли там адресная

---

<sup>2</sup> Если время получения ответа на ARP-запрос будет очень большим, отправитель может решить, что компьютер получателя недоступен, и удалить пакет из очереди на отправку.

привязка для отправителя. Если соответствующий элемент найден в кэш-памяти, обработчик полученного ARP-запроса должен заменить в нем физический адрес отправителя на тот, что был извлечен из ARP-пакета. После выполнения описанных действий, получатель должен обработать остальную часть ARP-пакета.

Получатель должен обрабатывать два типа ARP-пакетов. При поступлении ARP-запроса получателю нужно проверить, действительно ли этот запрос предназначается ему (т.е. некоторая машина в локальной сети отправила широковещательный запрос на определение физического адреса получателя). В этом случае программное обеспечение протокола ARP, запущенное на машине получателя, должно сформировать ответный пакет, поместить в него физический адрес и отправить его напрямую машине, пославшей запрос. Получатель должен также поместить адресную привязку отправителя в свою кэш-память ARP-запросов, если ее там еще нет. Если же в полученном ARP-запросе IP-адрес не соответствует IP-адресу локальной машины, то это означает, что данный пакет предназначается другой машине локальной сети и его нужно проигнорировать.

Интересно рассмотреть также процесс получения машиной ответа на ARP-запрос. В зависимости от реализации программного обеспечения, элемент в кэш-памяти создается либо обработчиком ответа, либо непосредственно перед отправкой ARP-запроса. В любом случае, после обновления содержимого кэш-памяти получатель должен попытаться сопоставить ответ с ранее выданным запросом. Обычно ответный пакет приходит в результате ARP-запроса, сгенерированного машиной, которой нужно доставить пакет с данными по назначению. За время, прошедшее с момента широковещательной посылки ARP-запроса и до получения на него ответа, некоторая прикладная программа или программное обеспечение протоколов высокого уровня может сгенерировать еще один ARP-запрос на тот же самый адрес. Поэтому в программах, реализующих протокол ARP, необходимо запоминать посланные запросы и не допускать отправку дублей. Чаще всего в программах протокола ARP дублирующие запросы помещаются в очередь. После получения ответа на запрос адресная привязка становится известной. Поэтому программа протокола ARP должна удалить повторные запросы из очереди, поместить каждый пакет во фрейм и, используя полученную адресную привязку, заполнить поле физического адреса получателя фрейма. Если же машина не выдавала запрос для IP-адреса, указанного в полученном ARP-ответе, после обновления элемента кэш-памяти для адресной привязки отправителя, она должна просто прекратить обработку полученного пакета.

## 5.11. Инкапсуляция и идентификация пакетов ARP

При передаче ARP-сообщений<sup>3</sup> от одной машины до другой они помещаются во фреймы физической сети. Как видно из рис. 5.2 ARP-сообщения занимают поле данных фрейма.

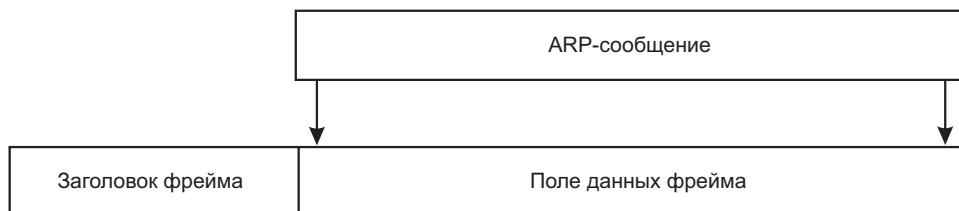


Рис. 5.2. Инкапсуляция ARP-сообщения во фрейм физической сети

<sup>3</sup> Под ARP-сообщениями мы будем понимать пакеты ARP-запросов и ответы на них.

Для идентификации фреймов, содержащих ARP-сообщения, отправитель должен поместить специальное значение в поле типа заголовка фрейма. Напомним, что при получении фрейма сетевое программное обеспечение использует значение поля типа для определения содержимого фрейма. В большинстве сетевых технологий для обозначения ARP-сообщений используется унифицированное значение поля типа. Поэтому для распознавания ARP-сообщений, содержащих ARP-запросы и ответы на них, сетевое программное обеспечение должно проанализировать содержимое ARP-пакета. Например, в технологии Ethernet в поле типа фреймов, содержащих ARP-сообщения, проставляется значение 0x0806. Это значение назначено органами, утвердившими стандарт Ethernet. В других сетевых технологиях используются другие значения.

## 5.12. Формат ARP-сообщений

В отличие от большинства других протоколов, данные в ARP-пакетах не имеют фиксированного формата заголовка. Это сделано для того, чтобы приспособить протокол ARP под существующее множество сетевых технологий. Таким образом, длина полей заголовка сообщения ARP, содержащих адреса, будет зависеть от типа используемого сетевого оборудования. Однако, чтобы можно было интерпретировать ARP-сообщения произвольного формата, в начало заголовка помещено несколько фиксированных полей, в которых указывается длина последующих полей адресов. В сущности, формат ARP-сообщений подобран так, чтобы в них можно было использовать любые физические и логические адреса.

В качестве примера на рис. 5.3 показан формат ARP-сообщения размером 28 октетов, которое используется в технологии Ethernet для преобразования логических адресов протокола IP, длина которых составляет 4 октета (напомним, что физический адрес Ethernet имеет длину 48 бит, или 6 октетов). Обратите внимание, что на рис. 5.3 ARP-сообщение разбито на строки, в каждой из которых содержится по 4 октета. Такой формат представлений мы будем использовать на протяжении всей книги. К сожалению, в отличие от большинства других протоколов, в данном случае восприятие диаграммы затруднено из-за использования полей адреса переменной длины, которые не удается точно выровнять на границу 32-х бит. Например, поле, содержащее физический адрес отправителя, занимает подряд шесть октетов. Поэтому на диаграмме оно изображено в две строки.

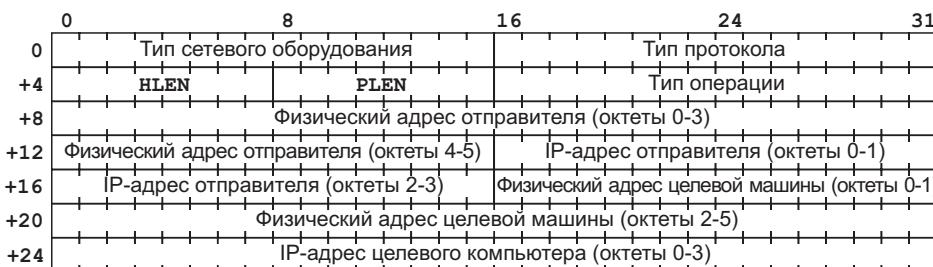


Рис. 5.3. Формат ARP-сообщения, которое используется в сетях Ethernet для преобразования IP-адреса в физический адрес. Длина полей зависит от типа используемого оборудования и длины логического адреса сетевого протокола. В сетях Ethernet физический адрес имеет длину 6 октетов, в IP-адрес — 4 октета

В поле, обозначающем тип сетевого оборудования, указывается тип интерфейса, для которого отправитель посылает запрос. В случае сетей Ethernet его значение равно единице. По аналогии, в поле типа протокола указывается тип протокола высокого уровня, адрес которого помещен отправителем в сообщение.

Для протокола IP его значение равно 0x0800. В поле типа операции указывается код пересылаемого сообщения. Для ARP-запроса его значение равно 1, для ARP-ответа — 2, для RARP-запроса<sup>4</sup> — 3 и для RARP-ответа — 4. Поля HLEN и PLEN позволяют использовать протокол ARP в любых типах сетей, поскольку они определяют, соответственно, длину физического адреса и адреса протокола высокого уровня. Отправитель указывает также свой физический адрес и IP-адрес (в случае если он известен) в соответствующих полях сообщения.

При выполнении запроса отправитель также указывает в сообщении физический адрес целевой машины (в случае протокола RARP) или IP-адрес целевого компьютера (в случае протокола ARP). При получении запроса, удаленный компьютер заполняет недостающие поля ARP-сообщения, меняет местами пары адресных привязок для отправителя и целевого компьютера, устанавливает соответствующее значение поля типа операции и отправляет пакет машине, выдавшей запрос. Таким образом, в ответном сообщении будут находиться как IP-адрес и физический адрес машины, выдавшей запрос, так и пара адресов машины, для которой выполняется адресная привязка.

### 5.13. Резюме

Каждому сетевому интерфейсу компьютера независимо от его физического адреса назначается уникальный IP-адрес. Прежде чем отправить IP-пакет по физической сети от одного компьютера к другому, сетевое программное обеспечение должно преобразовать IP-адрес в физический адрес машины получателя и поместить последний в соответствующее поле фрейма. Фрейм передается только по физическому адресу получателя. В случае если длина поля физического адреса меньше чем поля IP-адреса, можно установить прямое соответствие между физическим адресом компьютера и IP-адресом, закодировав первый во втором. В противном случае преобразование адресов должно выполняться динамически. Протокол ARP (Address Resolution Protocol) как раз и предназначен для такой цели. Для работы ему нужна только реализация коммуникационной системы на самом нижнем уровне. Протокол ARP позволяет компьютерам преобразовывать адреса без необходимости дополнительного сохранения записей о самой привязке. Для определения физического адреса другой машины компьютер должен отправить широковещательный ARP-запрос в сеть, содержащий ее IP-адрес. Данный ARP-запрос будет получен всеми машинами локальной сети, однако ответ на него отправляет только та машина, IP-адрес которой совпадает с указанным в запросе. В ответ на ARP-запрос отправителю посыпается физический адрес интересующей его машины. Ответ посыпается отправителю ARP-запроса напрямую без применения режима широковещательной передачи данных.

Для повышения эффективности работы системы ARP на каждом компьютере сети должна поддерживаться кэш-память, в которую помещаются так называемые адресные привязки (IP-адрес и соответствующий ему физический адрес). Поскольку потоки данных в объединенной сети в основном проходят между двумя взаимодействующими компьютерами, использование кэш-памяти позволяет снизить общее количество широковещательных ARP-запросов.

### Материал для дальнейшего изучения

Протокол ARP, описанный в этой главе, был предложен Пламмером (Plummer) в [RFC 826] и стал одним из стандартов в семействе протоколов TCP/IP. Взаимосвязь адресов Ethernet и IP-адресов раскрывается в работе Далала

<sup>4</sup> Протокол RARP будет описан в следующей главе. В нем используется тот же формат сообщения.

(Dalal) и Принтиса (Printis) [37]. Общие сведения об адресации и адресной привязке можно почерпнуть из весьма туманного документа [RFC 814]. Отказоустойчивая система преобразования адресов описана Парром (Parr) в [RFC 1029]. В [RFC 1166] Киркпатриком (Kirkpatrick) и Реккером (Recker) приведены значения номеров, которые используются для идентификации сетевых фреймов. Этот документ озаглавлен *Internet Numbers*. Во втором томе данной книги приведен пример реализации протокола ARP и обсуждаются стратегии кэширования.

## Упражнения

- 5.1. Предположим, что существует некоторое небольшое множество физических адресов, представляющих собой положительные целые числа. Попытайтесь придумать систему назначения IP-адресов и найти такую функцию  $f$ , чтобы можно было установить однозначное соответствие между физическими адресами и IP-адресами. Не забывайте, что вычисление функции  $f$  должно быть максимально эффективным. (*Подсказка*. Для решения этой задачи обратитесь к литературе, посвященной идеальной системе хеширования).
- 5.2. В каких случаях узлу, подключенному к сети Ethernet, для передачи IP-дейтаграммы не нужно посылать ARP-запрос или искать адресную привязку в кэш-памяти?
- 5.3. Существует один из известных алгоритмов управления содержимым кэш-памяти, который заключается в том, что в случае переполнения кэш-памяти программа помещает новый элемент на место дольше всего неиспользовавшегося элемента. При каких условиях использование этого алгоритма может привести к возникновению избыточного трафика в сети?
- 5.4. Внимательно прочитайте стандарт ARP. Должна ли сетевая программа обновлять содержимое кэш-памяти ARP-запросов в случае, если для заданного IP-адреса в ней уже существует элемент привязки? Поясните свой ответ.
- 5.5. Должна ли сетевая программа обновлять содержимое кэш-памяти ARP-запросов в случае, если она получает ответ на запрос, который не посыпалась? Поясните свой ответ.
- 5.6. Любая реализация сетевой программы протокола ARP, использующая кэш-память фиксированного размера, может перестать работать в сети, к которой подключено множество узлов и существует большой трафик ARP. Объясните почему.
- 5.7. Протокол ARP часто считается одним из слабых мест с точки зрения системы безопасности компьютера. Объясните почему.
- 5.8. Предположим, что в одной из некорректных реализаций протокола ARP, программа вообще не удаляет элементы из кэш-памяти если они часто используются. Объясните, что может произойти, если во время передачи по сети ответа на ARP-запрос будет искажено поле физического адреса машины.
- 5.9. Предположим, машина  $C$  получила ARP-запрос, посланный от машины  $A$  с целью определения физического адреса машины  $B$ . Предположим также, что в кэш-памяти ARP-запросов машины  $C$  существует элемент, содержащий адресную привязку машины  $B$ . Должна ли машина  $C$  ответить на поступивший запрос? Обоснуйте свой ответ.

- 5.10.** Как во время начальной загрузки рабочая станция может определить с помощью протокола ARP наличие в сети другой машины с аналогичным IP-адресом? Каковы недостатки подобного метода?
- 5.11.** Объясните, в каком случае посылка IP-пакетов по несуществующему адресу удаленным узлом Ethernet может привести к избыточному трафику в сети широковещательных запросов.

# 6

## *Определение IP-адреса при начальной загрузке (RARP)*

### **6.1. Введение**

В предыдущих главах уже говорилось о том, что физический сетевой адрес является низкоуровневым и зависит от используемого сетевого оборудования. Поэтому каждой машине, использующей протокол TCP/IP, назначается один или несколько 32-разрядных IP-адресов, которые не зависят от применяемого на компьютере сетевого оборудования и физических адресов. При указании получателя пакета прикладные программы всегда пользуются IP-адресом. Поскольку для передачи дейтаграммы по физической сети ее узлу или маршрутизатору должен быть известен физический адрес получателя, необходим механизм преобразования адресов наподобие протокола ARP, который бы позволял устанавливать однозначное соответствие между IP-адресами и эквивалентными им физическими адресами.

Обычно IP-адрес компьютера хранится на одном из внешних запоминающих устройств; при начальной загрузке операционная система находит его и загружает в оперативную память. При этом может возникнуть вопрос: как может определить IP-адрес компьютер, к которому не подключены никакие запоминающие устройства? Эта проблема особенно актуальна для тех рабочих станций, файлы которых расположены на удаленном сетевом сервере, а также для небольших встроенных систем, поскольку для получения данных начальной загрузки с помощью стандартного протокола пересылки файлов FTP система должна “знать” свой IP-адрес. В этой главе рассматриваются вопросы получения IP-адреса и описывается низкоуровневый протокол, который могут использовать системы для определения своего IP-адреса перед выполнением начальной загрузки с удаленного файлового сервера. Описание методов начальной загрузки будет продолжено в главе 23, “Начальная загрузка и автоконфигурация (BOOTP, DHCP)”, там же будут рассмотрены широко используемые альтернативы протоколу, о котором пойдет речь в этой главе.

Обычно разработчики программного обеспечения всячески стараются избегать жесткой привязки кода к определенному IP-адресу, поскольку такой код нельзя запускать на нескольких компьютерах локальной сети. Особенно это касается кода начальной загрузки операционной системы и служебных программ. В частности, код программы начальной загрузки компьютера, обычно хранящийся в ПЗУ (постоянном запоминающем устройстве), проектируется так, чтобы можно было использовать одну его копию на многих компьютерах. Как только этому коду во время начальной загрузки компьютера передается управление, он связывается по сети с сервером и узнает у него IP-адрес компьютера.

Описание процедуры начальной загрузки звучит весьма странно. Действительно, чтобы определить IP-адрес, необходимый для проведения сеансов связи с другими машинами, нужно сначала связаться с удаленным сервером. Однако никакой странности здесь нет, поскольку при начальной загрузке компьютер “знает”, как осуществить сеанс связи. Для связи по физической сети компьютер может использовать свой физический адрес. А это значит, что в процессе начальной загрузки компьютер должен временно воспользоваться физической системой адресации конкретной сети. Здесь можно провести аналогию с тем, как при инициализации таблиц системы виртуальной памяти операционная система пользуется физическими адресами памяти. Как только компьютеру станет известен его IP-адрес, он может взаимодействовать с любой машиной объединенной сети.

Идея, лежащая в основе поиска IP-адреса очень проста: компьютер, который хочет узнать свой IP-адрес, посыпает специальный запрос *серверу*<sup>1</sup>, запущенному на другом компьютере, и ожидает от него ответа. Само собой разумеется, что сервер имеет доступ к диску, на котором хранится база данных IP-адресов. В запросе компьютер, который хочет узнать свой IP-адрес, должен каким-то образом однозначно идентифицировать себя. По этим данным сервер может провести поиск IP-адреса в базе данных и отправить ответ. Во время короткого сеанса связи как компьютер, посылающий запрос, так и сервер, отвечающий на него, используют физическую систему адресации сети. Как же отправитель запроса узнает физический адрес сервера? Как правило, — никак. Он просто посыпает широковещательный запрос всем машинам локальной сети, на который отвечает один или несколько серверов.

Выше уже было сказано, что при отправке запроса на определение своего IP-адреса, компьютер должен каким-то образом однозначно идентифицировать себя. Какая же информация должна включаться в запрос, чтобы компьютер однозначно идентифицировался? Здесь достаточно любых уникальных данных, например серийного номера центрального процессора. Однако следует учитывать тот факт, что получение идентификационных данных не должно вызывать проблем у программы начальной загрузки. К сожалению, длина и формат идентификационной информации может существенно отличаться для разных моделей процессоров, тогда как сервер должен принимать запросы от всех машин, подключенных к физической сети, в едином формате. Более того, разработчики стараются создать универсальный код начальной загрузки, который бы мог выполняться на процессоре любого типа (речь, конечно же, идет о совместимых моделях, таких как процессоры семейства 80x86 фирмы Intel — Прим. ред.). При этом для каждой модели процессора предусматривается специальный набор команд для получения серийного номера.

## 6.2. Протокол обратного преобразования адресов RARP

Создатели семейства протоколов TCP/IP прекрасно понимали, что в компьютере, оснащенном сетевой платой, есть еще один универсальный источник получения идентифицирующей информации. Речь идет о физическом адресе сетевой платы. Его использование для идентификации компьютера имеет два преимущества. Во-первых, поскольку программа считывает физический адрес прямо с платы сетевого интерфейса, подобная идентификационная информация будет всегда доступна и ее не нужно каким-то образом генерировать в коде начальной загрузки. Во-вторых, идентификационная информация зависит от типа используемого сетевого оборудования и при этом никак не зависит от типа

---

<sup>1</sup> Описанию серверов посвящена глава 21, “Модель взаимодействия клиент/сервер”.

центрального процессора и его модели. Таким образом, все машины в конкретной сети могут предоставить серверу однотипную уникальную информацию. Поэтому основная проблема заключается в методах выполнения обратного преобразования адресов. Другими словами, необходимо разработать алгоритм, который бы позволял серверу по известному физическому адресу находить соответствующий ему IP-адрес.

В семействе протоколов TCP/IP существует специальный протокол, позволяющий компьютеру запрашивать у сервера свой IP-адрес. Он называется *протоколом обратного преобразования адресов* (*Reverse Address Resolution Protocol*, или *RARP*). Этот протокол является подмножеством протокола ARP, описанного в предыдущей главе. В нем используется тот же формат сообщений (см. рис. 5.3). На самом деле сфера использования RARP-сообщений немного шире, чем было описано выше. Они позволяют компьютеру с легкостью запросить не только свой IP-адрес, но и IP-адрес любой другой машины локальной сети. Кроме того, их можно использовать в физических сетях разных типов.

Подобно сообщению ARP, RARP-сообщение при отправке от одной машины до другой помещается в поле данных сетевого фрейма. Например, при пересылке RARP-сообщения во фрейме Ethernet, вначале передается заголовок фрейма, состоящий из стандартной преамбулы, адресов Ethernet отправителя и получателя и типа пакета. Для идентификации RARP-сообщения в поле типа фрейма помещается значение 0x8035. Поле данных фрейма, содержащее RARP-сообщение, имеет длину 28 октетов.

На рис. 6.1 показан процесс использования узлом сети сообщений RARP. Отправитель посыпает в широковещательном режиме RARP-запрос, в котором уникальным образом идентифицирует себя, помещая соответствующие одинаковые данные в поля логического сетевого адреса как отправителя, так и целевого компьютера. Кроме того, в поле физического адреса целевого компьютера он указывает свой физический адрес. Запрос RARP принимают все компьютеры локальной сети, но ответ на него могут отправить только специально уполномоченные машины, называемые *RARP-серверами*. Поэтому, чтобы получить ответ на RARP-запрос, в сети должен работать по крайней мере один RARP-сервер.

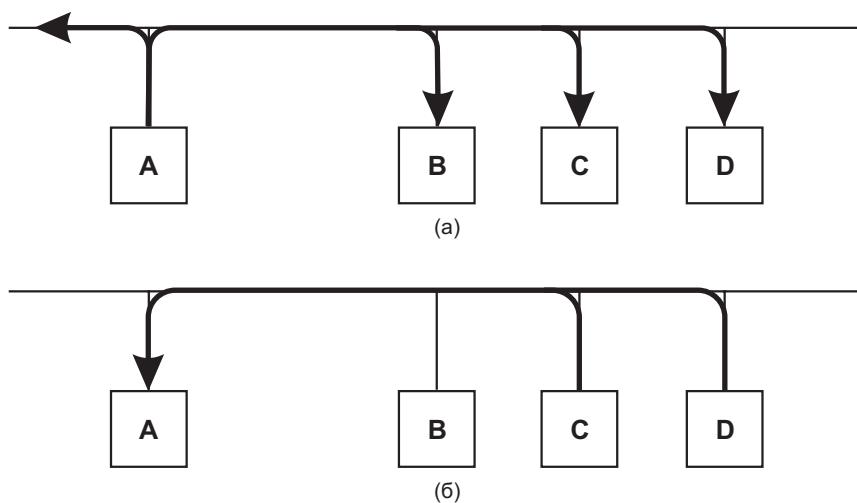


Рис. 6.1. Схема работы протокола RARP. Машине А посыпает в сеть широковещательный RARP-запрос, в котором она уникальным образом идентифицирует себя (а). Ответ на данный запрос напрямую машине А посыпают только специально уполномоченные для этой цели машины С и D (б)

При обработке RARP-запроса серверы помещают IP-адрес клиента в поле логического адреса целевого компьютера, изменяют тип сообщения (с *запроса на ответ*) и посылают ответ напрямую той машине, от которой был получен запрос. Обратите внимание, что клиенту посылаются ответы от всех RARP-серверов, даже если для работы достаточно только одного из них.

Следует иметь в виду, что взаимодействие между компьютером, пытающимся определить свой IP-адрес, и сервером, который этот адрес сообщает, происходит исключительно по одной физической сети. Кроме того, протоколом RARP предусмотрена возможность получения узлом сети IP-адреса любого другого узла этой физической сети. Поэтому в сообщении RARP предусмотрены отдельные поля для физического адреса отправителя и физического адреса целевого компьютера. Сервер всегда посылает ответ по адресу, записанному в поле физического адреса отправителя. В сетях Ethernet наличие в RARP-сообщении отдельного поля физического адреса отправителя может показаться излишним, поскольку такая же информация содержится в заголовке фрейма Ethernet. Однако следует заметить, что не во всех типах сетей Ethernet программы операционной системы могут получить доступ к заголовку физического фрейма.

### 6.3. Время выполнения транзакции RARP

Подобно другим пакетам, посылаемым по сетевой системе, не гарантирующей их доставку, RARP-запросы и ответы на них могут потеряться по пути следования. Например, их может отвергнуть сетевая плата, если при анализе кода циклической избыточной проверки (CRC) фрейма окажется, что он получен с ошибкой. Поскольку в протоколе RARP обмен информацией происходит непосредственно по физической сети, не существует другого протокола более низкого уровня, программа которого отслеживала бы временные задержки и посыпала бы в случае необходимости повторные запросы. Все эти операции должны выполняться программой поддержки протокола RARP самостоятельно. Вообще говоря, протокол RARP используется только в локальных сетях наподобие Ethernet, где вероятность потери пакета очень мала. Тем не менее, если в сети существует только один RARP-сервер и он по какой-либо причине оказывается перегружен, пакеты могут не достигнуть получателя за отведенное для этой цели время либо будут потеряны.

Существует несколько способов решения указанной проблемы. В одних системах, процесс загрузки которых завязан на протокол RARP, отправка запросов выполняется до тех пор, пока хотя бы на один из них не будет получен ответ. В других системах, чтобы избежать ненужного широковещательного трафика в сети, уже после нескольких неудачных попыток фиксируется состояние отказа сервера (как в случае, когда сервер отключен от сети). В сетях Ethernet вероятность отказа или сбоя оборудования менее вероятна, чем ситуация перегрузки сервера. Поэтому частый повтор запросов RARP может вызвать нежелательный эффект лавинообразной перегрузки и без этого сильно загруженного сервера. В то же время увеличение интервала между повтором запросов дает серверу достаточно времени на обработку поступившего запроса и возврат результата.

### 6.4. Основной и резервный RARP-серверы

Для повышения надежности системы в целом в одной сети может функционировать несколько серверов RARP, запущенных на отдельных компьютерах. Тогда, если один из серверов выходит из строя или сильно перегружен, ответ на посланный клиентом запрос будут присыпать другие серверы. В результате

повышается вероятность того, что служба RARP будет доступна всегда. Однако у такого подхода есть и существенный недостаток, поскольку ответ на один запрос клиента могут прислать сразу все серверы. В результате при достаточно большом количестве запросов может произойти перегрузка сети. Например, в сети Ether-net, применение нескольких RARP-серверов часто вызывает повышенный уровень коллизий.

Исходя из всего сказанного возникает вопрос: как организовать службу RARP, чтобы она надежно работала, была всегда доступна, и при этом не обладала недостатками, вытекающими из использования нескольких серверов, которые могут одновременно прислать одинаковые ответы? Существует по меньшей мере два способа решения указанной проблемы, однако оба они связаны с увеличением времени задержки ответа на RARP-запрос. В первом случае для каждого компьютера, посылающего RARP-запрос, назначается *основной сервер* (*primary server*). При нормальных условиях ответ на поступивший от конкретной машины RARP-запрос посылает только основной сервер. Все остальные серверы принимают запрос, но не отвечают на него, а только фиксируют время его поступления. Если основной сервер не ответит на запрос, то через заданный промежуток времени клиентский компьютер снова повторит его. Как только резервные серверы в течение короткого промежутка времени получат повторный запрос RARP, они должны послать на него ответ.

Во втором случае используется такой же алгоритм работы, как и в первом случае, но предпринимаются меры, чтобы резервные серверы не посыпали ответ на один и тот же запрос одновременно. После получения запроса каждый резервный RARP-сервер выбирает случайным образом величину интервала времени ожидания, по истечении которого отправляется ответ на запрос. Таким образом, при нормальных условиях один из резервных серверов отправит ответ практически сразу же после получения запроса, а ответы остальных резервных серверов будут задержаны. Благодаря этому снижается вероятность того, что на один запрос будет отправлено несколько ответов одновременно. Таким образом, если основной RARP-сервер выйдет из строя, клиентский компьютер получит ответ на свой запрос с небольшой задержкой. Тщательно подобрав величины задержек, можно добиться того, чтобы клиент получал ответ от одного из резервных серверов без повторной посылки RARP-запроса в сеть.

## 6.5. Резюме

Компьютеры, не укомплектованные внешними запоминающими устройствами, для определения своего IP-адреса в процессе начальной загрузки должны запросить специальный сервер. Только после этого они могут взаимодействовать с другими узлами сети с помощью протокола TCP/IP. В этой главе был описан протокол RARP, который используется для получения рабочими станциями своего IP-адреса. При этом используется физическая система адресации конкретной сети. Для уникальной идентификации машины в сети в протоколе RARP используется физический адрес ее сетевой платы. После того как поля с идентификационной информацией в RARP-сообщении заполнены, оно посылается в сеть в широковещательном режиме специализированным серверам. Получив сообщение, серверы, по идентификационным данным клиента ищут IP-адрес в своих таблицах, которые обычно хранятся во внешней памяти, и отправляют ответ. Получив свой IP-адрес, клиентский компьютер сохраняет его в оперативной памяти для дальнейшего использования. При следующей перезагрузке клиентский компьютер снова посылает запрос RARP.

## **Материал для дальнейшего изучения**

Более подробно протокол RARP описан Финлейсоном (Finlayson) и др. в [RFC 903]. В [RFC 906] Финлейсон описывает процесс начальной загрузки рабочей станции с помощью протокола TFTP. В [RFC 1293] Бредли (Bradley) и Браун (Brown) описывают еще один аналогичный RARP протокол, который называется *инверсным ARP* (*Inverse ARP*). Этот протокол позволяет рабочей станции запрашивать свой IP-адрес у компьютера, находящегося на другом конце сетевого соединения, и предназначен для использования в сетях с установкой соединения, таких как Frame Relay или ATM. Реализация протокола RARP описана во втором томе этой книги.

В главе 23, “Начальная загрузка и автоконфигурация (BOOTP, DHCP)”, описаны два альтернативных протокола: BOOTP и DHCP. В отличие от протокола RARP, в котором используется низкоуровневый механизм адресации, в протоколах BOOTP и DHCP задействованы сетевые протоколы более высокого уровня, такие как IP и UDP. В главе 23 проведено также сравнение двух разных подходов к решению проблемы получения IP-адреса при начальной загрузке, отмечены достоинства и недостатки каждого из них.

## **Упражнения**

- 6.1.** RARP-сервер может рассылать ответы на RARP-запросы всем машинам локальной сети либо посыпать ответ только той машине, которая выдала соответствующий запрос. В какой из сетевых технологий оправдано применение режима широковещания для отправки ответов на RARP-запросы?
- 6.2.** Один из недостатков протокола RARP состоит в его узкой направленности, т.е. в том, что он позволяет компьютеру получить по запросу только одну единицу информации (свой IP-адрес). При начальной загрузке, кроме IP-адреса, компьютеру нужно также определить свое имя. Расширьте спецификацию протокола RARP так, чтобы с его помощью можно было получить дополнительную информацию.
- 6.3.** Насколько увеличится размер фрейма Ethernet, если в ответ на запрос RARP поместить дополнительную информацию, как было описано в предыдущем упражнении?
- 6.4.** Очевидно, что запуск в сети второго сервера RARP повышает надежность работы протокола RARP. Имеет ли смысл запустить третий сервер RARP? А четвертый? Обоснуйте свой ответ.
- 6.5.** Предположим, что бездисковая рабочая станция, выпущенная некоторым производителем, использует протокол RARP для определения своего IP-адреса. При этом подразумевается, что ответ на RARP-запрос должен приходить от файлового сервера этой рабочей станции. Предположим также, что после получения IP-адреса, рабочая станция пытается получить с этого сервера файл с программой начальной загрузки. Если сервер не ответит на запрос, рабочая станция входит в бесконечный цикл, в котором периодически будет посыпаться запрос на получение кода программы начальной загрузки. Опишите ситуацию, когда введение в строй резервного сервера RARP в описанном выше случае может привести к перегрузке сети широковещательными запросами. (Подсказка. Рассмотрите случай пропадания напряжения питания.)

- 6.6.** Понаблюдайте за пакетами, передающими по вашей локальной сети в процессе начальной загрузки нескольких компьютеров. Определите, какие из компьютеров используют протокол RARP.
- 6.7.** В этой главе речь шла о том, что резервные RARP-серверы используют факт посылки в течение короткого промежутка времени повторного RARP-запроса как признак того, что нужно отправить на него ответ. Рассмотрите систему взаимодействия RARP-серверов, при которой все серверы должны отвечать на первый же запрос, однако при этом во избежание перегрузок в сети каждый сервер будет посыпать ответ по истечении случайно выбранного промежутка времени. При каких условиях подобная схема взаимодействия серверов будет давать лучшие результаты, чем та, что описана в этой главе?



# 7

## Протокол IP: доставка дейтаграмм без установки соединения

### 7.1. Введение

В предыдущих главах были рассмотрены сетевое оборудование и программное обеспечение, обеспечивающие межсетевое взаимодействие. Кроме того, были описаны низкоуровневые сетевые технологии и преобразование адресов. В этой главе будет рассмотрен основополагающий принцип доставки дейтаграмм без установки соединения, используемый в *межсетевом протоколе (Internet Protocol, или IP)*. Этот протокол является одним из двух основных протоколов, применяемых в процессе межсетевого взаимодействия (вторым протоколом является TCP). Ниже будет рассмотрен формат дейтаграмм и показано, как на их основе осуществляются все межсетевые взаимодействия. В последующих двух главах рассмотрение протокола IP будет продолжено. Там речь пойдет о маршрутизации дейтаграмм и обработке ошибок.

### 7.2. Виртуальная сеть

В главе 3, “Основы и структура межсетевого взаимодействия”, была рассмотрена структура объединенной сети, состоящей из нескольких физических сетей, соединенных между собой посредством маршрутизаторов. Рассмотрение структуры сети не должно вводить вас в заблуждение, поскольку основное внимание уделяется не технологиям межсетевого взаимодействия, а интерфейсу, который предоставляет объединенная сеть для пользователей.

*С точки зрения пользователей объединенная сеть представляет собой единую виртуальную сеть, соединяющую между собой все компьютеры. Благодаря этому можно легко осуществить взаимодействие между любыми двумя машинами объединенной сети. При этом внутренняя структура физических сетей не имеет значения — она скрыта от пользователя.*

В каком-то смысле, объединенная сеть является обобщенной физической сетью, поскольку на самом нижнем уровне она обеспечивает те же функциональные возможности — прием пакетов и доставку их по назначению. На ее основе межсетевым программным обеспечением более высокого уровня создаются развитые службы, используемые пользователями.

### 7.3. Структура и принцип действия объединенной сети

Объединенную сеть на основе протокола TCP/IP можно разделить на три логических уровня, как показано на рис. 7.1. Положение уровней на рисунке соответствует существующим между ними взаимосвязям. На самом нижнем уровне находится служба доставки пакетов, не требующая установки соединения. Она является своего рода фундаментом, на котором построены все остальные сетевые службы. На следующем уровне располагается надежная транспортная служба, которая обеспечивает для прикладных программ высокуровневый механизм передачи данных. Каждая из этих служб будет описана чуть ниже; мы рассмотрим выполняемые ими функции и используемые при этом протоколы.

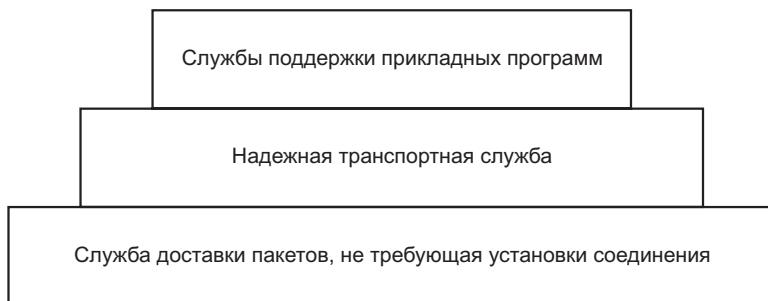


Рис. 7.1. Три логических уровня служб объединенной сети

### 7.4. Принцип организации служб

Каждой службе, показанной на рис. 7.1, можно сопоставить соответствующее протокольное программное обеспечение. Однако мы разбили весь спектр служб, предоставляемых объединенной сетью, на три логические составляющие части, поскольку такой подход позволяет прояснить основополагающие принципы, лежащие в основе проектирования объединенной сети.

*Программное обеспечение для объединенной сети разрабатывалось с учетом его принадлежности к одной из трех логических сетевых служб, организованных в иерархическом порядке. Его успешное применение в большей степени обусловлено тем, что такая логическая структура является необычайно надежной и гибкой.*

Одним из существенных преимуществ описанной логической структуры является возможность замены одной службы без влияния на все остальные службы. Таким образом, выполнение исследований и разработка новых сетевых служб может выполняться параллельно, независимо от логического уровня, который занимает служба в иерархической структуре.

### 7.5. Служба доставки, не требующая установки соединения

Служба доставки пакетов является одной из основных в объединенной сети. Формально, ее можно охарактеризовать как ненадежную, не гарантирующую доставку пакетов, и не требующую предварительной установки соединения с получателем. Рассматриваемую службу можно сравнить с сетевой аппаратной

технологией, такой как Ethernet, предпринимающей максимум усилий для доставки пакетов, но не гарантирующей их доставку получателю. Служба названа *ненадежной*, поскольку она не гарантирует доставку пакетов получателю. Это означает, что по пути следования пакет может быть утерян, продублирован, задержан или доставлен с нарушением порядка следования. При этом служба доставки не сможет обнаружить перечисленные выше проблемы и сообщить о них отправителю и получателю. Служба доставки *не требует установки соединения*, поскольку каждый пакет считается независимым от остальных. Последовательности пакетов, передаваемые между двумя компьютерами, могут проходить по разным маршрутам. При этом некоторые пакеты могут быть утеряны и не достигнуть компьютера получателя. И наконец, можно сказать, что рассматриваемая нами служба не обеспечивает каких-либо усилий для доставки пакетов, поскольку межсетевое программное обеспечение предпринимает для этого эффективные меры. Другими словами, в объединенной сети пакеты просто так не пропадают. Надежность системы нарушается только в случае перегрузки какого-либо сетевого ресурса или повреждении участка сети.

## 7.6. Назначение протокола IP

Протокол, в котором используется ненадежный, не требующий установки соединения с получателем механизм доставки, назвали *межсетевым протоколом*, или *Internet Protocol*. Однако чаще всего для обозначения этого протокола используют аббревиатуру *IP*<sup>1</sup>. В протоколе IP сделано три важных определения. Во-первых, в нем определен базовый элемент передачи данных, используемый во всей объединенной сети на основе протокола TCP/IP. Это означает, что в протоколе IP четко определен формат передаваемых по объединенной сети данных. Во-вторых, программы поддержки протокола IP выполняют функцию *маршрутизации*, которая заключается в выборе пути, по которому будут посыпаться данные. В-третьих, помимо точной, формальной спецификации форматов данных и функций маршрутизации, в протокол IP включен набор правил, воплощающих в жизнь идею ненадежной доставки пакетов данных. В этих правилах оговариваются способы обработки пакетов узлами сети и маршрутизаторами, а также условия, при возникновении которых должны генерироваться сообщения об ошибке и удаляться пакеты. Протокол IP стал неотъемлемой частью структуры всей объединенной сети, причем настолько, что сети, в которых используется протокол TCP/IP, иногда называют сетями на основе *IP-технологии*.

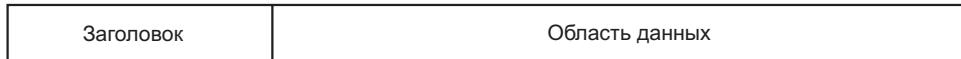
В этой главе мы начинаем рассмотрение протокола IP с описания формата используемых пакетов. В следующих главах мы обсудим вопросы маршрутизации и обработки ошибок.

## 7.7. Межсетевая дейтаграмма

Между физической сетью и объединенной сетью на основе протокола TCP/IP имеется большое сходство. В физической сети единицей передачи информации является фрейм, который состоит из заголовка и блока данных. В заголовке фрейма содержится служебная информация, а также физические адреса отправителя и получателя. В объединенной сети единицей передачи информации является *межсетевая дейтаграмма* (*Internet datagram*), которую иногда называют *IP-дейтаграммой*, или просто *дейтаграммой*. Как и обычный сетевой фрейм, дейтаграмма состоит из заголовка и блока данных. По аналогии с фреймом, в заголовке дейтаграммы указываются адреса отправителя и получателя пакета, а

<sup>1</sup> От аббревиатуры IP происходит термин *IP-адрес*.

также поле типа, которое позволяет определить ее содержимое. Различие состоит в том, что в заголовке дейтаграммы находятся IP-адреса, а в заголовке фрейма — физические адреса. Общий вид дейтаграммы представлен на рис. 7.2.



*Рис. 7.2. Общий вид IP-дейтаграммы, которая является аналогом сетевого фрейма. В протоколе IP оговаривается формат ее заголовка, содержащий адреса отправителя и получателя. Формат области данных не оговаривается, поэтому в дейтаграмме можно передавать информацию произвольного объема*

### 7.7.1. Формат дейтаграммы

После описания общего вида IP-дейтаграммы пришло время поговорить о ее содержимом более подробно. Расположение полей дейтаграммы показано на рис. 7.3.



*Рис. 7.3. Формат IP-дейтаграммы — основной единицы передачи данных в объединенной сети на основе протокола TCP/IP*

Поскольку обработкой дейтаграмм занимается специальное протокольное программное обеспечение, их содержимое и формат не должны привязываться к какому бы то ни было сетевому оборудованию. В первом поле заголовка дейтаграммы (его длина составляет 4 бита) указывается номер версии протокола IP, которому соответствует дейтаграмма. Этот номер устанавливается отправителем и используется для проверки согласованности форматов дейтаграмм получателем, а также промежуточными маршрутизаторами, через которые проходят пакеты. Перед обработкой дейтаграммы программное обеспечение протокола IP, установленное на компьютере, должно сверить версию протокола. Эта операция гарантирует соответствие формата дейтаграммы обрабатывающему ее программному обеспечению. В случае изменения формата дейтаграммы изменится и версия протокола IP. При несоответствии версий программного обеспечения и дейтаграммы, обработка последней не производится. Это гарантирует, что данные дейтаграммы не будут некорректно интерпретированы устаревшим программным обеспечением. Текущей версией протокола IP является 4. Поэтому для ее идентификации часто используется аббревиатура IPv4.

Поле, в котором содержится длина заголовка дейтаграммы, выраженная в 32-разрядных словах, также имеет длину 4 бита. Как вы увидите в дальнейшем, все поля заголовка имеют фиксированную длину, кроме поля, в котором указываются параметры протокола IP, и соответствующего ему поля выравнивания. Длина типового заголовка IP-дейтаграммы, не содержащей дополнительных параметров и поэтому не имеющей поля выравнивания, составляет 20 октетов. При этом в поле длины заголовка дейтаграммы заносится цифра 5.

Как следует из названия, в поле *общей длины дейтаграммы* заносится общий размер дейтаграммы (включая заголовок и область данных), выраженный в октетах. Размер области данных можно легко вычислить, вычтя из общей длины дейтаграммы размер заголовка, предварительно выразив его в октетах. Поскольку длина поля, в которое помещается длина дейтаграммы, составляет 16 бит, максимальный размер IP-дейтаграммы составляет  $2^{16}-1$  или 65 535 октетов. Для большинства приложений это ограничение не является слишком жестким. Однако, учитывая наметившуюся в последнее время тенденцию к повышению пропускной способности каналов связи, в будущем вполне могут появиться высокоскоростные сетевые технологии, в которых размер пакета данных превысит 65 535 октетов.

### 7.7.2. Способы обработки дейтаграммы и схемы дифференцированного обслуживания

В поле типа обслуживания (чаще всего его называют *тиром сервиса*, или *TOS*), размер которого составляет 8 бит, указывается способ обработки дейтаграммы. Это поле первоначально было разделено на пять подполей, как показано на рис. 7.4.

0	1	2	3	4	5	6	7
Приоритет	D	T	R	Не используется			

Рис. 7.4. Оригинальная разбивка на пять подполей поля типа обслуживания

В поле приоритета, длина которого составляет три бита, указывается важность дейтаграммы. Значение 0 соответствует обычной дейтаграмме, а 7 — сетевому управляющему пакету. Отправитель может также установить в этом поле любые промежуточные значения, соответствующие важности каждой посылаемой дейтаграммы. И хотя некоторые маршрутизаторы игнорируют значение этого поля, оно имеет важное значение, поскольку позволяет установить приоритет управляющих сетевых пакетов, над обычными пакетами данных. Например, многие маршрутизаторы при обмене служебной информацией помещают в поле приоритета значение 6 или 7. Это позволяет им обмениваться пакетами даже в том случае, когда сеть сильно перегружена.

Биты *D*, *T*, и *R* определяют желательный тип транспортировки дейтаграммы. При установке бита *D* дейтаграмма должна быть передана с минимальной задержкой. При установленном бите *T* дейтаграмму нужно передавать по каналу с высокой пропускной способностью, а бит *R* — говорит о том, что доставка дейтаграммы должна быть максимально надежной. Естественно, что в объединенной сети не всегда возможно обеспечить запрашиваемый дейтаграммой тип транспортировки (например, может оказаться, что нельзя проложить к получателю маршрут, удовлетворяющий некоторым требованиям). Поэтому указанные биты следует рассматривать только как рекомендации при выполнении алгоритма маршрутизации, а не как обязательное требование. Если существует несколько маршрутов следования пакетов к получателю, то при их выборе маршрутизатор может руководствоваться описанными битами, чтобы обеспечить требуемый тип транспортировки пакетов. Например, предположим, что маршрутизатор может отправлять пакеты получателю с малым временем задержки, но по каналу с низкой пропускной способностью (по наземной выделенной линии связи) и с большой задержкой — по высокоскоростному спутниковому каналу связи. Тогда в дейтаграммах, транспортирующих данные,

поступающие на удаленный компьютер от клавиатуры пользователя, следует установить бит  $D$ . В результате они будут доставлены получателю с минимальной задержкой. А для дейтаграмм, транспортирующих большой объем данных из файла, следует установить бит  $T$ . В результате они будут передаваться по высокоскоростному спутниковому каналу связи. При этом время задержки передачи пакетов не имеет существенного значения.

В конце 1990-х годов инженерной группой IETF была изменена первоначальная раскладка 8-битового поля типа обслуживания, как показано на рис. 7.5. Это позволило реализовать различные схемы *дифференцированного обслуживания* (*differentiated services, DS*).

0	1	2	3	4	5	6	7
Код указателя						Не используется	

Рис. 7.5. Интерпретация поля типа обслуживания IP-дейтаграммы для реализации различных схем дифференцированного обслуживания (DS)

В новой интерпретации в первых шести битах хранится *код указателя* (*codepoint*) службы, который иногда обозначается аббревиатурой *DSCP* (*differentiated services codepoint*), а последние два бита не используются. С помощью кода указателя программа протокола может осуществить поиск нужной службы, как правило через массив указателей. И хотя в поле размером 6 бит можно закодировать информацию о 64 отдельных службах, разработчики протокола, сочли нужным ограничить это число, поскольку на реально работающих маршрутизаторах задействовано всего несколько типов служб. В результате разные коды указателя могут соответствовать одной и той же службе. Более того, для совместимости с первоначальной раскладкой рассматриваемого нами поля в стандарте предусмотрены специальные различия для трех младших (раньше в них помещалось значение приоритета) и трех старших битов кода указателя. Если в трех старших битах кода содержатся нули, то три младших бита (приоритета) определяют восемь больших классов служб, соответствующих тем же основополагающим принципам, которым отвечала первоначальная раскладка 8-битового поля типа обслуживания. Речь идет о приоритете дейтаграмм, у которых в трех младших битах поля указателя содержится большее значение. Таким образом, для определения восьми упорядоченных классов используются коды указателя в формате  $xxx00$ , где  $x$  может быть как нулем, так и единицей.

В новой раскладке поля типа обслуживания учтен также другой часто используемый случай, когда для маршрутизации трафика используется значение приоритета 6 или 7. В стандарте отдельно оговорен метод обработки этих значений приоритета. Таким образом, в маршрутизаторе должно быть предусмотрено по меньшей мере два алгоритма приоритетного обслуживания: один для обычного трафика, а другой — для трафика с высоким приоритетом. Если три старших бита поля кода указателя равны нулю, маршрутизатор должен так настроить свои таблицы указателей, чтобы кодам 6 и 7 соответствовали классы служб с наивысшим приоритетом, а все остальным кодам — классы служб с более низким приоритетом. Таким образом, если будет получена дейтаграмма, созданная с помощью старой версии сетевого программного обеспечения (с первоначальной раскладкой поля типа обслуживания), маршрутизатор, используя систему дифференцированных классов служб, обработает значения приоритета 6 и 7 так, как и было задумано отправителем.

Все возможные значения кодов указателя (их может быть 64) разбиты на 3 административные группы, как показано в табл. 7.1.

---

**Таблица 7.1. Три административных группы значений кодов указателя**

---

Группа	Код указателя*	Описание
1	xxxxx0	Стандартизовано IETF
2	xxxx11	Для локального или экспериментального использования
3	xxxx01	В настоящее время доступны для локального или экспериментального использования

---

\*) Примечание. Здесь используется принятый в сети порядок битов “слева направо”, т.е. младший бит находится слева.

Как следует из табл. 7.1, процесс интерпретации половины значений кодов указателя (т.е. 32, относящихся к первой группе) стандартизован инженерной группой IETF. В настоящее время все значения, относящиеся к группам 2 и 3, пока доступны для локального или экспериментального использования. Однако следует отметить, что как только значения кодов из первой группы, назначаемые центральным органом, будут исчерпаны, выделение кодов может быть продолжено уже из третьей группы.

Таким образом, для определения номера группы необходимо проанализировать два старших бита кода указателя. Описанный принцип разбиения на группы был выбран для того, чтобы величины кодов указателя, соответствующие шаблону xxx000, находились в одной группе.

Независимо от того, какая раскладка поля типа обслуживания используется (оригинальная TOS или новая — с классами дифференцированного обслуживания) важно понимать, что алгоритмы маршрутизации могут выбирать способ доставки дейтаграмм только в том случае, если к получателю существует несколько маршрутов с разными характеристиками, которые в свою очередь зависят от применяемого сетевого оборудования и установленных локальных правил. Таким образом, запрошенный в дейтаграмме уровень обслуживания еще не означает, что все маршрутизаторы, расположенные по пути следования пакета смогут его удовлетворить. Подводя итоги, можно сказать следующее.

*Спецификацию типа обслуживания дейтаграммы следует рассматривать только как рекомендацию при выполнении алгоритма маршрутизации, а не как обязательное требование. В случае, если к конечному получателю существует несколько маршрутов с разными характеристиками, спецификация позволяет выбрать тот из них, который наиболее полно удовлетворяет выдвинутым требованиям. Следует иметь в виду, что эти требования могут вступать в противоречие с применяемым сетевым оборудованием и установленными локальными правилами. В объединенной сети невозможно гарантировать выполнение какого бы то ни было обслуживания заданного типа.*

### 7.7.3. Инкапсуляция дейтаграмм

Прежде чем перейти к рассмотрению следующих полей дейтаграммы, необходимо уяснить, как связаны между собой дейтаграммы и сетевые фреймы. Сразу же возникает вопрос: каков максимальный размер дейтаграммы? В отличие от фреймов физической сети, формат которых зависит от используемого сетевого оборудования, обработка дейтаграмм выполняется исключительно программными средствами. Поэтому, теоретически, их длина может быть любой, точнее такой, какой ее выбрали разработчики протокола. Выше мы уже говорили, что в

дейтаграмме стандарта IPv4 под поле, содержащее размер дейтаграммы, было выделено 16 бит. Таким образом, максимальная длина дейтаграммы не может превышать 65535 октетов.

Однако на практике на размер дейтаграмм налагаются более существенные ограничения. Вы уже знаете, что передача дейтаграмм от одного компьютера к другому всегда выполняется средствами физической сети. Поэтому, для повышения эффективности работы механизма транспортировки объединенной сети необходимо иметь гарантию, что каждая дейтаграмма поместится в один физический фрейм. Другими словами, нужно по возможности сделать так, чтобы существовала прямая связь между принятым абстрактным сетевым пакетом (дейтаграммой) и фреймом физической сети.

Метод транспортировки одной дейтаграммы в одной сетевом фрейме называется *инкапсуляцией* (*encapsulation*). С точки зрения физической сети, дейтаграмма ничем не отличается от любого другого сообщения, посылаемого от одной машины до другой. Для сетевого оборудования формат дейтаграммы не имеет никакого значения, точно так же, как и указанный в ней IP-адрес получателя. Таким образом, как показано на рис. 7.6, при передаче IP-дейтаграммы от одного компьютера до другого, она целиком помещается в область данных сетевого фрейма<sup>2</sup>.



Рис. 7.6. Инкапсуляция IP-дейтаграммы в сетевом фрейме. С точки зрения физической сети, дейтаграмма вместе со своим заголовком, рассматривается как обычные данные

#### 7.7.4. Размеры дейтаграммы, MTU и процесс фрагментации

В идеальном случае IP-дейтаграмма должна целиком помещаться в одном сетевом фрейме, что позволяет передать ее по сети с наибольшей эффективностью. Поэтому для достижения подобной эффективности разработчики протокола IP должны были выбрать максимальный размер дейтаграммы, который бы позволял всегда помещать ее в один фрейм. Однако здесь возникает вопрос: каким должен быть этот размер? Кроме того, не следует забывать, что в процессе передачи от отправителя к конечному получателю, дейтаграмма может проходить по физическим сетям разного типа.

Чтобы до конца понять суть проблемы, вспомним как работает сетевое оборудование. В любой сетевой технологии с коммутацией пакетов существует такое понятие, как максимальный размер данных, которые могут быть переданы в одном физическом фрейме. Например, в технологии Ethernet это значение равняется 1500 октетам<sup>3</sup>, а в технологии FDDI в одном фрейме может находиться

<sup>2</sup> Как правило, в заголовке сетевого фрейма предусмотрено специальное поле, предназначенное для идентификации находящихся в нем данных. Например, в сети Ethernet для идентификации инкапсулированной IP-дейтаграммы, помещенной в область данных сетевого фрейма, в поле типа фрейма записывается значение 0x0800.

<sup>3</sup> Ограничение в 1500 октетов взято из стандарта Ethernet. При использовании стандарта IEEE 802.3 и заголовка SNAP размер поля данных ограничивается 1492 октетами. При ис-

примерно 4470 октетов данных. Для обозначения подобных ограничений используют специальный термин — *максимальная единица передачи данных в сети*, или *MTU (maximum transfer unit)*. Величина MTU может быть достаточно маленькой. В некоторых сетях значение MTU составляет порядка 128 октетов и даже меньше. Таким образом, ограничение размеров дейтаграммы размерами минимального MTU, существующего в объединенной сети, приводит к падению эффективности передачи данных на тех участках, где сетевое оборудование позволяет передавать фреймы большего размера. В то же время, если размер дейтаграммы будет больше, чем размер минимального MTU в объединенной сети, то это означает, что на некоторых участках сети она может не поместиться в один сетевой фрейм.

Решение проблемы должно быть уже очевидным. Вспомним, что ключевым моментом при создании объединенной сети было удобство работы с ней. Поэтому разработчики сделали так, чтобы особенности используемых сетевых технологий были скрыты от пользователя. Они не стали создавать какой-то особый формат дейтаграммы, который бы удовлетворял всем существующим ограничениям для физических сетей. Вместо этого был выбран исходный размер дейтаграммы, удобный с точки зрения программного обеспечения протокола TCP/IP, а также оговорен способ разделения больших дейтаграмм на части в том случае, если дейтаграмма должна передаваться по физической сети с малым значением MTU. Части, на которые разделяется дейтаграмма, называются *фрагментами*, а сам процесс разделения дейтаграммы — *фрагментацией*.

Как показано на рис. 7.7, фрагментация обычно выполняется на одном из промежуточных маршрутизаторов, расположенных по пути следования дейтаграммы от ее отправителя до конечного получателя. В данном случае маршрутизатор получил дейтаграмму из сети с большим значением MTU и должен ее передать по сети, где значение MTU меньше, чем размер дейтаграммы.

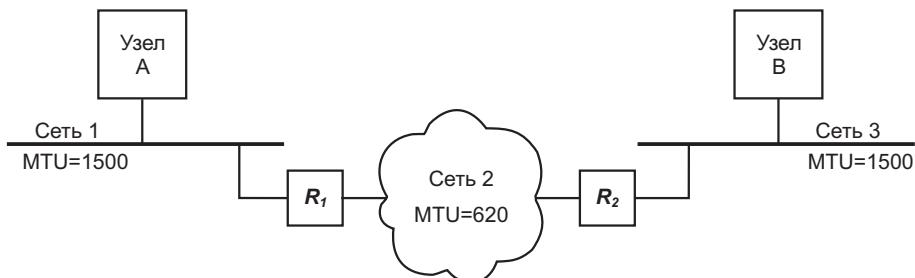


Рис. 7.7. Пример, иллюстрирующий процесс выполнения фрагментации в сети. Маршрутизатор  $R_1$  выполняет фрагментацию больших дейтаграмм, посланных от узла  $A$  до узла  $B$ ; маршрутизатор  $R_2$  выполняет фрагментацию больших дейтаграмм, посланных от узла  $B$  до узла  $A$ .

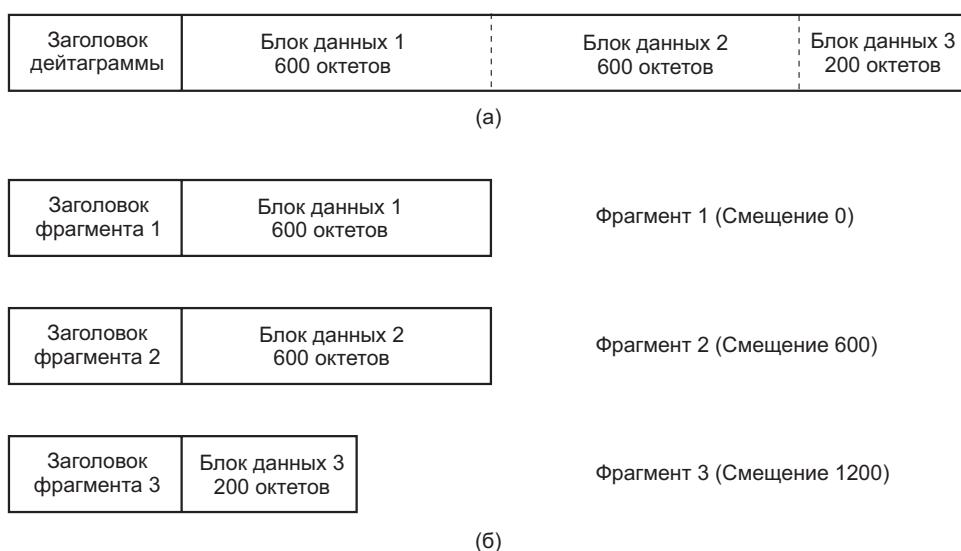
На рис. 7.7 оба узла подключены к сети Ethernet, в которой размер MTU составляет 1500 октетов. Это значит, что оба узла могут отправлять дейтаграммы, максимальный размер которых составляет 1500 октетов. Однако по пути от узла  $A$  до узла  $B$  пакеты попадают в промежуточную сеть, в которой размер MTU составляет 620 октетов. Если узел  $A$  посылает узлу  $B$  дейтаграмму, размер которой больше 620 октетов, маршрутизатор  $R_1$  должен выполнить фрагментацию. Точно так же, если узел  $B$  посылает большую дейтаграмму узлу  $A$ , фрагментацию дейтаграмм выполняет маршрутизатор  $R_2$ .

---

пользовании сетевого оборудования Ethernet, изготовленного некоторыми независимыми разработчиками, размер области данных может быть несколько большим.

Размер фрагмента выбирается так, чтобы он целиком помещался в одном сетевом фрейме. Кроме того, так как в протоколе IP величина смещения фрагмента от начала заголовка кратна 8 октетам, размер фрагмента также выбирается кратным 8 октетам. Само собой разумеется, что выбор фрагмента, размер которого приблизительно равен MTU участка сети и при этом кратен 8 октетам, приводит к делению дейтаграмм на неравные части. Обычно размер последней части намного меньше остальных. Перед тем, как получатель сможет обработать фрагментированную дейтаграмму, он должен получить все ее части и произвести *сборку* (т.е. восстановить ее первоначальный вид).

В протоколе IP нет никаких рекомендаций по поводу минимального размера дейтаграммы. Кроме того, нет никакой гарантии, что большая дейтаграмма будет доставлена получателю без фрагментации. Таким образом, получатель может выбирать размер дейтаграммы по своему усмотрению; ее фрагментация и последующая сборка выполняются автоматически на уровне протокола IP, без какого-либо вмешательства отправляющей стороны. В спецификации протокола IP указывается, что маршрутизаторы должны принимать дейтаграммы, размер которых не превосходит MTU сетей, к которым они подключены. Кроме того, маршрутизаторы должны всегда обрабатывать дейтаграммы размером до 576 октетов. От узлов сети также требуется, чтобы они принимали и при необходимости восстанавливали дейтаграммы, размером как минимум 576 октетов.



*Рис. 7.8. Схематическое изображение исходной дейтаграммы, содержащей 1400 октетов данных (а); та же дейтаграмма, разделенная на три фрагмента данных для сети с MTU, равным 620. В заголовках фрагментов 1 и 2 устанавливается специальный бит (*more fragments*), означающий, что существует следующий по порядку фрагмент. Смещения фрагментов выражены в октетах в десятичной системе счисления. Для определения значения, помечаемого в заголовок фрагмента, значение смещения нужно разделить на 8*

В процессе фрагментации дейтаграмма делится на части. Вас может удивить, что каждый фрагмент имеет такой же формат, как и исходная дейтаграмма. Результат фрагментации показан на рис. 7.8.

Как видно из рис. 7.8, у каждого фрагмента существует свой заголовок, информация в котором дублирует информацию из заголовка исходной дейтаграммы

(кроме одного бита в поле флагов, обозначающего фрагмент). За заголовком фрагмента следует поле данных. Его размер вместе с заголовком не должен превышать размера MTU той сети, по которой пересыпается фрагмент.

### 7.7.5. Сборка фрагментов

В этом разделе будет рассмотрен интересный вопрос, касающийся сборки дейтаграмм. Нужно ли собирать фрагменты дейтаграмм сразу после прохождения критичного участка сети или передать их по назначению и поручить сборку получателю? В объединенной сети TCP/IP после фрагментации дейтаграмм каждый фрагмент передается конечному получателю как самостоятельная дейтаграмма. Сборкой фрагментов занимается конечный получатель. Однако такой подход имеет два недостатка. Во-первых, поскольку дейтаграммы не собираются сразу после прохождения участка сети с малым размером MTU, конечному получателю могут отравляться фрагменты небольшого размера. Таким образом, сборка дейтаграмм конечным получателем приводит к снижению эффективности функционирования объединенной сети в случае, если после точки фрагментации пакеты будут проходить по сетям с большим значением MTU. Во-вторых, если хотя бы один из фрагментов потерян, конечному получателю не удастся собрать дейтаграмму. После доставки получателю первого фрагмента дейтаграммы запускается специальный *таймер сборки (reassemble timer)*. Если значение таймера истекает до того, как получены все фрагменты дейтаграммы, получатель не обрабатывает дейтаграмму и удаляет полученные фрагменты. То есть фрагментация существенно повышает вероятность потери целой дейтаграммы, поскольку потеря любого фрагмента автоматически означает потерю всей дейтаграммы.

Однако несмотря на описанные недостатки, процесс сборки дейтаграмм получателем работает очень хорошо. Кроме того, такой подход позволяет выполнять маршрутизацию каждого фрагмента независимо от остальных фрагментов дейтаграммы. При этом промежуточные маршрутизаторы не должны сохранять фрагменты дейтаграммы с целью их последующей сборки.

### 7.7.6. Управление фрагментацией

В заголовке дейтаграммы предусмотрены три поля, управляющие процессом ее фрагментации и последующей сборки: идентификационные данные, флагги и смещение фрагмента. В поле идентификационных данных указывается уникальное целое число, предназначенное для идентификации текущей дейтаграммы. Напомним, что при фрагментации маршрутизатор копирует информацию из большинства полей заголовка дейтаграммы в аналогичные поля заголовков фрагментов. Поэтому поле идентификационных данных также будет скопировано. Оно служит для того, чтобы получатель смог установить, какой из дейтаграмм принадлежат поступившие фрагменты. При получении очередного фрагмента получатель пытается идентифицировать дейтаграмму, используя предназначеннное для этой цели поле и адрес отправителя. При отправке IP-дейтаграммы компьютер должен занести в поле идентификационных данных ее заголовка уникальное число<sup>4</sup>. Один из методов получения уникального идентификатора, который используется программами протокола IP, заключается в организации в оперативной памяти глобального счетчика отправляемых пакетов.

---

<sup>4</sup> Теоретически, при повторной передаче пакета в поле его идентификационных данных может быть помещено такое же значение, что и у оригинала. Однако на практике, выполнением повторной передачи пакета занимаются протоколы более высокого уровня. В результате каждый повторный пакет будет размещаться в новой дейтаграмме, содержащей в заголовке уникальное число.

При создании каждой новой дейтаграммы счетчик увеличивается на единицу, и полученное значение помещается в поле идентификационных данных.

Напомним, что формат каждого из фрагментов практически совпадает с форматом целой дейтаграммы. В заголовке каждого фрагмента в соответствующем поле указывается смещение, по которому расположены переносимые им данные, относительно исходной дейтаграммы. Для первого фрагмента смещение будет равно нулю. С целью экономии места в заголовке фрагмента, смещение выражается не в октетах, а в относительных единицах. Каждая единица равна 8 октетам. Перед сборкой дейтаграммы, компьютер должен получить по сети все ее фрагменты и упорядочить их по значению смещения, начиная с нулевого и заканчивая самым максимальным. При этом фрагменты не обязательно должны приходить в том порядке, как это необходимо для сборки. Не забывайте, что маршрутизатор, выполнивший фрагментацию дейтаграммы, и ее получатель, собирающий фрагменты, не обмениваются между собой никакой информацией по этому поводу.

Два младших бита поля флагов (его размер составляет 3 бита) предназначены для управления процессом фрагментации. Обычно, процесс фрагментации не является прерогативой прикладного программного обеспечения, использующего протокол TCP/IP. Все дело в том, что фрагментация и последующая сборка дейтаграмм выполняются автоматически протоколами нижнего уровня, включенными в операционную систему, и поэтому скрыты от конечного пользователя. Тем не менее при тестировании сетевого программного обеспечения либо при устранении возникших проблем, может понадобиться определить размер фрагментируемой дейтаграммы. Первый (по номеру, а не по порядку) управляющий бит как раз и предназначен для целей отладки и указывает, можно ли фрагментировать дейтаграмму. Установка этого бита в единицу говорит маршрутизатору о том, что фрагментацию данной дейтаграммы выполнять не нужно. Прикладные программы могут отменить режим фрагментации только в том случае, если для нормального функционирования какой-либо системы требуется дейтаграмма целиком. В качестве примера рассмотрим процедуру начальной загрузки небольшой встроенной системы, программа управления которой защищена в ПЗУ. Система отправляет по объединенной сети запрос серверу, который в ответ присылает образ памяти, содержащий программу начальной загрузки. Обычно из-за ограниченного объема ПЗУ программы управления встроенными системами поддерживают только элементарные сетевые операции. Поэтому они не умеют собирать воедино фрагментированные дейтаграммы. Следовательно, отправляя ответ, сервер должен установить бит *запрета фрагментации*. Если, получив такую дейтаграмму, маршрутизатор не сможет передать ее дальше без фрагментации, он посыпает отправителю сообщение об ошибке и удаляет дейтаграмму из очереди на отправку.

Младший (нулевой) бит поля флагов предназначен для идентификации фрагментов дейтаграммы, т.е. указывает, является ли этот фрагмент последним или за ним следуют еще фрагменты. Этот бит называется *More fragments* (дополнительные фрагменты). Чтобы продемонстрировать, зачем нужен этот бит, давайте рассмотрим работу программного обеспечения протокола IP на компьютере конечного получателя, которое пытается собрать фрагментированную дейтаграмму. Фрагменты дейтаграммы доставляются конечному получателю по очереди, причем необязательно по порядку. Поэтому получателю нужно определить момент, когда ему будут доставлены все фрагменты. Вспомним, что в заголовке дейтаграммы предусмотрено поле, в котором указывается ее общая длина. Однако, при получении фрагмента, в данном поле будет находиться не общая длина дейтаграммы, а только длина текущего фрагмента. Поэтому, используя только описываемое поле, конечный получатель не сможет определить момент, когда

ему будут доставлены все фрагменты дейтаграммы. Для решения проблемы и предназначен бит, указывающий на наличие последующих фрагментов. Получив фрагмент, у которого этот бит сброшен, получатель знает, что ему доставлен последний фрагмент дейтаграммы. Просуммировав значения, находящиеся в его поле смещения и поле общей длины, получатель легко сможет вычислить длину исходной дейтаграммы. Проанализировав аналогичные поля всех доставленных фрагментов, получатель сможет определить, доставлены ли все необходимые для сборки дейтаграммы фрагменты.

### 7.7.7. Время жизни дейтаграммы (TTL)

В принципе, в поле *времени жизни* заголовка дейтаграммы должен указываться интервал времени в секундах, в течении которого дейтаграмме разрешено находиться в объединенной сети. Эта идея проста и в то же время очень важна. Суть ее состоит в том, что, запуская дейтаграмму в объединенную сеть, компьютер устанавливает для нее максимально допустимый интервал времени, в течение которого дейтаграмма должна достичь получателя. Проходя по объединенной сети от одного узла к другому, дейтаграмма последовательно подвергается обработке разными маршрутизаторами или обычным компьютерами. После выполнения обработки каждый компьютер уменьшает значение времени жизни дейтаграммы и передает ее на обработку другому узлу. Если при этом окажется, что время жизни исчерпано, обрабатывающий дейтаграмму узел должен удалить ее из объединенной сети.

Проблема заключается в том, что маршрутизатор не может точно узнать время, прошедшее с момента запуска дейтаграммы в сеть, поскольку нельзя определить время передачи дейтаграммы по разным физическим сетям без синхронизации часов на всех маршрутизаторах объединенной сети. Последнее требование выполнить довольно сложно, учитывая масштабы объединенной сети. Поэтому был принят ряд правил, упрощающих обработку дейтаграмм и не требующих синхронизации часов. Во-первых, от каждого маршрутизатора, находящегося по пути следования дейтаграммы от отправителя до конечного получателя, требуется, чтобы после ее обработки он уменьшил на единицу значение поля времени жизни. Во-вторых, в случае перегрузки маршрутизатора дейтаграмма может обрабатываться довольно продолжительное время. Поэтому после получения дейтаграммы каждый маршрутизатор должен зафиксировать локальное время ее прибытия, а после обработки дейтаграммы уменьшить значение поля времени жизни на число секунд, которые дейтаграмма провела в маршрутизаторе в ожидании обслуживания<sup>5</sup>.

Как только значение поля времени жизни дейтаграммы станет равно нулю, маршрутизатор должен удалить дейтаграмму из сети, а ее отправителю послать соответствующее сообщение об ошибке. Идея установки времени жизни дейтаграммы интересна еще и тем, что она не позволяет дейтаграмме бесконечно блуждать по сети от одного узла до другого, например, в случае повреждения таблиц маршрутизации или неправильной их настройки, когда маршрутизаторы начинают пересыпать дейтаграммы друг другу или по кругу.

Хотя последнее замечание относительно большого времени обработки дейтаграммы маршрутизатором крайне важно, следует отметить, что оно уже не актуально. Дело в том, что разработчики современных сетевых технологий добились того, чтобы каждая дейтаграмма проходила узлы маршрутизации за приемлемое время. Если же по какой-либо причине задержка при обработке дейтаграммы становится чрезмерной, маршрутизатор попросту удаляет такую дейтаграмму.

<sup>5</sup> На практике современные маршрутизаторы никогда не обрабатывают дейтаграммы по несколько секунд.

Поэтому на практике, значение поля времени жизни можно рассматривать скорее как максимальное число узлов маршрутизации, через которые должна пройти дейтаграмма, а не как время в секундах, за которое дейтаграмма должна достичь получателя. Не забывайте, что каждый маршрутизатор уменьшает его значение на единицу.

### 7.7.8. Остальные поля заголовка дейтаграммы

Поле *типа протокола* является аналогом поля типа сетевого фрейма. По указанному в нем значению можно узнать, к какому типу протокола высокого уровня относится сообщение, помещенное в область данных дейтаграммы. По существу значение поля типа протокола определяет формат области данных дейтаграммы. Соответствие между протоколом высокого уровня и значением, помещенным в поле типа протокола, должно устанавливаться на уровне центрального органа объединенной сети. Только в этом случае можно гарантировать адекватность обслуживания дейтаграммы по всей сети.

Поле *контрольной суммы* заголовка предназначено для проверки целостности полей заголовка дейтаграммы. При подсчете контрольной суммы в протоколе IP заголовок дейтаграммы рассматривается как состоящий из последовательности 16-разрядных целых чисел (с соблюдением сетевого порядка следования байтов). При сложении используется двоичная арифметика с представлением отрицательных чисел в инверсном коде (так называемое сложение с учетом знака). Напомним, что для представления отрицательного числа в инверсном коде следует инвертировать все значения его битов. Полученный результат инвертируется (т.е. его знак меняется на противоположный, чтобы значение контрольной суммы было положительным). При вычислении контрольной суммы всех полей заголовка дейтаграммы, значение поля контрольной суммы полагается равным нулю.

Важно отметить, что контрольная сумма вычисляется только для заголовка IP-дейтаграммы. При этом ее область данных не учитывается. Такой подход имеет как преимущества, так и недостатки. К преимуществам можно отнести скорость вычисления контрольной суммы маршрутизаторами, поскольку длина заголовка составляет всего несколько десятков октетов. Кроме того, это позволяет протоколам более высокого уровня использовать собственные методы вычисления контрольных сумм для инкапсулированных данных. Основной недостаток такого подхода заключается в том, что он заставляет другие протоколы более высокого уровня вычислять собственные значения контрольных сумм, иначе снижается вероятность обнаружения поврежденной дейтаграммы.

Как следует из названия, в полях IP-адреса отправителя и получателя указываются 32-х разрядные адреса отправителя дейтаграммы и ее *конечного получателя*. Хотя при выполнении маршрутизации дейтаграмма может проходить через множество промежуточных узлов, значение полей адресов отправителя и получателя никогда не изменяются. В них всегда содержится исходный адрес отправителя и адрес конечного получателя<sup>6</sup>.

В поле *области данных* (см. рис. 7.3), помещается переносимое дейтаграммой сообщение. Его длина, конечно же, зависит от типа пересылаемых в дейтаграмме данных. Поле параметров протокола IP имеет переменную длину. Длина поля выравнивания зависит от того, какие параметры протокола IP установлены. Оно содержит нулевые биты, количество которых выбирается так, чтобы длина заголовка дейтаграммы была кратна 32 битам. (Напомним, что в поле длины заголовка дейтаграммы помещается значение, выраженное в 32-х разрядных словах.)

<sup>6</sup> За исключением случаев, когда в дейтаграмме содержится параметр, определяющий маршрутизацию от источника, о котором речь пойдет ниже.

## 7.8. Параметры IP-дейтаграммы

За полем IP-адреса получателя следует поле параметров протокола IP, однако оно присутствует далеко не в каждой дейтаграмме. В основном параметры протокола IP помещаются в дейтаграммы только в целях тестирования и отладки. Тем не менее обработка параметров является составляющей частью протокола IP, поэтому она должна быть включена во все его стандартные реализации.

Длина рассматриваемого поля зависит от выбранных параметров. Для некоторых параметров длина этого поля составляет один октет. В него помещается так называемый *код параметра* (*option code*). Другие параметры имеют переменную длину. При включении в дейтаграмму параметров, они следуют один за другим без специальных разделителей. Каждый параметр состоит из одного октета кода, за которым может следовать один октет, в котором указывается длина расположенных следом данных. Октет кода параметра состоит из трех полей (рис. 7.9).

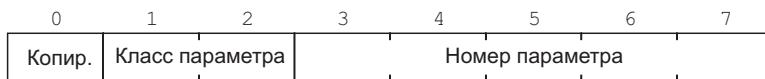


Рис. 7.9. Октет кода параметра поделен на три поля, длина которых составляет 1, 2 и 5 битов

В первом поле, размер которого составляет 1 бит, находится флаг копирования. Во втором поле (его размер составляет 2 бита) указывается класс параметра, а в третьем поле размером 5 битов — номер параметра. Флаг копирования предназначен для управления фрагментацией дейтаграммы, выполняемой маршрутизаторами. Когда его значение равно 1, маршрутизаторы переносят поле параметров во все фрагменты дейтаграммы. Если же значение флага равно нулю, поле параметров переносится не во все, а только в первый фрагмент дейтаграммы.

В следующих двух полях (*класс параметра* и *номер параметра*) указываются общий класс параметров и конкретный параметр в заданном классе. Распределение классов параметров протокола IP приведено в табл. 7.2.

Таблица 7.2. Распределение классов параметров протокола IP

Класс параметра	Описание
0	Управление дейтаграммой или сетью
1	Зарезервировано для дальнейшего использования
2	Для отладочных целей и измерения параметров сети
3	Зарезервировано для дальнейшего использования

В табл. 7.3 приведены значения классов и номера параметров, которые могут встречаться в IP-дейтаграмме. Как видно, большинство параметров используются для управляющих и служебных целей.

**Таблица 7.3. Значения классов и номера параметров IP-дейтаграммы**

Класс параметра	Номер параметра	Длина	Описание
0	0	—	Конец списка параметров. Используется в случае, если конец списка параметров не совпадает с концом заголовка дейтаграммы (см. описание поля выравнивания IP-дейтаграммы)
0	1	—	Холостая операция. Используется для выравнивания октетов в списке параметров
0	2	11	Управление системами безопасности и ограничений (используется в приложениях, применяемых в военных целях)
0	3	Перемен-ная	Нестрогая маршрутизация от источника. Используется для запроса на выполнение маршрутизации через указанные узлы
0	7	Перемен-ная	Регистрация маршрута следования. Используется для трассировки маршрута
0	8	4	Идентификатор потока. Используется для передачи идентификатора потока SATNET (устарело)
0	9	Перемен-ная	Строгая маршрутизация от источника. Используется для указания точного маршрута дейтаграммы
0	11	4	Запрос на MTU. Используется для определения значения MTU в сетях, по которым проходит маршрут пакета
0	12	4	Ответ MTU. Используется для определения значения MTU в сетях, по которым проходит маршрут пакета
0	20	4	Сообщение, привлекающее внимание маршрутизатора. Маршрутизатор должен проанализировать данную дейтаграмму даже в том случае, если он не является ее получателем
2	4	Перемен-ная	Межсетевые метки времени. Используется для регистрации временных меток на пути следования дейтаграммы
2	18	Перемен-ная	Трассировка маршрута. Используется программой traceroute для поиска маршрутизаторов на пути следования дейтаграммы

### 7.8.1. Параметры регистрации маршрута следования

Одними из самых интересных являются параметры маршрутизации и регистрации временных меток, поскольку они позволяют проконтролировать правильность передачи дейтаграмм в объединенной сети. Параметр *регистрации маршрута* позволяет отправителю создать пустой список IP-адресов и “попросить” каждый маршрутизатор, обрабатывающий дейтаграмму, поместить свой IP-адрес в этот список. Формат параметра регистрации маршрута приведен на рис. 7.10.

В октете кода параметра, который был описан выше, помещаются значения класса и номера параметра (для регистрации маршрута следования они, соответственно, равны 0 и 7). В поле длины, следующее за октетом кода, помещается значение общей длины поля параметра, которое он занимает в заголовке дейтаграммы (с учетом первых трех октетов). Сразу за тремя служебными октетами располагается область, зарезервированная для записи IP-адресов. Первый IP-адрес помещается в начало этой области (рис. 7.10), следом за ним — второй и т.д. В поле ука-

зателя помещается смещение следующего свободного элемента для записи IP-адреса, выраженное в октетах относительно начала текущего параметра.

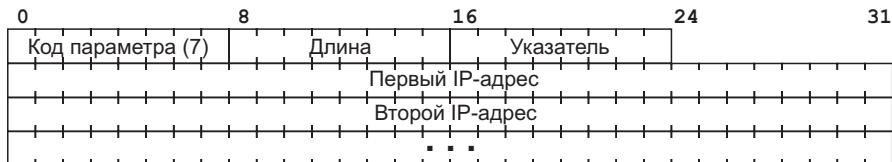


Рис. 7.10. Формат параметра регистрации маршрута IP-дейтаграммы. В начале поля параметров располагается три служебных октета, сразу за которыми помещается список адресов. Несмотря на то, что на рисунке формат параметра изображен в виде 32-разрядных блоков, на самом деле в заголовке дейтаграммы они не выравниваются на границу 32-х бит

Получив дейтаграмму, в заголовке которой задан параметр регистрации маршрута следования, компьютер должен добавить свой адрес в конец списка IP-адресов узлов маршрутизации. При этом отправитель подобной дейтаграммы должен предусмотреть в поле параметра регистрации достаточно места для размещения всех элементов списка. Перед добавлением нового элемента в список компьютер должен сравнить значения полей указателя и длины. Если величина указателя превосходит значение поля длины, значит, в области, выделенной под размещение списка IP-адресов, больше нет свободного места. В таком случае компьютер пересыпает дейтаграмму дальше без помещения в ее заголовок своего IP-адреса. Если же в заголовке дейтаграммы есть место, компьютер помещает в него свой 4-октетный IP-адрес. Место расположения нового элемента списка определяется значением поля указателя. После добавления нового элемента в список значение указателя увеличивается на 4.

После того как дейтаграмма будет доставлена конечному получателю, содержимое ее списка адресов узлов маршрутизации может быть легко извлечено из заголовка и обработано. Обычно после получения дейтаграммы этот список игнорируется. Поэтому использование параметра регистрации маршрута должно приводить к взаимодействию двух компьютеров — отправителя и конечного получателя. Причина состоит в том, что список узлов маршрутизации не посыпается автоматически отправителю дейтаграммы, в которой установлен параметр регистрации маршрута следования. Поэтому, прежде чем посыпать такую дейтаграмму, отправитель должен быть уверен, что ее получатель обработает результирующий список узлов маршрутизации и вернет его обратно.

### 7.8.2. Параметры маршрутизации от источника

Еще одной интересной идеей, реализованной разработчиками в протоколе IP, является возможность задания параметров *исходного маршрута* (*source route*) дейтаграммы, или маршрутизации от источника. Идея заключается в том, что отправитель дейтаграммы может указать в ее заголовке путь, по которому она должна пройти в объединенной сети. Например, эта возможность часто используется сетевыми администраторами для определения пропускной способности некоторой физической сети в случае, если при нормальных условиях маршрутизаторы не пересыпают через нее дейтаграммы. Возможность проведения подобных тестов на реально действующем оборудовании особенно важна в производственных условиях. При этом системный администратор может быть уверен, что при тестировании одного участка сети дейтаграммы пользователей будут пересыпаться по другому, уже протестированному участку. Естественно, что установка параметров маршрута дейтаграммы может быть интересна только тем се-

тевым администраторам, которые хорошо представляют топологию сети. Для обычного пользователя описанная в этом разделе возможность не представляет никакой ценности.

В протоколе IP предусмотрена поддержка двух форм маршрутизации от источника. Одна из них называется *строгой* маршрутизацией (*strict source routing*). Суть ее заключается в том, что в заголовке дейтаграммы перечисляются IP-адреса узлов сети, через которые дейтаграмма должна обязательно пройти. Формат описываемого параметра показан на рис. 7.11.

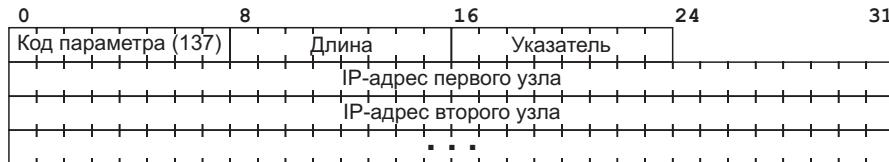


Рис. 7.11. В параметре строгой маршрутизации указывается список IP-адресов узлов, через которые дейтаграмма должна обязательно пройти

Строгая маршрутизация от источника означает, что в заголовке дейтаграммы указан точный путь, по которому она должна пройти до конечного получателя. Следует учитывать, что маршрут между двумя соседними IP-адресами, указанными в списке, должен проходить по одной физической сети. Если маршрутизатор не сможет отправить дейтаграмму по указанному маршруту, отправителю посыпается сообщение об ошибке.

Существует и другая форма маршрутизации от источника, называемая *нестрогой* (*loose source routing*). Суть ее состоит в том, что в заголовке дейтаграммы также указывается список адресов узлов, через которые должна пройти дейтаграмма. Только, в отличие от строгой маршрутизации, путь между соседними адресами, указанными в списке, не обязательно должен проходить по одой физической сети. То есть между двумя адресами узлов маршрутизации, указанными в списке, может находиться любое количество других маршрутизаторов.

Согласно протоколу, от устройств, выполняющих как строгую, так и нестрогую маршрутизацию, требуется, чтобы они заменяли IP-адреса, указанные в списке узлов маршрутизации дейтаграммы своими локальными сетевыми адресами. Таким образом, после доставки дейтаграммы получателю в ее заголовке будет содержаться список адресов узлов объединенной сети (так же, как и при использовании параметра регистрации маршрута), через которую она прошла.

Формат параметров маршрутизации от источника практически повторяет формат параметра, использовавшегося для регистрации маршрута пакетов, о котором речь шла выше. Перед обработкой очередного элемента списка IP-адресов каждый маршрутизатор должен проанализировать значения полей указателя и длины и убедиться, что список элементов еще не исчерпан. Если окажется, что значение указателя больше значения поля длины, маршрутизатор направляет дейтаграмму к получателю обычным образом. Если же список не исчерпан, маршрутизатор, опираясь на значение указателя, выбирает очередной элемент с IP-адресом, заменяет его своим адресом и отправляет дейтаграмму по адресу узла, взятому из списка<sup>7</sup>.

<sup>7</sup> Не забывайте, что каждому сетевому интерфейсу маршрутизатора назначается отдельный адрес. Поэтому в список заносится адрес, соответствующий сети, по которой передается дейтаграмма.

### 7.8.3. Параметр регистрации временных меток

Параметр *регистрации временных меток* во многом напоминает параметр регистрации маршрута следования дейтаграммы, о котором говорилось выше. Отправитель создает пустой список определенного размера, элементы в который добавляются промежуточными маршрутизаторами, расположенными по пути следования дейтаграммы. Каждый элемент в списке состоит из двух 32-разрядных слов: IP-адреса маршрутизатора, созавшего элемент, и целого числа, содержащего временную метку (рис. 7.12).

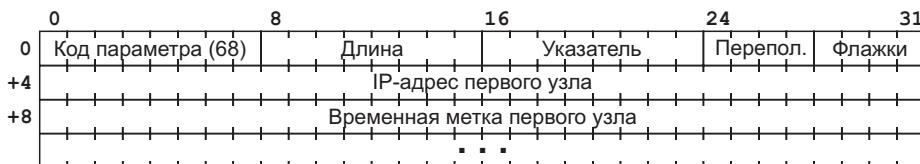


Рис. 7.12. Формат параметра регистрации временных меток. Точный формат этого параметра и правила, по которым он будет обрабатываться маршрутизаторами зависят от состояния битов флагов

Как показано на рис. 7.12, в поле длины указывается размер области параметра, зарезервированной для размещения элементов списка, а в поле указателя хранится смещение следующего свободного элемента (по аналогии с параметром регистрации маршрута, рассмотренным выше). В 4-битовом поле переполнения хранится целочисленный счетчик, указывающий на количество маршрутизаторов, которые не смогли занести в список свою временную метку из-за того, что размер выделенной области параметра оказался слишком мал. Значение 4-битового поля флагов определяет точный формат описываемого параметра и тип временной метки, которую должны поместить в список маршрутизаторы (табл. 7.4).

**Таблица 7.4. Значения поля флагов параметра регистрации временных меток**

Значение	Описание
0	Регистрировать только временные метки без записи IP-адресов
1	Помещать перед каждой временной меткой IP-адрес маршрутизатора. Формат этого параметра приведен на рис. 7.12
3	IP-адреса указаны отправителем. Маршрутизатор должен зарегистрировать временную метку только в том случае, если следующий IP-адрес в списке соответствует IP-адресу маршрутизатора

Временные метки представляют собой дату и время обработки дейтаграммы маршрутизатором, выраженные количеством миллисекунд, прошедших после полуночи по всемирному времени (Universal Time)<sup>8</sup>. Если по каким-либо причинам невозможно использовать представление стандартного времени, маршрутизатор может использовать любую форму представления местного времени. При этом он должен установить в единицу старший бит поля временной метки. Естественно, что временные метки, занесенные в список различными компьютерами, будут не всегда согласованными, даже если для их представления используется всемирное время. Причина в том, что каждый компьютер

<sup>8</sup> Всемирное время соответствует времени нулевого (Гринвичского) меридиана.

вычисляет значение временной метки, основываясь на показаниях своих локальных часов, которые могут отличаться от показаний часов другого компьютера. Поэтому временные метки следует рассматривать как приблизительные оценки, независимо от формата их представления.

Может показаться странным, что в параметре регистрации временных меток предусмотрена возможность записи маршрутизаторами своих IP-адресов. Выше уже шла речь о том, что аналогичную возможность обеспечивает параметр регистрации маршрута дейтаграммы. Однако запись IP-адресов вместе с регистрацией временных меток позволяет избежать неоднозначности. Кроме того, IP-адреса позволяют получателю определить не только временные параметры прохождения дейтаграммы, но и точный путь, по которому она была доставлена.

#### 7.8.4. Обработка параметров при фрагментации дейтаграммы

Идея применения бита копирования, расположенного в октете кода параметра, теперь уже должна быть понятна (см. рис. 7.9). При выполнении фрагментации дейтаграммы маршрутизатор может продублировать некоторые ее параметры во всех фрагментах, а часть параметров оставить только в первом фрагменте. В качестве примера рассмотрим параметр регистрации маршрута следования дейтаграммы. Выше уже было сказано, что каждый фрагмент дейтаграммы обрабатывается независимо от других. Следовательно, нет никакой гарантии, что все фрагменты достигнут конечного получателя по одному и тому же маршруту. Если параметр регистрации маршрута будет находиться во всех фрагментах дейтаграммы, то для одной дейтаграммы конечный получатель сможет выделить несколько разных списков IP-адресов узлов маршрутизации. В результате будет трудно понять, по какому маршруту прошла собранная дейтаграмма. Поэтому в стандарте IP указано, что параметр регистрации маршрута дейтаграммы должен быть скопирован только в один из ее фрагментов.

Однако не все параметры IP-дейтаграммы должны находиться только в одном фрагменте. В качестве примера давайте рассмотрим параметр маршрутизации от источника, который позволяет указать маршрут дейтаграммы в объединенной сети. Нетрудно догадаться, что этот параметр должен быть продублирован во все фрагменты дейтаграммы. В противном случае часть фрагментов может быть доставлена по другому маршруту, отличному от указанного. Таким образом, в октете кода параметра маршрутизации от источника всегда должен быть установлен бит копирования.

### 7.9. Резюме

В основе семейства протоколов TCP/IP, применяемых в объединенной сети, лежит служба доставки дейтаграмм, которую можно охарактеризовать как ненадежную, не гарантирующую доставку пакетов и не требующую предварительной установки соединения с получателем. В протоколе IP четко определен формат передаваемых по объединенной сети пакетов данных, называемых *дейтаграммами*, и воплощена в жизнь идея доставки пакетов, не требующая установки соединения. В этой главе были рассмотрены различные форматы дейтаграмм. В следующих главах речь пойдет о методах их маршрутизации и обработки ошибочных ситуаций.

Как и фрейм физической сети, IP-дейтаграмма состоит из заголовка и области данных. Среди прочей информации в заголовке дейтаграммы указываются IP-адреса отправителя и конечного получателя, биты управления фрагментацией, приоритет и контрольная сумма полей заголовка, используемая для обнаружения ошибок при передаче данных. Кроме полей фиксированной

длины, в каждой дейтаграмме может присутствовать поле параметров. Его длина определяется количеством и типом используемых параметров, а также размерами области данных, выделенной для каждого параметра. Параметры дейтаграммы предназначены для тестирования и управления работой объединенной сети. Они позволяют отправителю указать маршрут дейтаграммы. Кроме того, при использовании параметров конечный получатель может определить маршрут доставки дейтаграммы, а также временные характеристики ее прохождения по объединенной сети.

## Материал для дальнейшего изучения

С методами возможной реализации протоколов объединенной сети, адресации и маршрутизации можно ознакомиться в работе Постела [104]. В более поздних публикациях, в частности в [RFC 791], Постел описывает стандарт протокола IP. В [RFC 1122] Браден (Braden) уточняет этот стандарт. В [RFC 894] Хорниг (Hornig) описывает стандарт передачи IP-дейтаграмм по сети Ethernet. В [RFC 815] Кларк (Clark) предлагает метод эффективной сборки фрагментов. О преимуществах и недостатках фрагментации дейтаграмм речь идет в работе Кента (Kent) и Могула (Mogul) [78].

В [RFC 2474] Николс (Nichols) и др. описывают метод интерпретации битов типа обслуживания, находящихся в заголовке дейтаграммы, с помощью которого можно реализовать различные схемы дифференцированного обслуживания дейтаграммы. Система дифференцированных классов служб описана Блейком (Blake) и др. в [RFC 2475]. Кроме формата пакетов, стандартизованы также значения большого количества констант, использующихся в сетевых протоколах. За дополнительной информацией по этому поводу обращайтесь к периодическому изданию официальных протоколов Internet (Official Internet Protocols) и RFC.

Существует альтернативный вариант семейства межсетевых протоколов, который называется XNS. Ознакомиться с ним можно, обратившись к материалам фирмы Xerox [134]. В статье Боггса (Boggs) и др. [12] описан протокол PUP (PARC Universal Packet, или универсальный пакет PARC), который относится к абстрактному уровню семейства протоколов XNS и по сути является аналогом IP-дейтаграммы.

## Упражнения

- 7.1. В чем преимущество вычисления контрольной суммы только полей заголовка IP-дейтаграммы? Каковы недостатки этого метода?
- 7.2. Нужно ли при отправке IP-дейтаграммы по сети Ethernet вычислять ее контрольную сумму? Обоснуйте свой ответ.
- 7.3. Определите значение MTU для сетей Frame Relay, Hyperchannel и ATM.
- 7.4. Как по вашему мнению должно соотноситься значение MTU для высокоскоростной локальной сети со значением MTU для глобальной сети?
- 7.5. Докажите, что фрагменты дейтаграммы должны иметь небольшие нестандартные заголовки.
- 7.6. Узнайте, когда последней раз изменилась версия протокола IP. Имеет ли смысл для идентификации формата IP-дейтаграмм использовать номер версии протокола?

- 7.7.** Расширьте ответ на предыдущий вопрос и аргументировано объясните, почему в случае изменения номера версии протокола IP имеет смысл ввести новый тип фрейма, а не декодировать номер версии протокола из дейтаграммы на программном уровне.
- 7.8.** Можете ли вы объяснить, почему для вычисления контрольной суммы заголовка IP-дейтаграммы выбран алгоритм суммирования с учетом знака числа, представленного в инверсном коде, а не коды циклической избыточной проверки?
- 7.9.** В чем преимущество сборки дейтаграммы на машине конечного получателя, по сравнению со сборкой дейтаграммы сразу после прохождения критичного участка сети?
- 7.10.** Определите минимальный размер MTU для сети, допускающей пересылку IP-дейтаграмм, размер поля данных которых составляет по меньшей мере один октет.
- 7.11.** Предположим, что вам поручили реализовать обработку дейтаграмм на аппаратном уровне. Нужно ли будет при этом менять расположение и формат отдельных полей заголовка, чтобы сделать этот процесс более эффективным и простым?
- 7.12.** Если вы имеете доступ к исходным кодам программ, реализующих протокол IP, изучите их и проверьте, будет ли программа отклонять дейтаграммы с устаревшим номером версии.
- 7.13.** Определите размер фрейма сети Ethernet, в котором находится IP-дейтаграмма минимально допустимого размера.
- 7.14.** Как известно, в поле типа обслуживания можно закодировать информацию о 64 различных классах дифференцированного обслуживания. Аргументировано объясните, почему для реальной работы достаточно всего нескольких классов служб (т.е. составьте список всех возможных служб, к которым пользователь должен иметь доступ).
- 7.15.** Новая раскладка поля типа обслуживания IP-дейтаграммы, реализующая различные схемы дифференцированного обслуживания, была выбрана с учетом совместимости с полем приоритета первоначальной раскладки. Объясните, привело ли требование обратной совместимости к большому снижению эффективности программной реализации протокола IP, чем возможное альтернативное решение?

# 8

## Протокол IP: маршрутизация дейтаграмм

### 8.1. Введение

В предыдущих главах уже шла речь о том, что в основе работы всех служб объединенной сети лежит система доставки пакетов, не требующая установки соединения, а также о том, что основной единицей передачи данных в семействе протоколов TCP/IP является IP-дейтаграмма. В этой главе мы продолжим рассмотрение базовой службы доставки пакетов, не требующей установки соединения, и опишем, как маршрутизаторы осуществляют перенаправление дейтаграмм и их доставку до конечного получателя. В главе 7, “Протокол IP: доставка дейтаграмм без установки соединения”, был рассмотрен формат дейтаграммы, который можно считать статичным свойством протокола IP. В этой главе описываются механизмы маршрутизации, относящиеся к рабочим характеристикам данного протокола. В следующей главе мы закончим рассмотрение основных черт протокола IP, описав способы обработки ошибочных ситуаций. В главе 10, “Бесклассовая адресация и подсети (CIDR)”, будут описаны расширения протокола TCP/IP, относящиеся к методам бесклассовой адресации и адресации подсетей. В последующих главах будет показано, как на основе протокола IP создаются протоколы более высокого уровня, обеспечивающие нужный уровень услуг.

### 8.2. Маршрутизация в объединенной сети

В системах с коммутацией пакетов под термином *маршрутизация* (*routing*) понимается процесс выбора пути, по которому должны передаваться пакеты к конечному получателю. Термином *маршрутизатор* (*router*) мы будем называть компьютер, который выбирает маршрут следования дейтаграмм. Маршрутизация выполняется на нескольких уровнях. Например, в рамках глобальной сети, состоящей из нескольких коммутаторов пакетов, соединенных между собой физическими каналами связи, маршрутизация проходящих по ней пакетов выполняется самой сетью. Процесс маршрутизации начинается с момента попадания пакета в глобальную сеть и продолжается до момента его выхода за пределы сети. Причем внутри глобальной сети процесс маршрутизации происходит абсолютно автономно. Другими словами, машины, находящиеся за пределами глобальной сети, не могут повлиять на маршрутизацию проходящих по ней пакетов. Для них эта сеть является черным ящиком, выполняющим доставку пакетов из пункта А в пункт Б.

Напомним, что основная задача, которая ставилась при разработке протокола IP, — создать единую виртуальную сеть, объединяющую в себе множество физи-

ческих сетей, в которой реализованы средства доставки дейтаграмм, не требующие установки соединения с получателем. Поэтому основное внимание при изложении материала будет сосредоточено на процессе *перенаправления IP-дейтаграмм* (*IP forwarding*), который также называется *межсетевой маршрутизацией*, или *IP-маршрутизацией*<sup>1</sup>. Данные, которые используются для выбора пути следования пакетов до конечного получателя, называются *маршрутной информацией*. Процесс IP-маршрутизации в объединенной сети напоминает работу обычной физической сети, поскольку он связан с поиском маршрута, по которому будут передаваться дейтаграммы. Однако в отличие от одной физической сети, в алгоритмах IP-маршрутизации следует учитывать, что доставка дейтаграмм будет проходить по разным физическим сетям.

Процесс маршрутизации в объединенной сети может быть затруднен, особенно в случае использования компьютеров с несколькими сетевыми подключениями. В идеальном случае программы маршрутизации должны анализировать загрузку сети, длину дейтаграммы, а также запрошенный в ее заголовке тип обслуживания и на основе полученных данных выбирать оптимальный маршрут. Однако большинство программ маршрутизации не обладают столь развитой логикой и выбирают маршрут следования дейтаграммы, исходя из ряда сделанных допущений о его минимальной длине.

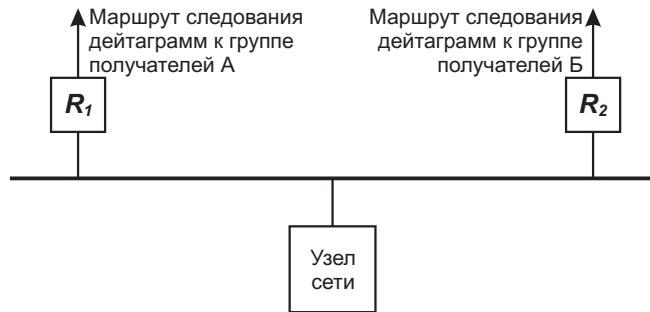
Чтобы до конца разобраться с процессом IP-маршрутизации, мы должны вспомнить структуру объединенной сети, в которой используется протокол TCP/IP. В предыдущих главах уже говорилось о том, что объединенная сеть состоит из нескольких физических сетей, связанных посредством специальных компьютеров, называемых *маршрутизаторами* (*routers*). Каждый маршрутизатор напрямую подключается как минимум к двум сетям. Для сравнения следует сказать, что обычно компьютеры пользователей подключаются к одной физической сети. В предыдущих главах отмечалось, что иногда компьютеры пользователей подключаются сразу к нескольким физическим сетям. Такие компьютеры, или узлы, называются *многоадресными* (*multi-homed host*).

В процессе доставки IP-дейтаграмм до конечного получателя участвуют не только специализированные маршрутизаторы, но и обычные компьютеры, за которыми работают пользователи. Когда одной из прикладных программ, запущенных на компьютере пользователя, необходимо обменяться информацией с удаленным узлом, она обращается к программам протокола TCP/IP. В конечном итоге это приводит к отправке с компьютера пользователя одной или нескольких IP-дейтаграмм. Перед отправкой дейтаграммы компьютер должен просчитать ее начальный маршрут и решить, какому из ближайших узлов маршрутизации она должна быть перенаправлена. Как показано на рис. 8.1, расчет начального маршрута дейтаграммы выполняется даже в том случае, если компьютер имеет только одно физическое подключение к сети.

Основным назначением маршрутизаторов является определение пути следования дейтаграмм и принятие решения об их перенаправлении на один из ближайших узлов маршрутизации. Если с маршрутизаторами все более-менее понятно, то каково же назначение многоадресных узлов? Суть в том, что любой компьютер, имеющий несколько физических сетевых подключений, может выступать в роли маршрутизатора. Как будет показано ниже, любой многоадресный узел, на котором поддерживается протокол TCP/IP, имеет все необходимые для этого средства. Более того, в небольших сетевых центрах часто экономят на выделенных маршрутизаторах и совмещают на одном компьютере функции

<sup>1</sup> В главе 18, “Протокол TCP/IP в сетях ATM”, описана *система коммутации третьего уровня* (*layer 3 switching*), или *IP-коммутация* (*IP-switching*), которая связана с темой этой главы.

маршрутизатора и рабочей станции. Тем не менее в описании стандартов протокола TCP/IP четко разграничены функции узла сети и маршрутизатора. В случае совмещения этих функций на одном компьютере пользователь должен быть готов к тому, что его многоадресный узел будет выполнять, помимо основной, дополнительную работу, причем в самые неожиданные моменты времени. Поэтому мы будем различать узлы сети и маршрутизаторы, предполагая, что узлы сети не могут выполнять функции маршрутизатора по пересылке пакетов из одной сети в другую.



*Рис. 8.1. Выполнение маршрутизации одноадресным узлом. При отправке дейтаграммы узел должен решить, какому из двух маршрутизаторов —  $R_1$  или  $R_2$  — ее следует перенаправить. Проблема состоит в том, что каждый из маршрутизаторов может доставить дейтаграмму по оптимальному маршруту только определенному количеству получателей*

### 8.3. Прямая и непрямая доставка дейтаграмм

Для того чтобы упростить изложение, разделим процесс маршрутизации на две фазы, которые назовем *прямой (direct delivery)* и *непрямой (indirect delivery)* доставкой. Прямая доставка является основой всех межсетевых взаимодействий. Она происходит при передаче дейтаграммы между двумя машинами по одной физической сети. Таким образом, прямая доставка дейтаграмм между двумя компьютерами может осуществляться только в том случае, если они подключены к одной физической системе передачи данных (например, сети Ethernet). *Непрямая* доставка происходит в том случае, когда конечный получатель дейтаграммы находится в другой физической сети. При этом отправитель пересыпает дейтаграмму ближайшему узлу маршрутизации, который выполняет ее дальнейшую доставку до конечного получателя.

#### 8.3.1. Доставка дейтаграмм по одной физической сети

В предыдущих главах уже говорилось о том, что два компьютера могут обмениваться между собой физическими фреймами, если они подключены к одной физической сети. Для передачи IP-дейтаграммы отправитель помещает ее в физический фрейм (инкапсулирует дейтаграмму в физический фрейм), конвертирует IP-адрес получателя в физический адрес и отправляет по нему физический фрейм, используя установленное в компьютере сетевое оборудование. В главе 5, “Преобразование IP-адресов в физические адреса (ARP)”, было описано два возможных механизма преобразования адресов, а также протокол ARP, который используется для динамической привязки адресов в сетях, наподобие Ethernet. Процесс инкапсуляции дейтаграмм рассмотрен в главе 7, “Протокол IP: доставка

дейтаграмм без установки соединения". Таким образом, у вас уже есть необходимые знания, чтобы понять механизм прямой доставки. Подытожив все сказанное выше, можно сделать такой вывод.

*Передача IP-дейтаграмм между двумя компьютерами, подключенными к одной физической сети происходит напрямую, без участия маршрутизаторов. Отправитель помещает дейтаграмму в физический фрейм, преобразует IP-адрес получателя в физический адрес и посыпает по нему фрейм, используя сетевое оборудование.*

Теперь осталось выяснить, как отправитель узнает о том, что он находится с конечным получателем в одной физической сети. Ответить на этот вопрос очень просто. В предыдущих главах говорилось о том, что IP-адрес состоит из двух частей: префикса, идентифицирующего сеть, и суффикса, определяющую узел в сети. Поэтому, чтобы определить, находится ли получатель в одной физической сети с отправителем, последний должен выделить номер сети из IP-адреса получателя и сравнить его с номером сети, выделенным из собственного IP-адреса. Если номера сетей совпадают, значит, дейтаграмма может быть послана получателю напрямую. Только что было продемонстрировано одно из основных преимуществ системы адресации в объединенной сети, использующей протокол TCP/IP.

*Для всех компьютеров, подключенных к одной физической сети, префиксы IP-адресов, идентифицирующие сеть, совпадают. Поскольку для извлечения префикса из IP-адреса требуется всего несколько машинных команд, процесс выяснения возможности напрямую отправить дейтаграмму конечному получателю является чрезвычайно эффективным.*

В объединенной сети прямая доставка выполняется на заключительном этапе пересылки любой дейтаграммы, независимо от того, через какое количество сетей и промежуточных маршрутизаторов она прошла. Последний из маршрутизаторов, находящийся на пути следования дейтаграммы и подключенный к одной физической сети с конечным получателем, выполняет ее прямую доставку до получателя. Таким образом, прямую доставку дейтаграмм следует рассматривать как частный случай общего процесса маршрутизации, выполняемого на всем этапе следования дейтаграммы от отправителя до конечного получателя. При прямой доставке дейтаграмма не проходит через промежуточные узлы маршрутизации.

### 8.3.2. Непрямая доставка

Процесс непрямой доставки является намного сложнее по сравнению с прямой доставкой дейтаграмм, поскольку отправителю нужно определить адрес ближайшего маршрутизатора, находящегося в одной с ним сети, которому должна быть послана дейтаграмма.

Чтобы понять как осуществляется маршрутизация, представьте себе большую объединенную сеть, к которой посредством маршрутизаторов подключено множество физических сетей. Представьте также два удаленных друг от друга компьютера, подключенных к объединенной сети, которым необходимо обменяться информацией. Как только одному из компьютеров нужно отправить дейтаграмму другому компьютеру, он инкапсулирует ее в сетевой фрейм и пересыпает его по физической сети ближайшему маршрутизатору. По определению, данную операцию может выполнить любой компьютер, поскольку все физические сети соединены между собой посредством маршрутизаторов. А это означает, что к каждой физической сети подключен как минимум один маршрутизатор. Таким образом, отправитель всегда может связаться с маршрутизатором по одной физической сети. Приняв дейтаграмму, маршрутизатор с помощью своего программного

обеспечения извлекает ее из сетевого фрейма и передает на обработку программам протокола IP. После этого программы маршрутизации выполняют поиск адреса следующего маршрутизатора, находящегося на пути следования пакета до конечного получателя. Как только будет определен IP-адрес следующего узла маршрутизации, дейтаграмма снова помещается в сетевой фрейм и пересыпается этому узлу по соответствующему участку физической сети. Этот процесс повторяется многократно до тех пор, пока дейтаграмма не будет отправлена конечно-му получателю методом прямой доставки. Подводя итог всему сказанному, можно отметить следующее.

*Маршрутизаторы в объединенной сети на основе протокола TCP/IP составляют взаимосвязанную и скооперированную структуру. Во время доставки дейтаграмма передается от одного маршрутизатора до другого до тех пор, пока она не достигнет маршрутизатора, находящегося с конечным получателем в одной физической сети. На заключительном этапе доставки дейтаграмма посылается конечному получателю напрямую.*

В описанном выше процессе маршрутизации непонятными остались два момента. Как маршрутизатор узнает, куда именно нужно передавать каждую из полученных дейтаграмм? Как компьютер отправителя определяет адрес ближайшего маршрутизатора, которому должна быть передана дейтаграмма, следующая в заданном направлении? Эти два вопроса тесно связаны, поскольку оба они затрагивают тему IP-маршрутизации. Чтобы ответить на них, мы должны сначала рассмотреть две важные темы. В этой главе будет описан простой алгоритм маршрутизации, использующий специальные таблицы. А в одной из следующих глав мы поговорим о том, как маршрутизаторы определяют новые пути следования дейтаграмм.

#### 8.4. Алгоритм табличной IP-маршрутизации

Обычно маршрутизация дейтаграмм в объединенной сети выполняется с помощью специальных таблиц межсетевой маршрутизации (*Internet routing table*), которые иногда называют таблицами IP-маршрутизации (*IP routing table*). В них хранится информация о возможных путях следования дейтаграмм и способах их достижения. Поскольку в процессе маршрутизации вовлечены как компьютеры пользователя, так и сетевые маршрутизаторы, такие таблицы должны храниться на *каждой* машине объединенной сети, независимо от выполняемых ею функций. В момент отправки дейтаграммы с компьютера пользователя запущенная на нем программа маршрутизации (она является частью программ поддержки протокола IP) с помощью таблицы маршрутизации определяет узел сети, которому следует послать эту дейтаграмму.

Какая же информация должна храниться в таблицах маршрутизации? Если на каждом компьютере в таблице маршрутизации будет содержаться информация о всех возможных адресах получателя, то, очевидно, что состояние такой таблицы очень трудно поддерживать актуальным постоянно. Более того, поскольку количество потенциальных адресатов очень велико, в компьютере пользователя может не хватить ни оперативной, ни дисковой памяти для хранения всего массива информации.

Интуитивно хотелось бы использовать принцип намеренного скрытия информации и сделать так, чтобы компьютеры выполняли маршрутизацию на основании минимально необходимых данных. Например, логично, чтобы подробная информация о конкретных узлах была сосредоточена в месте их подключения. Также крайне желательно, чтобы удаленный компьютер выполнял маршрутизацию пакетов к этим узлам, не вдаваясь в излишние подробности. Следует

отметить, что система IP-адресации как раз и создавалась с учетом этих требований. Напомним, что у всех машин, подключенных к одной физической сети, должны совпадать префиксы IP-адресов (т.е. как раз та часть IP-адреса, которая идентифицирует конкретную сеть). Выше уже шла речь о том, что подобный принцип назначения адресов позволяет с высокой эффективностью проверять, возможна ли прямая доставка дейтаграмм между двумя машинами. Таким образом, в таблицах маршрутизации достаточно указать только один префикс сети, а не IP-адреса всех ее машин.

## 8.5. Маршрутизация “на шаг вперед”

Использование выделенного из адреса получателя префикса, который идентифицирует сеть, вместо полного адреса узла позволяет резко повысить эффективность маршрутизации и сократить размер соответствующих таблиц. Кроме того, подобный подход позволяет скрыть информацию, сосредоточить данные о конкретных узлах в рамках локальной среды, к которой они подключены. Обычно в таблице маршрутизации содержатся пары значений ( $N, R$ ), где  $N$  представляет IP-адрес *сети* получателя, а  $R$  является IP-адресом “следующего” по порядку маршрутизатора, расположенного на пути следования пакетов до сети  $N$ . Маршрутизатор  $R$  называется *ближайшей точкой перехода* (*next hop*), а сама идея хранения в таблице маршрутизации адреса ближайшей точки перехода для каждого получателя называется *маршрутацией на шаг вперед* (*next-hop routing*). Таким образом, в таблице маршрутизации, хранящейся на узле  $R$ , содержатся данные о пути следования дейтаграмм от узла  $R$  до ближайшей точки перехода в направлении сети получателя. Заметьте, что маршрутизатор  $R$  не располагает данными о полном маршруте дейтаграммы к конечному получателю.

Важно отметить, что в каждом элементе таблицы маршрутизации содержится адрес маршрутизатора, которому могут быть посланы дейтаграммы по одной физической сети. Это означает, что все маршрутизаторы, адреса которых указаны в таблице маршрутизации узла  $M$ , должны находиться в тех же физических сетях, к которым напрямую подключена машина  $M$ . В момент отправки дейтаграммы с машины  $M$ , программное обеспечение протокола IP анализирует IP-адрес получателя и выделяет из него префикс сети. На основании значения этого префикса машина  $M$  принимает решение о том, какому из маршрутизаторов, находящихся “на расстоянии прямой видимости”, следует направить дейтаграмму.

На практике принцип скрытия информации применяется не только для маршрутизаторов, но и для узлов сети. Поэтому, хотя таблицы маршрутизации есть на всех узлах сети, в них должна храниться только необходимая для отправки дейтаграммы информация. Идея состоит в том, чтобы максимально освободить узлы сети от выполнения маршрутизации дейтаграмм и передать эти функции специализированным маршрутизаторам.

На рис. 8.2 приведен пример, позволяющий объяснить принцип применения таблиц маршрутизации. Объединенная сеть состоит из четырех физических сетей, соединенных посредством трех маршрутизаторов. На рис. 8.2 показана таблица маршрутизации, находящаяся на устройстве  $R$ . Поскольку маршрутизатор  $R$  напрямую подключен к двум сетям (20.0.0.0 и 30.0.0.0), для отправки пакетов любым машинам этих сетей он может пользоваться методом прямой доставки (преобразование логических адресов в физические, вероятнее всего, будет выполняться с помощью протокола ARP). При получении дейтаграммы, адресованной одному из узлов, расположенных в сети 40.0.0.0, маршрутизатор  $R$  должен перенаправить ее для дальнейшей доставки маршрутизатору  $S$ , адрес которого — 30.0.0.7. Поскольку маршрутизатор  $S$  имеет прямое подключение к сети 40.0.0.0, он сможет переслать данную дейтаграмму конечному получателю.

методом прямой доставки. В свою очередь маршрутизатор  $R$  может связаться напрямую с маршрутизатором  $S$ , поскольку последний имеет прямое подключение к сети 30.0.0.0.

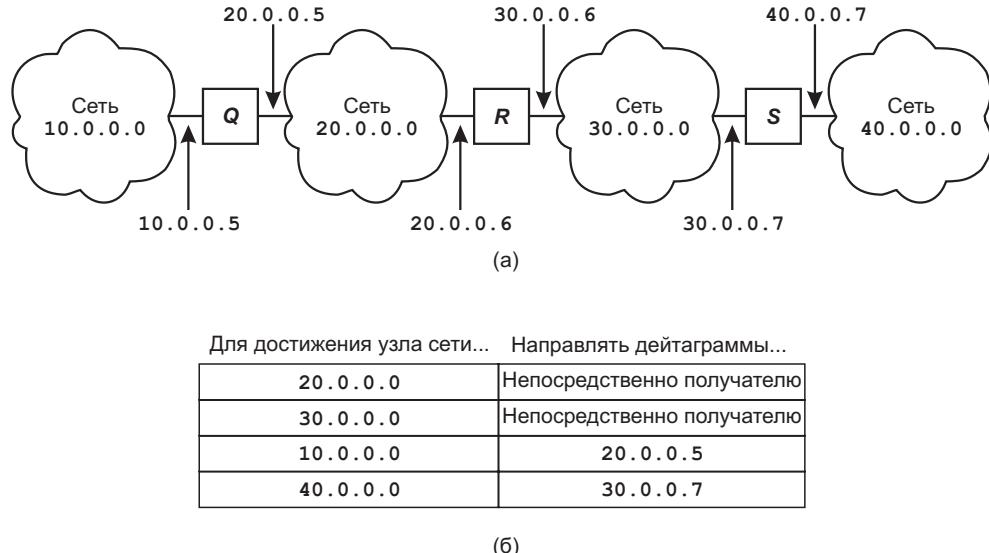


Рис. 8.2. Пример объединенной сети, состоящей из 4 физических сетей и трех маршрутизаторов (а); таблица маршрутизации устройства  $R$  (б)

Как видно из рис. 8.2, размер таблицы маршрутизации устройства  $R$  зависит от количества физических сетей в объединенной сети. Увеличение количества элементов в этой таблице происходит только в случае подключения к объединенной сети новых физических сетей. Однако следует отметить, что на размер таблицы маршрутизации и ее содержимое не влияет количество индивидуальных компьютеров, подключенных ко всем сетям. Таким образом, можно сформулировать следующий основополагающий принцип.

*Программное обеспечение протокола IP сохраняет в таблицах маршрутизации только информацию об адресах сетей (а не об адресах отдельных узлов этих сетей), в которых могут находиться потенциальные получатели дейтаграмм. Это позволяет избавиться от ненужной информации, уменьшить размер таблиц и повысить эффективность процесса маршрутизации.*

Выбор маршрута следования дейтаграмм на основе адреса сети получателя имеет несколько недостатков. Во-первых, в большинстве реализаций сетевого программного обеспечения такой подход означает, что весь поток данных, направленных в конкретную физическую сеть, будет проходить по одному и тому же маршруту. Как следствие, в случае если к сети получателя существует несколько возможных маршрутов следования, они не могут быть использованы одновременно. Кроме того, по одному и тому же маршруту будут направляться все дейтаграммы, независимо от запрошенного в их заголовках типа обслуживания (минимальное время доставки или высокая пропускная способность сети). Во-вторых, поскольку непосредственное соединение с конечным получателем может установить только последний из маршрутизаторов, находящихся на пути следования дейтаграммы, отправитель не в состоянии определить, подключен ли к физической сети узел получателя и нормально ли он функционирует. Таким

образом, нужно найти способ, с помощью которого маршрутизатор сможет сообщить отправителю о проблемах, возникших при доставке дейтаграммы. В третьих, поскольку все маршрутизаторы направляют потоки данных независимо друг от друга, может возникнуть ситуация, когда дейтаграмма, посланная от узла *A* к узлу *B*, будет проходить по другому маршруту, чем дейтаграмма, посланная от узла *B* к узлу *A*. Следовательно, нужно предусмотреть средства взаимодействия между маршрутизаторами, которые бы всегда гарантировали возможность двухстороннего обмена данными между этими узлами.

## 8.6. Стандартный маршрут следования дейтаграмм

Существует еще один метод, позволяющий избавиться от лишней информации и уменьшить размеры таблиц маршрутизации. Речь идет об объединении нескольких элементов таблицы маршрутизации в один стандартный блок. Суть его заключается в том, что программное обеспечение протокола IP при выполнении процедуры маршрутизации вначале должно просмотреть таблицу маршрутизации в поисках информации о сети получателя. Если в таблице маршрутизации информация не найдена, дейтаграмма посыпается *стандартному маршрутизатору (default router)*.

Понятие стандартного маршрута следования дейтаграммы широко применяется в небольших сетевых центрах, которым выделено несколько IP-адресов и которые имеют только одно сетевое соединение с остальной частью объединенной сети. В качестве примера можно привести небольшую локальную сеть, к которой подключено несколько компьютеров пользователей и только один маршрутизатор, связывающий их с внешним миром. Маршрутизация в такой сети будет выполняться в два этапа: анализ адресов локальной сети и отправка дейтаграмм, предназначенных для внешних адресатов, локальному маршрутизатору. Даже если локальная сеть предприятия состоит из нескольких физических сетей, применение стандартных маршрутов позволяет упростить процедуру маршрутизации — после проверки узлом сети адресов всех локальных сетей дейтаграммы, предназначенные для получателей за пределами локальной сети, передаются стандартному маршрутизатору.

## 8.7. Маршрутизация отдельных узлов сети

Выше мы уже говорили о том, что в сетях TCP/IP маршрутизация выполняется на основе адреса сети, а не адресов отдельных ее узлов. Тем не менее в большинстве реализаций протокола IP как частный случай предусмотрена возможность выполнения маршрутизации для адресов отдельных узлов. Это позволяет сетевым администраторам более точно распределять потоки данных в сети, тестировать участки сети, а также управлять правами доступа к сетевым ресурсам. Возможность задать индивидуальный маршрут дейтаграмм до конкретной машины особенно полезна при отладке сетевых подключений или проверке таблиц маршрутизации.

## 8.8. Алгоритм IP-маршрутизации

Принимая во внимание все сказанное выше, можно сформулировать алгоритм, используемый для маршрутизации IP-дейтаграмм<sup>2</sup> (листинг 8.1). По заданному IP-адресу конечного получателя и существующей таблице маршрутизации,

<sup>2</sup> В главе 10, “Бесклассовая адресация и подсети (CIDR)”, будет описан немногого видоизмененный алгоритм, который используется для маршрутизации дейтаграмм в бесклассовых сетях.

данный алгоритм определяет адрес ближайшей точки перехода, куда следует направить дейтаграмму. Не забывайте, что адрес ближайшей точки перехода должен находиться в той же сети, к которой имеет прямое подключение отправитель дейтаграммы.

#### Листинг 8.1. Алгоритм маршрутизации IP-дейтаграмм

Маршрутизация\_дейтаграммы (Дейтаграмма, Таблица\_Маршрутизации)

- 1: Извлечь из Дейтаграммы IP-адрес конечного получателя  $D$  и определить префикс сети  $N$ .
- 2: Если  $N$  совпадает с префиксом одной из сетей, к которой непосредственно подключена машина, выполнить прямую доставку дейтаграммы получателю  $D$  по соответствующей сети. (При этом нужно определить физический адрес машины  $D$ , инкапсулировать дейтаграмму в сетевой фрейм и отправить фрейм получателю).
- 3: Иначе, если в таблице маршрутизации указан специфический маршрут к машине  $D$ , переслать дейтаграмму в ближайшую точку перехода, адрес которой берется из таблицы.
- 4: Иначе, если в таблице маршрутизации указан маршрут для сети  $N$ , переслать дейтаграмму в ближайшую точку перехода, адрес которой берется из таблицы.
- 5: Иначе, если в таблице маршрутизации указан стандартный маршрут следования, отправить дейтаграмму стандартному маршрутизатору, адрес которого берется из таблицы.
- 6: Иначе, сгенерировать ошибку маршрутизации.

## 8.9. Маршрутизация с использованием IP-адресов

Прежде всего следует отметить, что, за исключением уменьшения значения поля времени жизни дейтаграммы и пересчета контрольной суммы ее заголовка, при выполнении маршрутизации в тело дейтаграммы не вносятся никакие изменения. В частности, не изменяются IP-адреса, находящиеся в полях заголовка дейтаграммы. Они всегда будут указывать на начального отправителя дейтаграммы и ее конечного получателя<sup>3</sup>. В результате выполнения алгоритма маршрутизации программа должна определить новый IP-адрес промежуточного получателя или IP-адрес машины, которой следует переслать дейтаграмму. В качестве промежуточного получателя чаще всего выступает следующий маршрутизатор, расположенный на пути следования дейтаграммы к конечному получателю. Если дейтаграмма может быть доставлена непосредственно получателю, то новый IP-адрес будет совпадать с адресом конечного получателя.

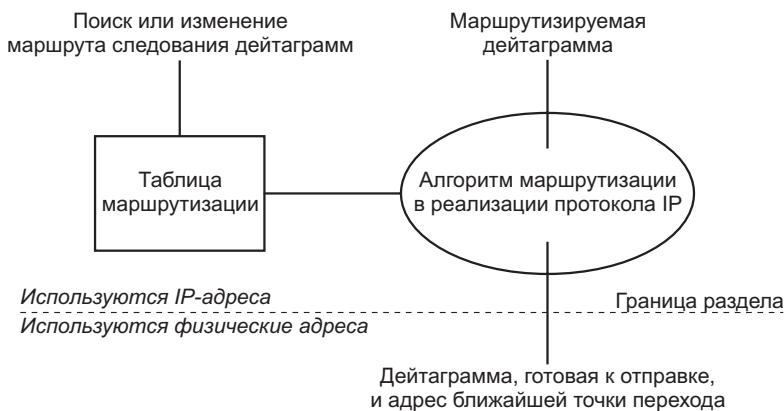
Выше уже шла речь о том, что IP-адрес, полученный в результате выполнения алгоритма маршрутизации, называется адресом *ближайшей точки перехода*, поскольку он указывает на узел в сети, которому должна быть послана дейтаграмма. Возникает вопрос, где в протоколе IP хранится адрес ближайшей точки перехода, поскольку в заголовке дейтаграммы поле для него не предусмотрено?

<sup>3</sup> За исключением случая, когда в дейтаграмме указаны параметры маршрутизации от источника.

Ответ прост: нигде. Выполнив маршрутизацию, программа протокола IP передает дейтаграмму и адрес ближайшей точки перехода сетевому программному обеспечению более низкого уровня. Оно поддерживает работу физической сети, по которой дейтаграмма должна быть послана. Низкоуровневое сетевое программное обеспечение преобразует IP-адрес ближайшей точки перехода в соответствующий физический адрес, помещает его в заголовок фрейма, а дейтаграмму — в область данных сетевого фрейма. Сформированный сетевой фрейм посыпается в сеть. После определения физического адреса ближайшей точки перехода ее IP-адрес теряется.

Может показаться странным, что в таблицах маршрутизации хранятся IP-адреса ближайших точек перехода (а не их физические адреса) для каждой из сетей, в которых находятся машины получателей. Ведь выше шла речь о том, что перед посылкой дейтаграммы в ближайшую точку перехода ее IP-адрес преобразуется в физический адрес. Поэтому, если представить себе абстрактную машину, отправляющую большое количество дейтаграмм по одним и тем же адресам, использование IP-адресов в таблице маршрутизации приведет к резкому снижению эффективности работы сети в целом. Ведь каждый раз после получения очередной дейтаграммы программное обеспечение протокола IP должно извлечь из ее заголовка IP-адрес получателя и с помощью таблицы маршрутизации определить адрес ближайшей точки перехода. После этого дейтаграмма вместе с адресом ближайшей перехода передается сетевому программному обеспечению более низкого уровня, отвечающему за работу сетевого интерфейса. Низкоуровневое программное обеспечение должно по IP-адресу определить физический адрес ближайшей точки перехода. Если бы в таблице маршрутизации хранились физические адреса ближайших точек перехода, то каждый из IP-адресов нужно было бы преобразовывать в физический адрес только один раз. В результате можно было бы сэкономить время центрального процессора сетевой машины, не выполняя ненужных вычислений.

Почему же все-таки в реализациях протокола IP не используются физические адреса при определении маршрутов следования дейтаграмм и создании таблиц маршрутизации? На то есть две причины, которые вы поймете, рассмотрев рис. 8.3.



*Рис. 8.3. Программа протокола IP и используемая ею таблица маршрутизации находятся выше воображаемой границы раздела адресов. Использование при маршрутизации только IP-адресов облегчает поиск или изменение маршрута следования дейтаграмм и позволяет скрыть особенности структуры физических адресов сетевого оборудования*

Во-первых, таблица маршрутизации является связующим звеном между программами протокола IP, выполняющими маршрутизацию дейтаграмм, и высокоуровневым программным обеспечением, которое управляет маршрутами следования дейтаграмм. При возникновении проблем в сети администраторы часто прибегают к анализу таблиц маршрутизации. Поэтому использование в них IP-адресов упрощает задачу сетевым администраторам и позволяет легко проконтролировать правильность обновления таблиц маршрутизации соответствующими программами. Во-вторых, основная цель при разработке протокола IP состояла в том, чтобы скрыть детали используемых низкоуровневых сетевых технологий.

На рис. 8.3 изображена важная концептуальная деталь — воображаемая *граница раздела адресов* (*address boundary*), которая позволяет показать, на каком уровне происходит разделение между низкоуровневым сетевым программным обеспечением, использующим физические адреса, и высокоуровневыми прикладными программами, работающими исключительно с логическими IP-адресами. При разработке программного обеспечения, логика работы которого относится к уровню, расположенному выше этой линии, должны использоваться исключительно IP-адреса. Круг применения физических адресов должен ограничиваться только несколькими небольшими низкоуровневыми подпрограммами. Ниже будет показано, что введение этой границы облегчает понимание принципа реализации программ остальных протоколов семейства TCP/IP, а также их отладку и дальнейшее сопровождение

## 8.10. Обработка входящих дейтаграмм

Выше был описан механизм IP-маршрутизации, а также методы обработки исходящих дейтаграмм. Однако, должно быть уже понятно, что программы протокола IP должны также обрабатывать и входящие дейтаграммы.

Получив сетевой фрейм, низкоуровневое сетевое программное обеспечение компьютера выделяет из него IP-дейтаграмму, которая затем передается на обработку модулю протокола IP. Если IP-адрес получателя, указанный в заголовке дейтаграммы, совпадает с одним из IP-адресов интерфейсов компьютера, программа поддержки протокола IP принимает такую дейтаграмму и передает ее для дальнейшей обработки соответствующему модулю протокола более высокого уровня. Если же IP-адреса не совпадают, программное обеспечение протокола IP должно аннулировать дейтаграмму (не забывайте, что рабочим станциям запрещено пересыпать дальше дейтаграммы, которые случайно попали к ним из-за ошибок в настройке таблиц маршрутизации).

В отличие от узлов сети, маршрутизаторы выполняют пересылку пакетов. Поэтому, когда IP-дейтаграмма поступает на маршрутизатор, она передается на обработку модулю протокола IP. В нем анализируется адрес получателя и определяется, может ли маршрутизатор доставить дейтаграмму непосредственно конечному получателю или он должен передать ее для дальнейшей обработки другому маршрутизатору. Как и в случае рабочей станции, при совпадении IP-адреса конечного получателя, находящегося в заголовке дейтаграммы с одним из IP-адресов интерфейсов маршрутизатора программы протокола IP передают поступившую дейтаграмму на обработку программам протокола более высокого уровня<sup>4</sup>. Если маршрутизатор не может непосредственно передать дейтаграмму

<sup>4</sup> Обычно маршрутизатор обрабатывает только служебные дейтаграммы, предназначенные для проверки работоспособности канала связи или содержащие команды управления процессом маршрутизации. Однако в некоторых случаях маршрутизатор отслеживает и сохраняет у себя все дейтаграммы, посланные по локальной сети в широковещательном режиме.

конечному получателю, программы протокола IP выполняют для нее маршрутизацию, используя в этих целях стандартный алгоритм и информацию из локальной таблицы маршрутизации.

Определение возможности доставки дейтаграммы получателю — не такая простая задача, как может показаться на первый взгляд. Напомним, что даже обычная рабочая станция может иметь несколько сетевых интерфейсов, подключенных к разным физическим сетям, каждому из которых назначается отдельный IP-адрес. Получив дейтаграмму, компьютер должен сравнить IP-адрес получателя с IP-адресами всех своих сетевых интерфейсов. Если один из адресов совпадет, дейтаграмма принимается для дальнейшей обработки. Компьютер также принимает для обработки дейтаграммы, разосланные по локальной сети в широковещательном режиме. При этом IP-адрес получателя в заголовке дейтаграммы, должен совпадать с ограниченным широковещательным адресом или направленным широковещательным адресом этой сети. Как будет показано в главах 10, “Бесклассовая адресация и подсети (CIDR)”, и 17, “Режим многоадресатной передачи в объединенной сети”, использование бесклассовых методов адресации, подсетей и многоадресатных адресов еще больше усложняет дело. Однако в любом случае, если IP-адрес получателя дейтаграммы не совпадет ни с одним из локальных IP-адресов данного компьютера, программы протокола IP уменьшают на единицу значение поля времени жизни в заголовке дейтаграммы и на основе этого значения решают, как поступить с дейтаграммой. Если значение времени жизни стало равным нулю, дейтаграмма аннулируется. В противном случае вычисляется новое значение контрольной суммы заголовка и выполняется маршрутизация дейтаграммы.

Должен ли любой подключенный к сети компьютер выполнять дальнейшую пересылку всех полученных IP-дейтаграмм? Очевидно, что эти функции должен по определению выполнять маршрутизатор. Выше уже шла речь о том, что функции маршрутизатора может выполнять практически любой многоадресный узел сети, даже если обычно он используется только в качестве рабочей станции пользователя. И хотя такое решение нельзя назвать удачным, тем не менее, установив соответствующие параметры настройки протокола TCP/IP можно добиться того, чтобы многоадресный узел пересыпал дейтаграммы так же, как обычный маршрутизатор. А как быть с остальными узлами сети, которые не предназначены для работы в качестве маршрутизаторов? Ответ прост: узлы сети, которые не являются маршрутизаторами, *не должны* выполнять пересылку случайно попавших к ним дейтаграмм; такие дейтаграммы должны быть аннулированы.

Существует по крайней мере четыре причины, по которым узлы сети, не являющиеся маршрутизаторами, не должны пересыпать дейтаграммы. Во-первых, получение компьютером пользователя дейтаграмм, предназначенных для других машин сети, свидетельствует о возникновении проблем в системе межсетевой адресации, маршрутизации или доставки. Если бы узел сети, выполняя функции маршрутизатора и корректно пересыпал дейтаграммы, то о возникших проблемах в сети никто бы не узнал. Во-вторых, процесс маршрутизации связан с дополнительными потоками пересылаемых по сети данных, а также нагрузкой на центральный процессор компьютера, предсказать которые заранее невозможно. Все это может замедлить работу прикладных программ, запущенных на компьютере пользователя. В-третьих, возникновение простых ошибочных ситуаций может привести к хаосу в сети. Предположим, что любой узел в сети может выполнять маршрутизацию дейтаграмм. Представьте себе, что произойдет, если одна из машин случайно отправит в широковещательном режиме дейтаграмму, которая предназначается для одной из машин сети, например *H*. Поскольку для отправки дейтаграммы был выбран широковещательный режим,

ее копию получит каждый из узлов сети и перешлет ее узлу  $H$ . В результате компьютер  $H$  будет завален множеством копий дейтаграммы. В-четверых, как будет описано в следующих главах, маршрутизаторы, кроме перенаправления потоков данных в сети, выполняют ряд других полезных функций. Одна из них — уведомление отправителя о возникших в сети проблемах, связанных с доставкой его дейтаграммы. Для этой цели используется специальный протокол, о котором пойдет речь в следующей главе. Следует отметить, что обычные узлы сети не рассыпают подобных уведомлений (причина все та же — чтобы не перегружать отправителя одними и теми же сообщениями об ошибках). Кроме того, маршрутизаторы периодически рассыпают информацию о маршрутах следования дейтаграмм. Делается это для согласования информации в таблицах маршрутизации. Таким образом, если узел сети будет выполнять маршрутизацию дейтаграмм, не поддерживая при этом всех функций маршрутизатора, то результат подобных действий — непредсказуем.

## 8.11. Создание таблиц маршрутизации

Выше в этой главе было описано, как в протоколе IP выполняется маршрутизация дейтаграмм на основе информации, хранящейся в таблицах маршрутизации. Однако мы не упомянули о том, как в системе инициализируются эти таблицы или обновляются их содержимое при изменении топологии сети. Затронутый нами вопрос более подробно описан в последующих главах. Там же речь пойдет о специальном протоколе, который используется маршрутизаторами для согласования содержимого своих таблиц маршрутизации. А пока вам достаточно понять один важный момент. Программы протокола IP обращаются к таблице маршрутизации всякий раз, когда нужно определить адрес ближайшей точки перехода, по которому следует передать дейтаграмму. Поэтому изменение содержимого таблиц маршрутизации автоматически влечет за собой изменение маршрута следования дейтаграмм.

## 8.12. Резюме

Программы протокола IP используют маршрутную информацию для пересылки дейтаграмм. На основании IP-адреса конечного получателя компьютер проводит ряд вычислений, целью которых является определение ближайшего адреса узла сети, которому следует переслать дейтаграмму. Прямая доставка возможна в том случае, если IP-адреса машин получателя и отправителя дейтаграммы принадлежат к одной сети. Такой метод доставки применяется на заключительном этапе пересылки дейтаграммы. Если же отправитель не может напрямую переслать дейтаграмму получателю, он должен отправить ее маршрутизатору. В этом и состоит основной принцип маршрутизации, т.е. дейтаграмма пересыпается по объединенной сети от одного маршрутизатора до другого до тех пор, пока она не будет доставлена напрямую конечному получателю по одной физической сети.

Процесс поиска маршрута следования дейтаграммы, выполняемый программным обеспечением протокола IP, заключается в определении IP-адреса следующего компьютера (т.е. адреса ближайшей точки перехода), которому следует передать дейтаграмму для дальнейшей доставки ее конечному получателю. Для отправки дейтаграммы по физической сети программам низкого уровня, обслуживающим плату сетевого интерфейса, передается сама дейтаграмма и адрес ближайшей точки перехода. Прежде чем дейтаграмма будет передана по сети, она помещается в физический фрейм, IP-адрес ближайшей точки перехода преобразовывается в физический адрес и записывается в заголовок фрейма. После

выполнения таких подготовительных действий дейтаграмма отправляется в ближайшую точку перехода.

В алгоритме маршрутизации, применяемом в объединенной сети, используются только IP-адреса, а его работа зависит от содержимого таблиц маршрутизации. В большинстве этих таблиц хранятся не IP-адреса отдельных узлов (хотя такое тоже возможно), а адреса сетей, что позволяет сократить размеры таблиц. Использование стандартных маршрутов дейтаграмм позволяет еще больше сократить размеры таблиц маршрутизации, особенно в том случае, когда узлы сети имеют доступ только к одному маршрутизатору.

## Материал для дальнейшего изучения

Маршрутизация является одной из основных тем при изучении протокола TCP/IP. Общие вопросы маршрутизации рассмотрены в работах Фрэнка (Frank) и Чу (Chou) [54]; Шварца (Schwartz) и Штерна (Stern) [116], а также Постела (Postel) [104]. В [RFC 1009] Браден (Braden) и Постел обобщили эти сведения и описали способы обработки дейтаграмм маршрутизаторами объединенной сети. Обзор методов маршрутизации в Internet сделан в работе Нартена (Narten) [91]. Методы адаптивной маршрутизации проанализированы в статье Фальца (Fultz) и Клейнрока (Kleinrock) [55]. Адаптивные алгоритмы маршрутизации, используемые в сети ARPANET, описаны в статье Мак-Куиллана (McQuillan), Ричера (Richer) и Розена (Rosen) [85].

Идея сформулировать основные принципы маршрутизации на основе ряда правил витала в воздухе уже давно. В [RFC 1124] Лейнер (Leiner) рассматривает правила маршрутизации для объединенных сетей. Браун (Braun) в [RFC 1104] описывает несколько моделей маршрутизации для объединенных сетей. В [RFC 1092] Рехтер (Rekhter) описывает правила маршрутизации, которые были приняты во втором магистральном канале NSFNET. Правила маршрутизации, принятые в протоколе IP, описаны Кларком (Clark) в [RFC 1102].

## Упражнения

- 8.1. Создайте таблицы маршрутизации для всех маршрутизаторов, изображенных на рис. 8.2. Для каких из маршрутизаторов наиболее оправдано использование стандартных маршрутов дейтаграмм?
- 8.2. Изучите алгоритмы маршрутизации, используемые в локальной сети вашего предприятия. Учтены ли в них все ситуации, которые были описаны в этой главе? А те, которые не описаны?
- 8.3. Как поступает маршрутизатор со значением поля *времени жизни* в заголовке дейтаграммы?
- 8.4. Предположим, что существует компьютер с двумя сетевыми интерфейсами, подключенными к двум физическим сетям. Интерфейсам назначены IP-адреса  $I_1$  и  $I_2$ . Может ли такой компьютер получать по сети, к которой подключен интерфейс  $I_1$ , дейтаграммы, адресованные интерфейсу  $I_2$ ? Обоснуйте свой ответ.
- 8.5. Предположим, что два узла сети,  $A$  и  $B$ , подключены к общей физической сети  $N$ . Может ли узел  $A$  при использовании описанного в этой главе алгоритма маршрутизации получить дейтаграмму, предназначенную для узла  $B$ ? Обоснуйте свой ответ.

- 8.6.** Измените алгоритм маршрутизации так, чтобы в нем учитывались параметры маршрутизации от источника, описанные в главе 7, “Протокол IP: доставка дейтаграмм без установки соединения”.
- 8.7.** Поясните, почему время, затрачиваемое IP-маршрутизатором на обработку входящей дейтаграммы, пропорционально длине ее заголовка?
- 8.8.** Для того чтобы облегчить наблюдение за локальной сетью и выявление в ней проблем сетевой администратор собирается изменить алгоритм маршрутизации. При этом он хочет, чтобы проверка возможности прямой доставки дейтаграмм выполнялась *после* проверки маршрутов к конкретным узлам. Как, по вашему мнению, он собирается использовать этот новый алгоритм для создания программы сетевого мониторинга?
- 8.9.** Можно ли в качестве адреса конечного получателя дейтаграммы указать один из IP-адресов маршрутизатора? Имеет ли смысл так поступать?
- 8.10.** Рассмотрите работу измененного алгоритма маршрутизации, описанного в упр. 8.8. При каких условиях применение этого алгоритма оправдано, а при каких — нет?
- 8.11.** Постарайтесь распутать одну детективную историю. Проведя наблюдение в один из вечеров за потоком данных, протекающим по локальной сети в течение 10 мин, некто заметил, что во всех фреймах, предназначенных для машины A, находятся IP-дейтаграммы, в заголовке которых в качестве адреса получателя указан IP-адрес машины A. В то же время во всех фреймах, предназначенных для машины B, находятся IP-дейтаграммы, адрес получателя которых не совпадает с IP-адресом машины B. Пользователи говорят, что обе машины A и B, находились в рабочем состоянии и могли свободно обмениваться данными. Обоснуйте ответ.
- 8.12.** Как следовало бы изменить формат IP-дейтаграммы, чтобы добиться максимальной скорости коммутации пакетов в маршрутизаторах? (*Подсказка.* Маршрутизатор должен пересчитать контрольную сумму заголовка дейтаграммы после уменьшения на единицу значения поля времени жизни.)
- 8.13.** Сравните протокол CLNP (протокол доставки пакетов, не требующий установки соединения с получателем, принятый ISO, стандарт 8473) с протоколом IP. Насколько хорошо протокол ISO приспособлен к высокоскоростной коммутации пакетов? (*Подсказка.* Учтите, что наличие в заголовке дейтаграммы полей переменной длины снижает производительность маршрутизатора.)



# 9

## *Протокол IP: обработка ошибок и управляющие сообщения (ICMP)*

### **9.1. Введение**

В предыдущих главах был описан механизм ненадежной доставки дейтаграмм протокола IP, не требующий установки соединения с получателем. Напомним, что доставка дейтаграммы конечному получателю происходит за счет ее пересылки от одного маршрутизатора до другого. Этот процесс продолжается до тех пор, пока дейтаграмма не попадет в ту сеть, к которой подключен конечный получатель. В результате последний из маршрутизаторов сможет переслать ее напрямую получателю. Однако в некоторых случаях маршрутизатор по какой-либо причине не может переслать дейтаграмму другому маршрутизатору или доставить ее получателю. Кроме того, иногда в сети возникают непредвиденные ситуации, которые могут повлиять на возможность пересылки дейтаграмм самим маршрутизатором (например, в случае перегрузки сети). Поэтому для маршрутизатора должны быть предусмотрены средства извещения отправителя о проблемах с доставкой дейтаграммы, чтобы он смог принять необходимые меры или попытался устраниить причину возникшей проблемы. В этой главе описывается механизм взаимодействия маршрутизаторов и узлов объединенной сети, позволяющий им обмениваться служебной информацией и сообщениями об ошибках. Ниже будет показано, как этот механизм используется в маршрутизаторах для извещения узлов сети о возникших проблемах, а также возможность его использования узлами сети для проверки связи с конечным получателем.

### **9.2. Протокол межсетевых управляющих сообщений (ICMP)**

В системах не требующих установки соединения с конечным получателем, которые мы рассматривали до сих пор, каждый маршрутизатор является автономным устройством. А это означает, что пересылка и доставка поступающих на маршрутизатор дейтаграмм выполняется без какого-либо участия со стороны их отправителей. Такая система будет работать очень хорошо только в том случае, если все компьютеры, вовлеченные в процесс доставки дейтаграмм, нормально функционируют, а все маршруты следования дейтаграмм согласованы. К сожалению, составляющие любой большой коммуникационной системы рано или поздно выходят из строя. Кроме обрывов линий связи и сбоев в работе компьютеров, доставка дейтаграмм также невозможна в случае отключения (временного или постоянного) машины конечного получателя от сети, при обнулении счетчи-

ка времени жизни или когда один или несколько промежуточных маршрутизаторов оказываются настолько перегруженными, что не могут справиться со входящим потоком данных. Основное отличие сети, реализованной на аппаратном уровне, от объединенной сети, реализованной на программном уровне, заключается в том, что в первом случае разработчик может добавить специальное оборудование, которое будет информировать подключенные к сети узлы о возникших проблемах. В объединенной сети, которая не имеет подобного аппаратного механизма оповещения, отправитель не сможет определить причину, по которой дейтаграмма не была доставлена получателю, — обусловлено ли это сбоем в работе локального или удаленного компьютера. В подобных случаях выявление источника проблемы становится крайне затруднительным. В протоколе IP также не предусмотрено никаких средств, позволяющих отправителю проверить связь с получателем или получить информацию о возникшей проблеме.

Поэтому, чтобы маршрутизаторы объединенной сети могли оповещать узлы сети о возникающих ошибках или нештатных ситуациях, разработчики добавили в семейство протоколов TCP/IP механизм рассылки специальных сообщений. Этот механизм, который называли *протоколом межсетевых управляющих сообщений* (*Internet Control Message Protocol*, или *ICMP*), считается неотъемлемой частью протокола IP и должен быть обязательно включен в любую его реализацию.

Управляющие ICMP-сообщения (по аналогии с любыми другими сообщениями) перед передачей по объединенной сети помещаются в область данных IP-дейтаграммы. Однако конечным получателем таких сообщений является не конкретная прикладная программа или пользователь удаленного компьютера, а программное обеспечение протокола IP, запущенное на этом компьютере. Другими словами, полученное ICMP-сообщение, обрабатывает специальный модуль ICMP протокола IP. Обнаружив, что проблема была вызвана протоколом высокого уровня или прикладной программой, модуль ICMP информирует об этом соответствующий программный модуль. Таким образом, из всего сказанного выше напрашивается следующий вывод.

*Протокол ICMP позволяет маршрутизаторам отправлять другим маршрутизаторам или узлам сети сообщения об ошибках или управляющие сообщения. Этот протокол обеспечивает средство связи между программами протокола IP двух компьютеров.*

Первоначально протокол ICMP был предназначен для уведомления маршрутизаторами узлов сети о возникших проблемах с доставкой посланных ими пакетов. Однако его область применения не исчерпывается исключительно маршрутизаторами. Узел сети может отправить ICMP-сообщение любому узлу сети. Хотя стоит отметить, что в стандарте ограничивается использование узлами сети некоторых типов ICMP-сообщений. Таким образом, узел сети может использовать протокол ICMP как средство связи с маршрутизатором или другим узлом. Основным преимуществом использования протокола ICMP узлами сети является то, что при этом обеспечивается единый универсальный механизм обмена управляющими и информационными сообщениями любого типа.

### **9.3. Уведомление об ошибках или коррекция ошибок?**

Говоря формально, протокол ICMP является *механизмом уведомления об ошибках*. В нем предусмотрены средства, позволяющие маршрутизатору сообщить отправителю о возникшей при доставке дейтаграммы проблеме. Хотя в спецификации протокола ICMP описаны допустимые способы его использования и предполагаемые действия, которые должны выполняться в ответ на получение

уведомлений о ряде ошибок, тем не менее там не указан полный перечень действий, которые должны быть выполнены при возникновении каждой из возможных ошибок. Подводя итог, можно отметить, что

*если при доставке дейтаграммы возникает нештатная ситуация, то с помощью средств протокола ICMP отправителю дейтаграммы может быть послано только соответствующее уведомление. Отправитель должен переслать это уведомление нужной прикладной программе или предпринять другие действия для устранения проблемы.*

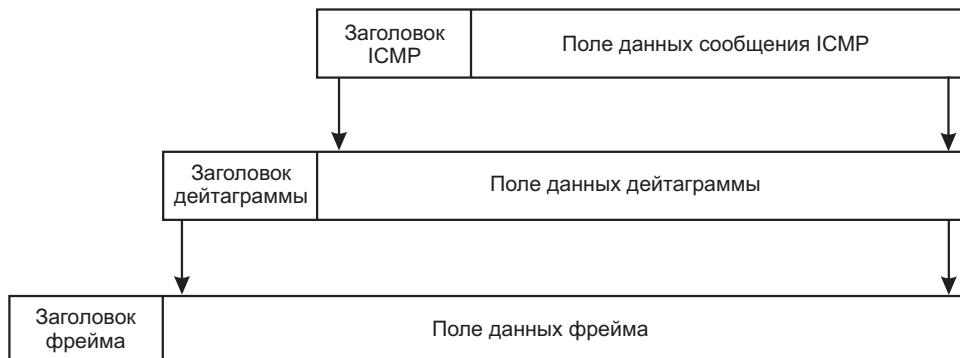
Во многих случаях ошибочные ситуации возникают вследствие некорректных действий со стороны отправителя дейтаграммы, однако это не всегда так. Поскольку протокол ICMP предназначен для уведомления отправителя дейтаграммы, его нельзя использовать для рассылки сообщений о возникшей проблеме промежуточным маршрутизаторам. Например, предположим, что дейтаграмма передается по некоторому маршруту и последовательно проходит через маршрутизаторы  $R_1, R_2, \dots, R_k$ . Предположим также, что в таблице маршрутизации устройства  $R_k$  содержится некорректная информация, поэтому дейтаграмма будет переслана устройству  $R_E$ . При этом маршрутизатор  $R_E$  с помощью протокола ICMP не сможет уведомить о возникшей проблеме устройство  $R_k$ , поскольку сообщение об ошибке будет послано отправителю дейтаграммы. К сожалению, в данном случае отправитель дейтаграммы никак не сможет повлиять на источник проблемы и перенастроить таблицу маршрутизации устройства  $R_k$ . Более того, отправитель может и не определить, какой из маршрутизаторов работает некорректно.

Почему же тогда протокол ICMP предназначен для уведомления только отправителя дейтаграммы? Ответ на этот вопрос станет понятен после анализа формата дейтаграммы и принципов маршрутизации, которые были описаны в предыдущих главах. В заголовке дейтаграммы предусмотрены поля только для адресов отправителя и конечного получателя. В нем не предусмотрено места для адресов всех маршрутизаторов, через которые прошла дейтаграмма по объединенной сети (за исключением особых случаев применения параметра регистрации маршрута дейтаграммы). Причина проста — поскольку каждый маршрутизатор ведет собственную таблицу маршрутизации и может без предупреждения изменить ее содержимое, заранее предсказать маршрут конкретной дейтаграммы нельзя. Поэтому нет никакой возможности определить путь, по которому прошла дейтаграмма до заданного маршрутизатора. Обнаружив проблему, маршрутизатор не сможет сообщить об этом всем промежуточным устройствам, через которые прошла дейтаграмма, поскольку их адреса неизвестны. Однако вместо того, чтобы просто “тихо” аннулировать дейтаграмму, маршрутизатор с помощью протокола ICMP информирует отправителя о возникшей проблеме. При этом подразумевается, что, получив уведомление, пользователь компьютера свяжется с сетевым администратором, и проблема будет установлена и устранена.

## 9.4. Доставка ICMP-сообщений

Для доставки ICMP-сообщений требуется выполнить процесс инкапсуляции два раза, как показано на рис. 9.1. ICMP-сообщение помещается в область данных IP-дейтаграммы, которая, в свою очередь, помещается в область данных сетевого фрейма. Процесс маршрутизации дейтаграмм с ICMP-сообщениями ничем не отличается от маршрутизации любых других дейтаграмм, содержащих пользовательские данные. Им не уделяется особое внимание, они не имеют также никакого приоритета перед обычными дейтаграммами. Сообщения об ошибках могут быть потеряны по дороге к отправителю дейтаграммы, их может также

аннулировать один из промежуточных маршрутизаторов. Более того, в перегруженной сети сообщения об ошибках приводят к еще большему увеличению трафика. Однако следует отметить, что если нештатная ситуация возникает при доставке IP-дейтаграммы, содержащей ICMP-сообщение, ее обработка выполняется по особому алгоритму. Суть его состоит в том, что в сети не должны появляться сообщения об ошибках на сообщения об ошибках. Другими словами, ICMP-сообщения не генерируются в ответ на ошибки, вызванные доставкой дейтаграмм, содержащих ICMP-сообщения.



*Рис. 9.1. Двухуровневая инкапсуляция ICMP-сообщений. Вначале ICMP-сообщение помещается в поле данных IP-дейтаграммы, а затем сама дейтаграмма помещается в поле данных фрейма для передачи по физической сети. Для идентификации ICMP-сообщения в поле типа протокола заголовка дейтаграммы помещается значение 1*

Однако следует иметь в виду, что, хотя ICMP-сообщения инкапсулируются в IP-дейтаграммы и посылаются с помощью протокола IP, протокол ICMP не относится к высокоуровневым сетевым протоколам, так как он является неотъемлемой частью протокола IP. Поскольку по пути до своего получателя ICMP-сообщения могут проходить по сетям разных типов, для их доставки нельзя использовать протоколы физического уровня. Поэтому и был выбран протокол IP.

## 9.5. Формат ICMP-сообщений

Каждый тип ICMP-сообщений имеет собственный формат, однако все они начинаются с трех одинаковых полей: 8-разрядного целого поля *типа* сообщения, идентифицирующего сообщение; 8-разрядного поля *кода* сообщения, в котором хранится дополнительная информация о сообщении; и 16-разрядного поля *контрольной суммы*. В протоколе ICMP используется такой же аддитивный алгоритм вычисления контрольной суммы, как и в протоколе IP, однако в данном случае вычисляется только контрольная сумма самого ICMP-сообщения.

При возникновении ошибочной ситуации в соответствующее ICMP-сообщение помещается заголовок дейтаграммы, вызвавшей проблему, и ее первые 64 бита данных. Это позволяет получателю более точно определить, какой из протоколов и какая из прикладных программ несет ответственность за возникновение проблемы. Как будет показано ниже, высокоуровневые протоколы, входящие в семейство протоколов TCP/IP, спроектированы так, чтобы вся ключевая информация содержалась в первых 64 битах их сообщений.

Поле типа ICMP-сообщения определяет его формат и содержимое. Типы ICMP-сообщений приведены в табл. 9.1. Форматы сообщений и их назначение будут описаны в следующих разделах.

---

**Таблица 9.1. Типы ICMP-сообщений**

---

<b>Значение в поле типа</b>	<b>Описание ICMP-сообщения</b>
0	Ответ на запрос эха
3	Получатель недостижим
4	Подавление источника данных
5	Переадресация (изменение маршрута)
8	Запрос эха
9	Извещение о маршрутизаторе
10	Запрос на адрес маршрутизатора
11	Истекло время ожидания получения фрагментов или время жизни дейтаграммы
12	Ошибка в параметрах дейтаграммы
13	Запрос временной метки
14	Ответ, содержащий временную метку
15	Информационный запрос (устаревший)
16	Информационный ответ (устаревший)
17	Запрос маски адреса
18	Ответ, содержащий маску адреса

---

## 9.6. Проверка связи с получателем и его состояния (ping)

В семействе протоколов TCP/IP предусмотрены специальные средства, облегчающие сетевым администраторам и пользователям поиск неисправностей в сети. Чаще всего для целей отладки используют два типа ICMP-сообщений: *запрос эха* и *ответ на него*. Любой узел сети или маршрутизатор может отправить заданному получателю ICMP-сообщение с запросом эха. Получив такой запрос, компьютер должен сформировать ответ и отослать его отправителю. Обычно в запрос включается необязательная область данных, содержимое которой переписывается в ответное сообщение. Запрос эха и ответное сообщение на него могут использоваться для проверки возможности установки с получателем двухсторонней связи. Поскольку ICMP-сообщения пересылаются по сети в виде IP-дейтаграмм, получение ответа на запрос эха свидетельствует о том, что основные элементы системы транспортировки дейтаграмм работают нормально. Напомним их вкратце. Во-первых, программное обеспечение, работающее на компьютере отправителя, должно корректно выполнять маршрутизацию дейтаграмм. Во-вторых, все промежуточные маршрутизаторы, находящиеся по пути следования дейтаграммы от отправителя до конечного получателя, должны быть в рабочем состоянии и правильно пересыпать дейтаграммы. В-третьих, машина конечного получателя должна находиться в рабочем состоянии (по крайней мере отвечать на поступившие запросы), а ее программное обеспечение протокола IP и ICMP должно функционировать в штатном режиме. И, наконец, в таблицах маршрутизации всех промежуточных устройств должна содержаться корректная информация.

В большинстве операционных систем для отправки запросов эха предусмотрена специальная команда, которая называется `ping`<sup>1</sup>. Усложненные версии этой команды посылают несколько последовательных ICMP-запросов, обрабатывают поступающие на них ответы и выдают статистику по прохождению и потере дейтаграмм. С помощью специальных параметров команды `ping`, пользователь может определить размер посылаемых данных и интервал времени между запросами. Более простые версии этой программы посылают только один ICMP-запрос и ожидают на него ответ.

## 9.7. Форматы запроса на эхо и ответного сообщения

На рис. 9.2 показан формат запроса на эхо и ответа на него.

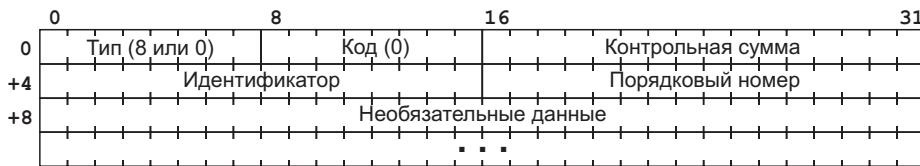


Рис. 9.2. Формат запроса на эхо и ответного ICMP-сообщения

Область необязательных данных имеет переменную длину, в нее помещаются данные, которые должны быть возвращены отправителю. В ответном ICMP-сообщении всегда содержатся те же данные, которые были получены в первоначальном запросе эха. Поля идентификатора и порядкового номера предназначены для того, чтобы отправитель мог сопоставить запросы эха и получаемые на них ответы. В запросе эха значение поля “тип” устанавливается равным 8, а в ответе на него — 0.

## 9.8. Извещение об отсутствии связи с получателем

Если маршрутизатор не может переслать или доставить IP-дейтаграмму, он посылает ее отправителю уведомление о недостижимости получателя. Формат сообщения показан на рис. 9.3.

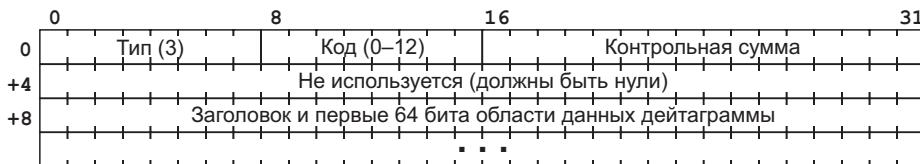


Рис. 9.3. Формат ICMP-сообщения о недостижимости получателя

В поле кода ICMP-сообщения рассматриваемого типа помещается целое число, позволяющее уточнить суть проблемы (табл. 9.2).

Хотя в протоколе IP применен механизм, не гарантирующий доставку пакетов, дейтаграммы в нем не теряются бесследно. Как только маршрутизатор обнаруживает, что не сможет переслать или доставить дейтаграмму получателю, он посылает отправителю уведомление о недостижимости получателя и только тогда *аннулирует* (т.е. уничтожает) дейтаграмму. Ошибки, при которых сеть

<sup>1</sup> Как заметил однажды Дэйв Миллс (Dave Mills), название *PING* является сокращением от *Packet InterNet Groper*, или *межсетевой пакетный тестер*.

получателя оказывается недостижимой, обычно свидетельствуют об отказе одного из промежуточных маршрутизаторов. Если же недостижим узел сети, это свидетельствует о проблемах с доставкой дейтаграммы<sup>2</sup>. Поскольку в ICMP-сообщении об ошибке содержится фрагмент начала дейтаграммы, которая вызвала проблему, отправитель всегда сможет точно определить, какой из адресов получателя недостижим.

**Таблица 9.2. Коды ICMP-сообщений о недостижимости получателя**

<i>Код</i>	<i>Описание</i>
0	Сеть недостижима
1	Узел сети недостижим
2	Протокол недоступен
3	Порт недоступен
4	Необходима фрагментация, однако установлен бит ее запрета
5	Ошибка маршрутизации от источника
6	Сеть получателя неизвестна
7	Узел получателя неизвестен
8	Узел отправителя изолирован
9	Связь с сетью получателя запрещена на административном уровне
10	Связь с узлом получателя запрещена на административном уровне
11	Сеть недостижима для данного типа обслуживания
12	Узел недостижим для данного типа обслуживания

Связь с узлом получателя может отсутствовать по нескольким причинам. Одна из них — отказ оборудования. Другая — отправитель указал несуществующий адрес получателя. Третья (в редких случаях) — маршрутизатор не располагает данными о маршруте к сети получателя. Обратите внимание, что маршрутизаторы сообщают только о тех нештатных ситуациях, которые они обнаружили, поскольку другие проблемы доставки им могут быть не известны. Например, если машина получателя подключена к сети Ethernet, то ее сетевое оборудование никак не сообщает о факте получения дейтаграммы. Более того, маршрутизатор может продолжать некоторое время посыпать пакеты этой машине даже после ее внезапного отключения от сети (например, в случае пропадания напряжения питания или обрыва кабеля). При этом маршрутизатор так и не узнает, доставлены ли пакеты. Можно сделать следующий вывод.

*Несмотря на то что при невозможности переслать или доставить дейтаграмму получателю, маршрутизатор посылает ее отправителю соответствующее уведомление, сам маршрутизатор не может выявить все возможные проблемы с ее доставкой.*

Смысл сообщений об ошибке, связанный с недоступностью протокола и порта, станет понятен чуть позже, когда мы рассмотрим, как в протоколах высокого

<sup>2</sup> За исключением случаев, когда в маршрутизаторе применяется система адресации подсетей, описанная в главе 10, “Бесклассовая адресация и подсети (CIDR)”. Тогда сообщение о недостижимости узла сети свидетельствует о проблемах с маршрутизацией подсети.

уровня используются абстрактные точки назначения, называемые *портами* (*ports*). Смысл большинства остальных сообщений об ошибках очевиден из их названия. Если в дейтаграмме указан параметр маршрутизации от источника, и в нем задан некорректный маршрут следования, то маршрутизатор может прислать извещение об ошибке *маршрутизации от источника*. Если для дальнейшей передачи дейтаграммы маршрутизатор должен ее фрагментировать, а в ее заголовке установлен бит запрета фрагментации, отправителю посыпается соответствующее уведомление.

## 9.9. Перегрузка сети и управление потоком дейтаграмм

Поскольку в протоколе IP для обмена данными не требуется предварительная установка соединения с получателем, маршрутизатор не может зарезервировать память или другие сетевые ресурсы до получения дейтаграммы. В результате входящий поток данных может привести к тому, что все ресурсы маршрутизатора будут исчерпаны, т.е. произойдет его *перегрузка*. Перегрузка может произойти по двум совершенно разным причинам. Во-первых, современные высокоскоростные компьютеры способны генерировать поток данных, во много раз превышающий пропускную способность канала связи. Например, представьте себе, что по объединенной сети должен передаваться трафик от суперкомпьютера. Даже если предположить, что сам суперкомпьютер подключен к высокоскоростной локальной сети, в конечном итоге данные попадут в низкоскоростную глобальную сеть. Очевидно, что в этом случае возникнет перегрузка маршрутизатора, соединяющего локальную сеть с глобальной, поскольку дейтаграммы будут прибывать быстрее, чем они могут быть отправлены. Во-вторых, перегрузка маршрутизатора может возникнуть в том случае, если через него будут проходить дейтаграммы, посланные за короткий промежуток времени с нескольких компьютеров. При этом поток данных от одного компьютера чаще всего не вызывает проблем с маршрутизатором.

Если дейтаграммы прибывают быстрее, чем может обработать узел сети или маршрутизатор, они временно помещаются в очередь в памяти компьютера. Если дейтаграммы являются частью кратковременного пикового трафика, подобная буферизация позволяет решить проблему перегрузки маршрутизатора. Однако если поток данных продолжает поступать, вся доступная на узле сети или маршрутизаторе память рано или поздно исчерпается. При этом все “лишние” дейтаграммы будут попросту утеряны. При возникновении подобной ситуации маршрутизатор шлет отправителю ICMP-сообщение о *подавлении источника данных* (*source quench*). Этим сообщением маршрутизатор “просит” отправителя уменьшить генерируемый им поток данных. Обычно при перегрузке маршрутизатора отправителю посыпается сообщение о подавлении источника данных для каждой утерянной дейтаграммы. В современных маршрутизаторах применяются более сложные алгоритмы обработки ситуаций перегрузки. В частности, в некоторых маршрутизаторах измеряется скорость входящего потока данных, и если объем трафика превышает заданный уровень, узлом сети, генерирующим максимальный трафик, посыпается сообщение о подавлении источника данных. В других системах предпринимается попытка избежать перегрузки маршрутизатора в целом. При этом сообщения о подавлении источника данных рассыпаются как только размеры очереди становятся угрожающими, но до возникновения перегрузки устройства.

Следует заметить, что не существует ICMP-сообщений, отменяющих подавление источника данных. Узел сети должен снижать уровень трафика, отсылаемый в заданном направлении, до тех пор, пока ему не перестанут приходить сообщения о подавлении источника данных, работающего в этом направлении. После этого он может постепенно увеличивать трафик до тех пор, пока снова не начнут приходить сообщения о подавлении источника данных.

**Печатай файл этой страницы отдельно**

Преимущество системы переадресации протокола ICMP — в ее простоте. При начальной загрузке узлу сети достаточно сообщить только адрес одного маршрутизатора, находящегося в локальной сети. Как только узел сети отправит маршрутизатору дейтаграмму, для которой существует другой, оптимальный маршрут, узлу будет послано соответствующее ICMP-сообщение. Таким образом, удается сохранить минимальный размер таблицы маршрутизации и при этом поместить в нее информацию об оптимальных маршрутах ко всем возможным получателям.

Однако сообщения о переадресации не могут в общем случае решить проблему распространения информации о маршрутах дейтаграмм. Причина состоит в том, что маршрутизатор может отправлять такие сообщения только тем узлам, которые находятся в одной с ним физической сети. Эта проблема проиллюстрирована на рис. 9.5. Предположим, что машина *S* посыпает дейтаграммы машине *D*. Предположим также, что маршрутизатор *R*<sub>1</sub> некорректно выполняет маршрутизацию и вместо устройства *R*<sub>4</sub> пересыпает дейтаграммы устройству *R*<sub>2</sub> (т.е. *R*<sub>1</sub> выбирает более длинный маршрут). После того, как устройство *R*<sub>5</sub> получит дейтаграмму, оно не сможет отправить ICMP-сообщение о переадресации устройству *R*<sub>1</sub>, поскольку адрес последнего ему неизвестен. Проблема распространения информации о маршрутах дейтаграмм по нескольким сетям будет описана в следующих главах.

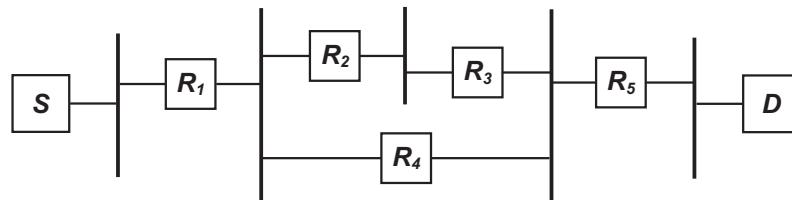


Рис. 9.5. ICMP-сообщения не могут решить проблему обмена информации между маршрутизаторами о путях следования дейтаграмм. В представленном примере устройство *R*<sub>5</sub> не сможет отправить ICMP-сообщение о переадресации устройству *R*<sub>1</sub>, чтобы сократить путь следования дейтаграмм от узла *S* к узлу *D*

Кроме трех стандартных полей (типа, кода и контрольной суммы), в каждом сообщении о переадресации указывается 32-разрядное поле IP-адреса маршрутизатора, а также начальный фрагмент дейтаграммы (рис. 9.6).

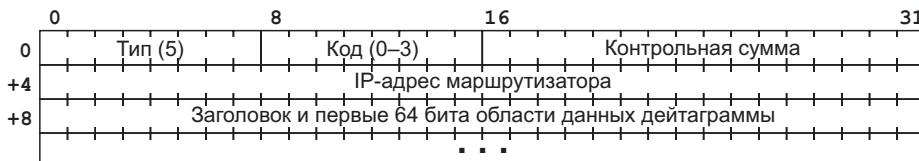


Рис. 9.6. Формат ICMP-сообщения о переадресации

В поле IP-адреса маршрутизатора указывается адрес устройства, которому узел сети должен переслать дейтаграмму при отправке ее конечному получателю, адрес которого указан в заголовке. В ICMP-сообщение о переадресации включается также заголовок дейтаграммы, посланной по неоптимальному маршруту, и следующие за ним 64 бита данных. Таким образом, получив ICMP-сообщение о переадресации, узел сети анализирует начальный фрагмент содержащийся в нем дейтаграммы и определяет адрес ее конечного получателя. В поле кода ICMP-сообщения указывается число, предназначенное для определения способа интерпретации адреса получателя (табл. 9.3).

**Таблица 9.3. Значения кодов интерпретации адреса получателя**

Код	Описание
0	Переадресовывать дейтаграммы для указанной сети (устарело)
1	Переадресовывать дейтаграммы для указанного узла сети
2	Переадресовывать дейтаграммы с указанным типом обслуживания* и для указанной сети
3	Переадресовывать дейтаграммы с указанным типом обслуживания и для указанного узла сети

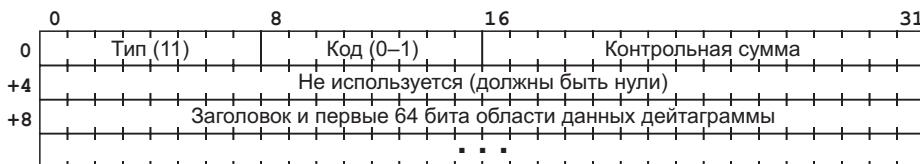
\*.) Напомним, что в каждом заголовке IP-дейтаграммы существует поле, в котором указывается тип обслуживания дейтаграммы. Значение этого поля интерпретируется в процессе маршрутизации.

Итак, можно сказать, что маршрутизаторы отправляют ICMP-сообщения о переадресации только узлам сети, а не другим маршрутизаторам. В следующих главах будут описаны специальные протоколы, с помощью которых маршрутизаторы обмениваются информацией о маршрутах дейтаграмм.

## 9.12. Обнаружение замкнутых и слишком длинных маршрутов

Поскольку маршрутизаторы объединенной сети определяют адрес ближайшей точки перехода на основании данных, находящихся в локальных таблицах маршрутизации, ошибки в них могут привести к *зацикливанию маршрутов (routing cycle)* следования дейтаграмм в направлении некоторых получателей D. Зацикливание маршрутов может произойти на участке сети между двумя и более маршрутизаторами. При этом маршрутизаторы будут передавать дейтаграммы, следующие в направлении получателей D, друг другу (или по кругу, если в цикл вовлечено больше двух маршрутизаторов). Если дейтаграмма попадает в замкнутый цикл маршрутизации, она будет передаваться по нему бесконечно. Как было отмечено в предыдущих главах, для предотвращения явления зацикливания в объединенной сети на основе протокола TCP/IP в заголовке дейтаграмм предусмотрено специальное поле времени жизни, которое иногда называют *счетчиком переходов (hop count)*. Обработав дейтаграмму, маршрутизатор уменьшает на единицу значение счетчика ее времени жизни. Когда значение счетчика становится равным нулю, маршрутизатор удаляет дейтаграмму из сети.

Маршрутизатор может удалить дейтаграмму из сети по двум причинам: при достижении счетчиком времени жизни нулевого значения и при истечении интервала времени, установленного для ожидания получения всех фрагментов дейтаграммы. В обоих случаях маршрутизатор посыпает отправителю дейтаграммы специальное ICMP-сообщение об истечении времени ожидания, формат которого показан на рис. 9.7.



В поле кода ICMP-сообщения об истечении времени ожидания помещается значение 0 или 1, которое помогает отправителю установить причину ошибки (табл. 9.4).

**Таблица 9.4. Значения кодов ошибки сообщения об истечении времени ожидания**

Код	Описание
0	Исчерпано значение счетчика времени жизни
1	Истекло время ожидания при получении фрагментов дейтаграммы

Напомним, что процесс сборки фрагментов выполняется конечным получателем дейтаграммы. При поступлении первого фрагмента дейтаграммы получатель запускает специальный таймер. Если значение таймера истечет до того, как будут получены все фрагменты дейтаграммы, считается, что возникла ошибочная ситуация. В этом случае отправителю дейтаграммы посыпается соответствующее ICMP-сообщение, в поле кода которого помещается значение 1. Такое сообщение отправляется каждый раз при возникновении подобной ошибки.

### 9.13. Уведомление об остальных проблемах

Если маршрутизатор или узел сети обнаруживают проблему, связанную с доставкой дейтаграммы, не описанную в предыдущих разделах (например, если у дейтаграммы оказался некорректный заголовок), то ее отправителю посыпается сообщение об ошибке в параметрах. Одна из возможных причин возникновения подобной ошибки — указание некорректных параметров дейтаграммы. Сообщение, формат которого показан на рис. 9.8, посыпается отправителю дейтаграммы только в том случае, если ошибка настолько серьезна, что дейтаграмма должна быть удалена из сети.

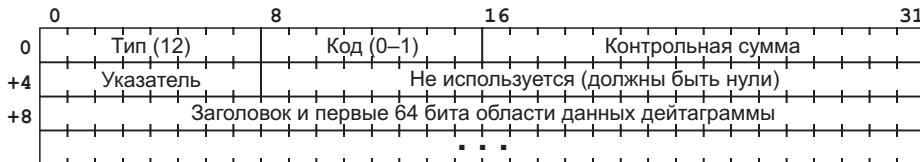


Рис. 9.8. Формат ICMP-сообщения об ошибке в параметрах дейтаграммы. Подобное сообщение посыпается отправителю только в случае удаления дейтаграммы из сети

Чтобы помочь отправителю дейтаграммы конкретизировать проблему, в заголовок ICMP-сообщения об ошибке было введено специальное поле указателя. В него помещается смещение октета исходной дейтаграммы, значение которого вызвало ошибку. Значение поля кода равное 1 указывает на то, что отсутствует нужный параметр дейтаграммы (например, параметр безопасности при работе в защищенной среде). Следует заметить, что значение поля указателя не используется в том случае, если значение поля кода равно 1.

### 9.14. Синхронизация часов и оценка времени передачи

Хотя все компьютеры в объединенной сети могут взаимодействовать между собой, обычно они функционируют независимо друг от друга. При этом текущее время на каждом компьютере может быть разным. Подобная разница в представлении времени может сбить с толку пользователей распределенных программных

систем. Поэтому для синхронизации часов в семейство протоколов TCP/IP включено несколько протоколов. Один из простейших способов синхронизации времени — запросить текущее время у другого компьютера сети с помощью ICMP-сообщения. При этом машина отправителя посыпает другому компьютеру специальное ICMP-сообщение, которое запрашивает у него *временную метку* (*timestamp request*). В ответе на запрос другой компьютер должен присыпать отправителю значение своего текущего времени, помещенное в специальное ICMP-сообщение. Форматы запроса временной метки и ответа на него приведены на рис. 9.9.



Рис. 9.9. Формат ICMP-сообщения, запрашивающего временную метку, и ответ на него

В поле типа рассматриваемого ICMP-сообщения помещается значение 13 (для запроса временной метки) или значение 14 (для ответного сообщения). Поля идентификатора и порядкового номера используются отправителем сообщения для сопоставления запросов и полученных на них ответов. В оставшихся трех полях указываются значения текущего времени, выраженные в миллисекундах, прошедших после полуночи по всемирному времени (Universal Time)<sup>3</sup>. Прежде чем послать пакет получателю, отправитель помещает в поле *времени отправления* запроса свое текущее время. При получении пакета компьютер помещает в поле *времени получения* запроса свое текущее время. И непосредственно при посыпке ответа отправителю в поле *времени отправления* запроса снова помещает-ся текущее время.

Описанные выше три временных метки используются узлами сети для вычисления задержки распространения пакета по сети, что позволяет точно синхронизировать их часы. Поскольку в ответном сообщении содержится время посыпки запроса, отправитель может легко вычислить общее время, затрачиваемое на передачу запроса получателю, формирования ответа на него и обратную пересыпку. Так как в ответном сообщении содержится как время получения запроса удаленным компьютером, так и время отправки ответа на него, отправитель может легко определить время передачи пакета по сети и исходя из него оценить разницу во времени между локальным и удаленным компьютерами.

На практике часто бывает сложно оценить время передачи запроса по сети и возвращения на него ответа, что существенно снижает эффективность использования ICMP-сообщений, запрашивающих временные метки. Для точного определения времени прохождения пакетов можно выполнить несколько измерений и усреднить полученный результат. Однако задержка прохождения пакетов между двумя машинами, подключенными к большой объединенной сети, часто колеблется в очень широких пределах, причем в течение короткого промежутка времени. Более того, поскольку протокол IP не гарантирует доставку пакетов, при прохождении по сети дейтаграммы могут быть аннулированы, задержаны или доставлены в порядке, отличном от исходного. Таким образом, проведение нескольких измерений еще не гарантирует точности полученного результата. Для правильной оценки результатов измерений необходимо прибегнуть к сложным методам статистического анализа.

<sup>3</sup> Всемирное время соответствует времени нулевого (Гринвичского) меридиана.

## 9.15. Информационные запросы и ответы

Информационные запросы и ответы протокола ICMP (типы сообщений 15 и 16) считаются устаревшими и не должны использоваться в современном программном обеспечении. Они были разработаны для того, чтобы узел сети мог определить свой IP-адрес при начальной загрузке. В дальнейшем для этих целей были разработаны протоколы RARP (см. главу 6, “Определение IP-адреса при начальной загрузке (RARP)”) и BOOTP (см. главу 23, “Начальная загрузка и автоконфигурация (BOOTP, DHCP)”).

## 9.16. Определение маски подсети

О том, что такое подсеть, как она работает и зачем она нужна, будет описано в главе 10, “Бесклассовая адресация и подсети (CIDR)”. А пока достаточно просто понять, что, если для адресации узла используется концепция подсети, часть битов поля `hostid` IP-адреса идентифицирует физическую сеть. Таким образом, чтобы можно было использовать адресацию подсетей, узел сети должен “знать”, какая часть битов его 32-разрядного IP-адреса идентифицирует физическую сеть, а какая — узел в этой сети. Эту информацию кодируют в виде 32-разрядного числа, называемого *маской подсети* (*subnet mask*).

Чтобы определить маску подсети, которая используется в некоторой локальной сети, компьютер должен послать запрос (ICMP-сообщение) соответствующему маршрутизатору и получить ответное сообщение, содержащее маску. При отправке запроса компьютер может послать сообщение либо напрямую маршрутизатору, если он знает его IP-адрес, либо послать это же сообщение в широковещательном режиме, если адрес маршрутизатора неизвестен. Формат запроса маски подсети показан на рис. 9.10.

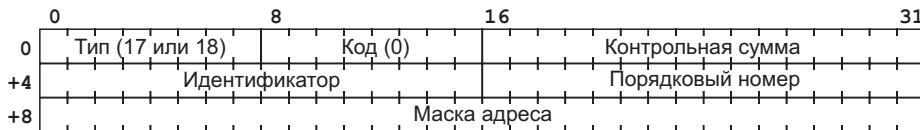


Рис. 9.10. Форматы ICMP-запроса маски подсети и ответного сообщения. Обычно запрос на маску подсети посыпается в широковещательном режиме, поскольку адрес маршрутизатора, который должен прислать ответ, еще не известен

В поле типа ICMP-сообщения указывается код 17, если это запрос на маску подсети, либо код 18, если это ответное сообщение. Маска адреса подсети помещается в соответствующее поле ответного сообщения. Назначение полей идентификатора и порядкового номера традиционное. Они позволяют отправителю запроса сопоставить присланный на него ответ.

## 9.17. Поиск маршрутизатора

Для того чтобы после выполнения начальной загрузки компьютер смог посылать дейтаграммы получателям из других сетей, он должен знать адрес хотя бы одного маршрутизатора, расположенного в локальной сети, к которой он подключен. В протоколе ICMP предусмотрен специальный механизм *поиска маршрутизатора*, который позволяет узлу сети обнаружить адрес ближайшего к нему маршрутизатора.

Следует заметить, что этот механизм поиска адреса маршрутизатора — не единственный. В главе 23, “Начальная загрузка и автоконфигурация (BOOTP, DHCP)”, описаны два других протокола — BOOTP и DHCP, каждый из которых

позволяет узлу сети получить во время начальной загрузки адрес стандартного маршрутизатора, а также другую необходимую информацию. Однако следует отметить, что протоколы BOOTP и DHCP имеют серьезный недостаток: они возвращают из базы данных только ту информацию, которая занесена туда вручную сетевым администратором. А это значит, что информация в базе данных не может меняться достаточно оперативно.

Естественно, что в некоторых случаях адрес маршрутизатора можно прописать статично в один из файлов конфигурации, и это будет работать очень хорошо. В качестве примера рассмотрим сеть, которая имеет выход во внешний мир через единственный маршрутизатор. В этом случае узлу сети нет необходимости динамически определять адреса маршрутизаторов или на ходу изменять маршруты дейтаграмм. Хотя, если в сети для выхода во внешний мир предусмотрено несколько маршрутизаторов, то полученный узлом во время начальной загрузки стандартный маршрут может оказаться недействующим в случае, если маршрутизатор, принятый по умолчанию, выйдет из строя. Более того, об отказе маршрутизатора узлу сети ничего не будет известно.

Механизм поиска маршрутизаторов протокола ICMP имеет два преимущества. Во-первых, он позволяет узлу сети получать информацию непосредственно от самого маршрутизатора, а не из статичных файлов конфигурации при начальной загрузке. Во-вторых, в нем используется технология систем с *неустойчивым состоянием* (*soft state*) на основе таймеров. Это позволяет предотвратить получение узлом сети неправильной информации при выходе маршрутизатора из строя. Маршрутизаторы периодически рассылают узлам сети информацию о себе. Если эта информация не будет получена в течение установленного интервала времени, узел сети считает, что данный маршрутизатор более недоступен. Формат ICMP-сообщения, которое присылают маршрутизаторы, показан на рис. 9.11.

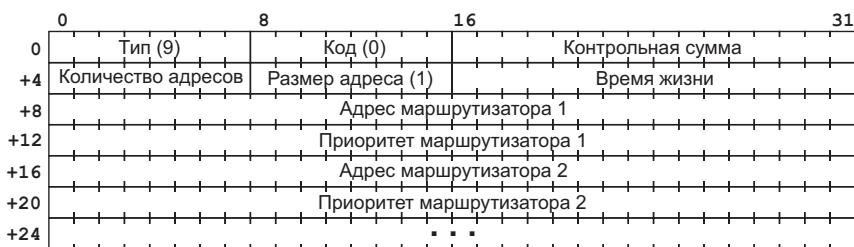


Рис. 9.11. Формат ICMP-сообщения, используемого в протоколе IPv4 для извещения узлов сети об адресах маршрутизаторов. Такие сообщения периодически рассылаются самими маршрутизаторами

Кроме уже знакомых нам полей типа, кода и контрольной суммы, в рассматриваемом ICMP-сообщении находятся еще три новых поля. Количество адресов маршрутизаторов (обычно 1), находящихся в сообщении, указывается в одноименном поле. В поле размера адреса указывается количество блоков по 32 бита, которое занимает поле адреса. Для протокола IPv4 в этом поле указывается значение 1. В поле времени жизни указывается время в секундах, в течение которого узел сети может использовать полученную информацию о маршрутизаторах. Стандартное значение времени жизни равно 30 мин., а рассылка информации о маршрутизаторах по умолчанию выполняется каждые 10 мин. Таким образом, даже если одно из ICMP-сообщений об адресе маршрутизатора не будет получено узлом сети, информация об адресе маршрутизатора останется действительной еще по меньшей мере 20 мин.

В остальной части сообщения находятся пары чисел (их количество указывается в поле *Количество адресов*), первое из которых соответствует IP-адресу маршрутизатора, а второе определяет приоритет данного маршрутизатора (или

**Печатай файл этой страницы отдельно**

В протоколе межсетевых управляющих сообщений (ICMP) предусмотрены специальные средства взаимодействия между маршрутизаторами и узлами сети. Этот протокол является неотъемлемой и обязательной частью протокола IP. В нем определен формат нескольких служебных сообщений. Основными из них являются:

- сообщение о *подавлении источника данных* (*source quench*), позволяющее уменьшить интенсивность потока данных, генерируемых отправителем;
- сообщение о *переадресации* (*redirect*), предписывающее узлу сети изменить информацию, находящуюся в его таблице маршрутизации;
- *запрос эха* и ответ на него, позволяющие узлу сети проверить связь с получателем;
- запрос *адреса маршрутизатора* и ответ на него, позволяющие узлу сети динамически изменять стандартный маршрут следования дейтаграмм.

Перед отправкой ICMP-сообщения помещаются в область данных IP-дейтаграммы. В начале каждого ICMP-сообщения находится заголовок, содержащий три стандартных поля фиксированной длины: *тип сообщения*, *код сообщения* и *контрольную сумму*. Значение поля типа определяет формат остальной части сообщения и его назначение.

## Материал для дальнейшего изучения

Общие сведения об управляющих сообщениях и их взаимосвязь с различными сетевыми протоколами приведены в книгах Таненбаума (Tanenbaum) [127] и Столлингса (Stallings) [122]. Авторы сосредотачивают свое внимание не на том, как послать управляющее сообщение, а на том, когда это следует делать. В работах Гренджа (Grange) и Гина (Gien) [46], а также Драйвера (Driver), Хоупвелла (Hopewell) и Яквинто (Iaquinto) [45] рассматриваются проблемы, для решения которых большую роль играют управляющие сообщения (речь идет об управлении потоками данных). Аналитический обзор методов управления потоками данных приведен в статье Герла (Gerla) и Клейнрока (Kleinrock) [56]. Протоколы синхронизации времени описаны Миллсом в [RFC 956, 957 и 1305].

Протокол ICMP, о котором идет речь в этой главе, является стандартным протоколом семейства TCP/IP. Впервые он был описан Постелом (Postel) в [RFC 792] и дополнен Браденом (Braden) в [RFC 1122]. Сообщения ICMP, предназначенные для подавления источника данных, описаны Нейглом (Nagle) в [RFC 896]. Там же описаны методы обработки маршрутизаторами возникающих в сети перегрузок. Самые последние сведения, относящиеся к технологии обработки маршрутизаторами сообщений о подавлении источника данных, приведены Пру (Prue) и Постелом в [RFC 1016]. В статье Нейгла [90] высказывается мнение, что в сетях с коммутацией пакетов перегрузки неизбежны. Запросы на маску подсети, и ответы на них обсуждаются Могулом (Mogul) и Постелом в [RFC 950]. Поиск маршрутизаторов, запросы на адрес маршрутизатора и ответы на них описаны Дирингом (Deering) в [RFC 1256]. Взаимодействие маршрутизаторов с программами протоколов транспортного уровня, позволяющее избежать перегрузок в сети, описано в отчете Джайна (Jain), Рамакришнана (Ramakrishnan) и Хиу (Chiu) [68].

## Упражнения

- 9.1. Продумайте методику эксперимента, с помощью которого можно подсчитать общее количество ICMP-сообщений каждого типа, проходящих за сутки по вашей локальной сети.

- 9.2.** Проверьте экспериментально, может ли узел сети генерировать такой поток данных, который бы вызвал перегрузку маршрутизатора и появление ICMP-сообщения о подавлении источника данных.
- 9.3.** Разработайте алгоритм синхронизации времени с помощью ICMP-сообщений, содержащих временную метку.
- 9.4.** Проверьте, доступна ли на вашем локальном компьютере команда ping. Как эта программа взаимодействует с протоколами операционной системы? В частности, предоставляет ли операционная система средства, позволяющие любому пользователю создать аналог команды ping, или для этого нужны особые права доступа? Поясните свой ответ.
- 9.5.** Предположим, что в ответ на посылку дейтаграммы, все маршрутизаторы присыпают ICMP-сообщения об истечении времени ожидания, которые программы протокола TCP/IP вашего компьютера пересыпают прикладной программе. Воспользуйтесь этой возможностью для создания команды traceroute, выводящей список адресов всех маршрутизаторов, расположенных между машиной отправителя и конечного получателя.
- 9.6.** Если вы подключены к глобальной сети Internet, попытайтесь с помощью команды ping проверить связь с машиной 128.10.2.1, находящейся в университете Пердью (Purdue University).
- 9.7.** Должен ли маршрутизатор устанавливать для ICMP-сообщений более высокий приоритет, чем для обычных сообщений? Поясните свой ответ.
- 9.8.** Предположим, что к сети Ethernet подключен один обычный узел  $H$  и 12 маршрутизаторов. Опишите такой формат фрейма (слегка нарушающий стандарт), при котором посланный узлом  $H$  единственный IP-пакет, приведет к получению узлом  $H$  24 пакетов.
- 9.9.** Сравните ICMP-сообщение, подавляющее источник данных, с однобитовой системой Джейна, используемой в DECNET. Какая из двух методик обработки перегрузки в сети, на ваш взгляд, более эффективна? Поясните свой ответ.
- 9.10.** Поясните, почему в протоколе ICMP не предусмотрено сообщений, информирующих отправителя о том, что из-за ошибок при передаче данных полученная дейтаграмма была повреждена?
- 9.11.** При каких условиях описанный в предыдущем примере тип сообщений может оказаться полезным?
- 9.12.** Должны ли ICMP-сообщения содержать временные метки, указывающие на то, когда они были посланы? Поясните свой ответ.
- 9.13.** Если маршрутизаторы вашего сетевого центра рассыпают о себе ICMP-сообщения, подсчитайте, сколько адресов рассыпает каждый маршрутизатор по каждому сетевому интерфейсу.
- 9.14.** Попытайтесь обратиться к несуществующему узлу вашей локальной сети. Попытайтесь также связаться с несуществующим узлом удаленной сети. В каком из случаев вашему компьютеру будет прислано сообщение об ошибке? Поясните свой ответ.
- 9.15.** Попытайтесь с помощью команды ping проверить связь с широковещательным адресом вашей локальной сети. Сколько при этом компьютеров ответит? Обратитесь к документации по сетевым протоколам и определите, как должен реагировать узел сети на широковещательный запрос? Обязан ли он всегда отвечать на запрос, или ему запрещено это делать? Может быть в стандарте узлу сети рекомендовано (или не рекомендовано) отвечать на подобные запросы?

# 10

## *Бесклассовая адресация и подсети (CIDR)*

### **10.1. Введение**

В главе 4, “Классовая адресация”, рассматривались оригинальная система адресации протокола IP, а также первые три формы IP-адресов. В этой главе речь пойдет о пяти расширениях системы адресации протокола IP, предназначенных для экономного использования выделенного адресного пространства. Мы рассмотрим причины введения каждого из расширений, а также опишем используемые ими механизмы. В частности, будет подробно рассмотрена система адресации подсетей, которая является составной частью современного стандарта TCP/IP, и система бесклассовой адресации, относящаяся к рекомендованным стандартам (*elective standard*).

### **10.2. Обзор основных моментов**

В главе 4, “Классовая адресация”, вы познакомились с принципом адресации, используемым в объединенной сети, а также с основами системы адресации протокола IP. Мы говорили, что каждому узлу объединенной сети назначается уникальный 32-разрядный IP-адрес и что все машины, подключенные к одной физической сети, имеют общий префикс IP-адреса. Разработчики оригинальной системы IP-адресации разделили IP-адрес на две части: префикс, идентифицирующий физическую сеть, и суффикс, идентифицирующий узел в физической сети. Таким образом, из всего сказанного можно сделать важный вывод.

*В оригинальной системе IP-адресации каждой физической сети назначается уникальный сетевой адрес, который является префиксом IP-адреса узлов, подключенных к этой сети.*

Основное преимущество от разделения IP-адреса на две части проявляется при рассмотрении процесса маршрутизации, и, в частности, размеров таблиц маршрутизации. При таком подходе у маршрутизатора появляется возможность хранить в таблице маршрутизации только один элемент для всех машин одной сети, а не один элемент для каждого узла сети. Кроме того, при выполнении маршрутизации нужно проанализировать только сетевую часть IP-адреса.

Напомним, что в оригинальной системе IP-адресации все доступное адресное пространство разбивается на три основных класса, каждый из которых предназначен для использования в сетях с разным количеством узлов. В сетях класса A 32-х разрядный IP-адрес разбивается на 8-разрядный идентификатор сети и 24-

разрядный номер узла. В сетях класса *B* IP-адрес разбивается на 16-разрядный идентификатор сети и 16-разрядный номер узла. В сетях класса *C* идентификатор сети IP-адреса занимает 24-разряда, а номер узла сети — 8 разрядов.

Чтобы до конца разобраться с расширениями системы адресации, описанными в этой главе, важно понимать, что изменение системы адресации и маршрутизации находится в компетенции администрации сетевого центра. Более того, эти изменения должны оставаться незаметными для других сетевых центров. Другими словами, администрация центра может самостоятельно назначать IP-адреса или использовать систему адресации, отличную от стандартной, если соблюдаются перечисленные ниже условия.

- Принятая система адресации согласована со всеми узлами и маршрутизаторами сетевого центра.
- Используемые IP-адреса должны стандартным образом интерпретироваться в других сетевых центрах (т.е. они должны разделяться на префикс сети и суффикс узла).

### 10.3. Принцип минимизации количества адресов сетей

На первый взгляд в оригинальной системе IP-адресации предусмотрены все возможные ситуации, тем не менее в ней есть существенный недостаток. В чем же он проявляется? Как могло так получиться, что разработчики протокола чего-то не учли? Ответ очень прост: всему виной масштабы роста объединенной сети. Первоначально создатели протокола TCP/IP работали в мире дорогостоящих больших ЭВМ, а масштабы объединенной сети оценивались тысячами узлов и сотнями сетей. Разработчики даже не могли себе представить, что через десять лет после создания протокола TCP/IP вдруг появятся десятки тысяч небольших сетей, состоящих из дешевых персональных компьютеров.

Масштабы роста глобальной сети Internet можно оценить по количеству подключавшихся к ней новых сетей. За 9–15 месяцев размер объединенной сети практически удваивался! Огромное количество сетей небольшого размера расстроило четкую картину первоначально созданной структуры объединенной сети, поскольку, во-первых, требовалась большая административная работа по управлению пространством сетевых адресов, во-вторых, размеры таблиц, с которыми работали маршрутизаторы, существенно выросли, и, в-третьих, стало очевидным, что рано или поздно все доступное адресное пространство будет исчерпано<sup>1</sup>. Важность второй проблемы налицо, поскольку маршрутизаторы периодически обмениваются информацией, находящейся в их таблицах, что приводит к увеличению трафика в объединенной сети, а также требует привлечения дополнительных вычислительных мощностей при выполнении маршрутизации. Третья проблема считается одной из самых основных, поскольку в настоящий момент в глобальной сети Internet общее количество сетей превышает возможности оригинальной системы адресации. В частности, уже сейчас префиксов сети класса *B* не хватает на все существующие сети среднего размера. Поэтому вопрос ставится так: как минимизировать общее количество выделенных адресов сетей, особенно тех, что относятся к классу *B*, без изменения принятой 32-разрядной системы адресации?

Для минимизации количества адресов сетей нужно стараться по возможности избегать назначения новых сетевых префиксов и использовать один сетевой

<sup>1</sup> Хотя многие прогнозировали, что адресное пространство протокола IPv4 будет исчерпано уже к 2000 году, сейчас можно сказать, что при правильном подходе к распределению адресов и использовании методик, описанных в этой главе, адресного пространства протокола IPv4 хватит еще как минимум лет на 20, т.е. до 2020 года.

**Печатай файл этой страницы отдельно**

информация кодируется и помещается в неиспользуемые части адреса. Например, сети ARPANET был назначен адрес класса A 10.0.0.0. Каждому узлу коммутации пакетов (packet switch node, или PSN) сети ARPANET присваивается уникальный идентификатор, представляющий собой целое число. Внутри ARPANET IP-адрес, состоящий из 4-х октетов, представляется в виде 10.*p.u.i*, т.е. в виде четырех отдельных октетов, определяющих номер сети (*10*), номер порта узла коммутации пакетов (*p*), к которому подключен получатель, и номер самого коммутатора пакетов (*u*). Октет *i* не используется. Так, адреса ARPANET 10.2.5.37 и 10.2.9.37 обозначают один и тот же порт 2 узла коммутации пакетов 37. Таким образом, если к порту 2 устройства 37 подключить “прозрачный” маршрутизатор, то в октете *u* можно закодировать информацию о компьютере локальной сети, которому предназначена дейтаграмма. Из приведенного примера видно, что пользователи глобальной сети даже не заметят, что к одному порту узла коммутации пакетов подключено несколько компьютеров.

По сравнению с обычной маршрутизацией прозрачная маршрутизация имеет как преимущества, так и недостатки. Основным преимуществом этого метода можно считать то, что благодаря ему снижается количество используемых сетевых адресов, поскольку локальной сети не нужно назначать отдельный префикс IP-адреса. Еще одно преимущество — возможность перераспределения нагрузки между несколькими “прозрачными” маршрутизаторами. Например, если два таких маршрутизатора подключены к одной локальной сети, то между ними можно разделить потоки данных, поступающие к компьютерам локальной сети и от них. Для сравнения: при использовании обычной маршрутизации к абстрактной локальной сети может быть проложен только один маршрут для пакетов.

Один из существенных недостатков метода прозрачной маршрутизации заключается в том, что он работает только в сетях с большим адресным пространством, из которого можно выбрать адреса узлов локальной сети. Таким образом, описанный выше метод чаще всего применяется в сетях класса A и практически не применяется в сетях класса C. Еще один недостаток “прозрачного” маршрутизатора заключается в том, что он не является полноценным маршрутизатором, поскольку не поддерживает весь спектр услуг обычного маршрутизатора. В частности, в сетевом программном обеспечении “прозрачного” маршрутизатора полностью не поддерживается протокол ICMP, а также протоколы сетевого управления, наподобие SNMP. Поэтому “прозрачный” маршрутизатор не возвращает ответ на запрос эха протокола ICMP (т.е. определить с помощью команды ping, работает ли устройство, не так-то просто).

## 10.5. Агенты ARP

Существует еще один метод, позволяющий использовать один IP-префикс в двух физических сетях. Речь идет о так называемом *агенте ARP* (*proxy ARP*). Этот метод еще называют *смешанным ARP* (*promiscuous ARP*) или *усовершенствованным ARP* (*ARP hack*). Такой метод может использоваться только в тех сетях, где для преобразования IP-адресов в физические адреса используется протокол ARP. Принцип работы этого метода лучше всего продемонстрировать на конкретном примере (рис. 10.2).

На рис. 10.2 для двух физически сетей назначен один префикс IP-адреса. Представьте себе, что в некоторой организации имелась физическая сеть, обозначенная на рис. 10.2 как *Основная сеть*. Со временем сеть была расширена — к ней добавили еще один физический сегмент, который на рисунке обозначен как *Скрытая сеть*. Две сети объединены через маршрутизатор *R*, которому известна их топология (т.е. какой из компьютеров к какой из сетей подключен). Для создания иллюзии одной сети используется протокол ARP. Если два компьютера, подключенные к

разным сегментам сети, хотя обмениваться дейтаграммами, маршрутизатор  $R$  выполняет подмену их физических адресов. В результате создается впечатление, что эти компьютеры взаимодействуют по одной физической сети. Предположим, что узел  $H_1$  хочет обмениваться пакетами с узлом  $H_4$ . Для этого сперва он должен преобразовать IP-адрес узла  $H_4$  в физический адрес. Как только этот адрес станет известен, узел  $H_1$  сможет напрямую послать дейтаграмму получателю. Таков алгоритм работы протокола IP.

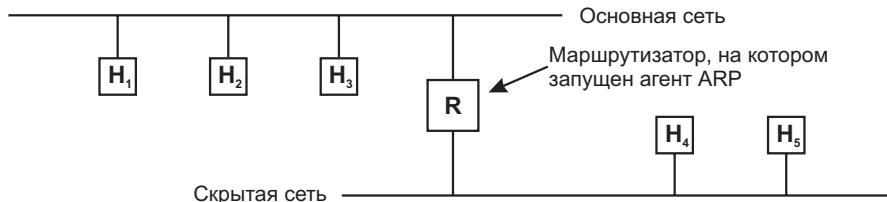


Рис. 10.2. Агент ARP (смешанный ARP) позволяет использовать один префикс IP-адреса в двух физических сетях. Маршрутизатор  $R$  отвечает на ARP-запросы, поступившие от узлов одной сети, на физические адреса узлов другой сети. При этом отправителю запроса возвращается физический адрес маршрутизатора  $R$ , а после получения дейтаграммы выполняется ее корректная маршрутизация между двумя сетями. По сути маршрутизатор  $R$  подменяет физические адреса двух сетей

На маршрутизаторе  $R$  запускается специальная программа (ее называют агентом ARP), которая перехватывает широковещательные ARP-запросы, поступающие от узла  $H_1$ . Так как узел  $H_4$  расположен в другой физической сети, ответ на ARP-запрос отправляет маршрутизатор  $R$ . При этом он указывает в ответном сообщении не физический адрес узла  $H_4$ , а собственный физический адрес. Получив ответ, узел  $H_1$  заносит адресную привязку для узла  $H_4$  в свою таблицу ARP для дальнейшего использования. Таким образом, при отправке дейтаграммы узлу  $H_4$ , она на самом деле будет посыпаться по физическому адресу маршрутизатора  $R$ . Как только подобная дейтаграмма попадает на маршрутизатор  $R$ , последний просматривает специальную таблицу маршрутизации и определяет адрес ее дальнейшей пересылки. Очевидно, что маршрутизатор должен перебрасывать все дейтаграммы, предназначенные узлу  $H_4$  в скрытую сеть. Чтобы компьютеры, подключенные к скрытой сети, могли обмениваться дейтаграммами с компьютерами из основной сети, маршрутизатор  $R$  выполняет описанные выше действия, только в обратном направлении.

Технология агентов ARP может использоваться в маршрутизаторах благодаря одной важной особенности протокола ARP — возможности установки доверительных отношений между компьютерами локальной сети. При этом подразумевается, что все ответы, посылаемые на ARP-запросы, правильные, и поэтому информация в них не проверяется отправителем запроса. Практически все узлы сети заносят в свой локальный кэш адресные привязки, полученные в результате ARP-запроса, без какой-либо проверки их корректности и непротиворечивости. Таким образом, в ARP-таблице вполне может находиться несколько IP-адресов, которым соответствует один и тот же физический адрес. Следует отметить, что подобный случай не является нарушением спецификации протокола ARP.

Некоторые реализации протокола ARP не столь лояльны по отношению к описанной выше ситуации. В частности, существуют реализации протокола ARP, уведомляющие администратора сети о возможных нарушениях в системе безопасности. При обнаружении в ARP-таблице нескольких IP-адресов, которым соответствует один физический адрес, генерируется соответствующее предупреждение. Это сделано для того, чтобы предотвратить возможность *подлога* (*spoofing*) —

ситуации, когда одна из машин выдает себя за другую для того, чтобы перехватить посланные ей пакеты. Таким образом, описанные выше защищенные реализации протокола ARP не могут использоваться в сети, где установлены маршрутизаторы с агентами ARP, поскольку при этом достаточно часто будут генерироваться предупреждающие сообщения о возможном подлоге.

Основным преимуществом агента ARP является то, что он может быть установлен на одном из маршрутизаторов локальной сети. При этом таблицы маршрутизации на других узлах сети и маршрутизаторах остаются неизменными. Таким образом, агент ARP позволяет полностью скрыть особенности физических соединений.

Основной недостаток агента ARP заключается в том, что его нельзя использовать в тех сетях, где для преобразования логических адресов в физические не используется протокол ARP. Кроме того, метод агента ARP нельзя применить к более сложной сетевой топологии (например, когда две физические сети соединены посредством нескольких маршрутизаторов); он также не поддерживает какие бы то ни было формы маршрутизации. В большинстве реализаций агентов ARP используется ручная настройка таблиц соответствия машин и адресов, что делает их применение чрезвычайно громоздким и не исключает появление ошибок.

## 10.6. Адресация подсетей

Третий способ, позволяющий применить один сетевой адрес для нескольких физических сетей, называется *адресацией подсетей (subnet addressing)*, *маршрутизацией подсетей (subnet routing)*, или просто *методом подсетей (subnetting)*. Из описанных методов адресации метод подсетей используется чаще всего, поскольку он самый универсальный и, кроме того, является стандартом. Фактически метод адресации подсетей является неотъемлемой частью системы адресации протокола IP.

Объяснить метод адресации подсетей проще всего на следующем примере. Представьте себе, что некоторому сетевому центру (назовем его автономной системой), состоящему из нескольких физических сетей, выделен блок IP-адресов класса B. При этом топология физических сетей и способы маршрутизации трафика известны только локальным маршрутизаторам центра. Для маршрутизаторов других автономных систем все физические сети данного центра представляются в виде одной большой физической сети класса B. Поэтому они маршрутизируют потоки данных всех физических сетей центра так, как если бы это была действительно одна большая физическая сеть (рис. 10.3).

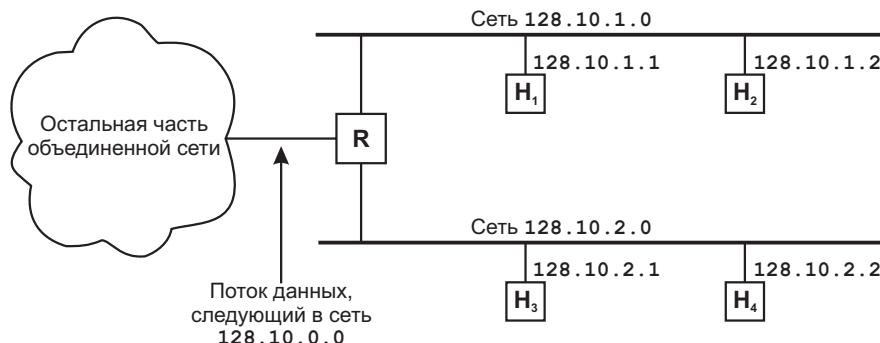


Рис. 10.3. Пример автономной системы с двумя физическими сетями, адресация которых осуществляется методом подсетей. Для узлов сетей назначены IP-адреса класса B. Весь поток данных для сети 128.10.0.0 поступает на маршрутизатор R, который на основании значения третьего октета IP-адреса перераспределяет его между двумя физическими сетями

В нашем примере для двух физических сетей автономной системы выделен один адрес сети класса *B* 128.10.0.0. За исключением маршрутизатора *R*, все остальные маршрутизаторы объединенной сети выполняют маршрутизацию дейтаграмм так, как если бы автономная система состояла из одной большой сети. И только после того, как дейтаграмма достигнет маршрутизатора *R*, он примет решение, по какой из двух физических сетей ее следует доставить получателю. Чтобы сделать процесс выбора нужной сети более эффективным, администрацией автономной системы было принято решение различать физические сети по значению третьего октета IP-адреса. При этом машинам первой сети назначаются IP-адреса вида 128.10.1.*X*, а машинам второй сети — 128.10.2.*X*, где *X* — последний октет адреса, который содержит небольшое целое число, идентифицирующее конкретный узел физической сети. Тогда, чтобы решить, по какой из сетей нужно доставить дейтаграмму получателю, маршрутизатор *R* должен проверить значение третьего октета IP-адреса получателя. Если его значение равно 1, дейтаграмма направляется по сети 128.10.1.0, а если — 2, то по сети 128.10.2.0.

При использовании идеи подсетей лишь незначительно изменяется способ интерпретации IP-адресов. При этом 32-разрядный IP-адрес разбивается не на префикс сети и суффикс узла (как это было в оригинальной системе адресации), а на *сетевую часть* (*network portion*) и *локальную часть* (*local portion*). Сетевая часть интерпретируется точно так же, как и в оригинальной системе адресации (когда подсети не используются). Как и прежде, чтобы внешние автономные системы могли взаимодействовать с данной сетью, маршрут к ней должен быть объявлен (т.е. известен) этим системам. При этом весть поток данных, направляющийся в данную сеть, будет следовать по объявленным маршрутам. Способ интерпретации локальной части сетевого адреса отдан на откуп администрации местной автономной системы (естественно, что при этом не должны нарушаться принятые официально стандарты маршрутизации подсетей). Из всего сказанного выше можно сделать такой вывод.

*При адресации подсетей 32-разрядный IP-адрес разбивается на две части: сетевую и локальную. При этом сетевая часть определяет сетевой центр (или автономную систему), которая может состоять из нескольких физических сетей. Локальная часть сетевого адреса определяет физическую сеть автономной системы и узел в этой сети.*

В примере, показанном на рис. 10.3, метод адресации подсетей применяется к сети класса *B*. При этом сетевая и локальная части IP-адреса занимают по 2 октета. Чтобы сделать процесс маршрутизации дейтаграмм между двумя физическими сетями эффективным, в нашем примере администратор автономной системы решил выделить один октет локальной части IP-адреса для идентификации сети, а другой — для идентификации узла (рис. 10.4).

Адресация подсетей является формой иерархической адресации (*hierarchical addressing*); при ее использовании выполняется так называемая *иерархическая маршрутизация* (*hierarchical routing*), т.е. маршрутизация, осуществляемая автономно на каждом уровне глобальной сети. В рассматриваемом случае на верхнем уровне иерархии маршрутизации (т.е. речь идет о других автономных системах объединенной сети) используются первые два октета IP-адреса. На следующем уровне (т.е. на уровне локальной автономной системы) используется третий октет адреса, и, наконец, на самом нижнем уровне (т.е. при выполнении доставки дейтаграммы по конкретной физической сети) — последний октет IP-адреса получателя.

Иерархическая адресация не является чем-то новым, она использовалась и раньше во многих сложных системах. В качестве примера, пожалуй, можно

назвать телефонную систему США. В ней 10-значный телефонный номер разбивается на три части следующим образом:

- 3-значный междугородный код региона;
- 3-значный номер автоматической телефонной станции (АТС);
- 4-значный номер абонента.



*Рис. 10.4. Абстрактная интерпретация 32-разрядного IP-адреса, принятая в оригинальной системе IP-адресации (а); интерпретация IP-адреса при использовании подсетей для автономной системы, показанной на рис. 10.3 (б). Локальная часть адреса разбивается на две части, определяющие физическую сеть и узел в этой сети*

Основное преимущество иерархической адресации — ее приспособленность к быстрому росту системы. Причина очевидна — отдельно взятому маршрутизатору совсем не обязательно знать все те подробности внутреннего устройства удаленной автономной системы, которые ему нужны для выполнения маршрутизации внутри локальной системы. Одним из недостатков метода является трудность выбора иерархической структуры, которая усугубляется еще и тем, что, приняв такую структуру, изменить ее не так-то просто.

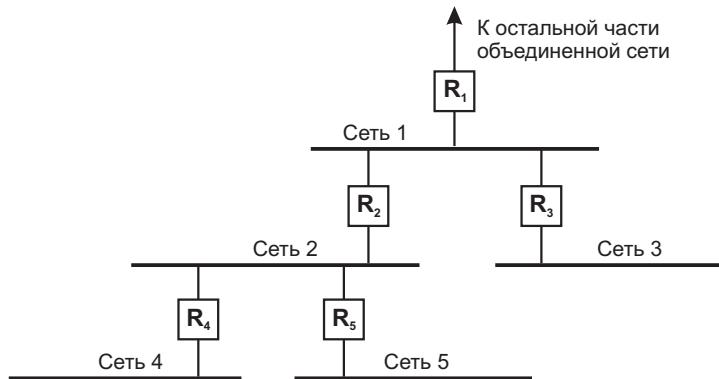
## 10.7. Выбор способа адресации подсетей

В стандарте TCP/IP, описывающем адресацию подсетей, официально признается, что для разных автономных систем нужна своя иерархия адресов. Поэтому в нем предусмотрена определенная гибкость в отношении выбора адресов подсетей. Чтобы понять необходимость такой гибкости, представьте себе автономную систему, состоящую из пяти связанных друг с другом физических сетей (рис. 10.5). Предположим также, что для всей автономной системы выделен блок IP-адресов класса *B*, который должен использоваться для адресации всех узлов физических сетей. Как же разбить локальную часть IP-адреса, чтобы повысить эффективность процесса маршрутизации?

В рассматриваемом примере локальную часть IP-адреса следует разбить с учетом возможности будущего роста всей системы в целом. Разбиение 16-разрядной локальной части на 8-разрядный идентификатор сети и 8-разрядный идентификатор узла, как показано на рис. 10.4, позволяет построить систему, состоящую максимум из 256 сетей, каждая из которых будет содержать до 256 узлов<sup>2</sup>.

<sup>2</sup> На практике этот предел составляет 254 подсети, к каждой из которых можно подключить до 254 узлов. Напомним, что IP-адреса, содержащие в поле идентификатора узла все единицы, являются специальными и не могут быть использованы для адресации реальных узлов.

В табл. 10.1 приведены возможные варианты разбиения локальной части IP-адреса в том случае, если для идентификатора подсети выделяется *фиксированное число разрядов* (как было описано выше). При этом не допускаются номера сетей и узлов, состоящие из всех единиц или всех нулей. Чаще всего под идентификаторы сети и узла выделяют по 8 бит.



*Рис. 10.5. Топология автономной системы, организованной в виде трех “уровней”. В данном случае простейший метод разбиения локальной части IP-адреса на две части, определяющие физическую сеть и узел в ней, не является оптимальным*

**Таблица 10.1. Способы разбиения локальной части IP-адреса класса B**

<i>Размер идентификатора подсети (бит)</i>	<i>Число подсетей</i>	<i>Число узлов в одной подсети</i>
0	1	65534
2	2	16382
3	6	8190
4	14	4094
5	30	2046
6	62	1022
7	126	510
8	254	254
9	510	126
10	1022	62
11	2046	30
12	4094	14
13	8190	6
14	16382	2

ницы и все нули зарезервированы для широковещательной передачи, а идентификаторы сети, состоящие из всех единиц или нулей использовать нельзя, поскольку они предназначены для служебных целей.

Как следует из табл. 10.1, при выборе способа разбиения локальной части IP-адреса нужно достичь определенного компромисса между количеством подсетей и количеством узлов в них. Суть в том, что при увеличении числа физических сетей, сокращается количество узлов в них, и, наоборот, при увеличении числа узлов в физической сети сокращается общее количество сетей. Например, если под поле, идентифицирующее физическую сеть, выделить 3 бита, то в системе может существовать до 6 физических сетей, каждая из которых может содержать до 8190 узлов. Если же размер идентификатора сети будет составлять 12 бит, то общее количество физических сетей может достигать 4094, однако при этом к каждой из таких сетей можно подключить всего 62 узла.

## 10.8. Подсети переменной длины

Выше мы уже говорили о том, что под выбором системы адресации подсетей по сути подразумевается способ разбиения локальной части IP-адреса на две части, определяющие физическую сеть и узел в этой сети. В самом деле, в большинстве автономных систем, где используется адресация подсетей, локальная часть IP-адреса разбивается на две фиксированные части. Должно быть уже понятно, почему разработчики стандарта не включили в него каких-либо жестких рекомендаций по разбиению локальной части IP-адреса, поскольку не существует универсальной схемы, которая подходила бы для всех случаев жизни. Для одних организаций нужно большое количество физических сетей с малым количеством узлов, а для других — наоборот, большое количество узлов при малом количестве сетей. Более того, разработчики прекрасно понимали, что подобные проблемы могут возникнуть не только в разных организациях, но и внутри одной организации. Поэтому, чтобы предоставить пользователям максимум свободы выбора, в стандарте адресации подсетей протокола TCP/IP предусмотрены еще более широкие возможности, чем те, что были описаны выше. Организациям предоставлено право выбирать способ разбиения IP-адреса на части отдельно для каждой сети. Речь идет об использовании *подсетей переменной длины* (*variable-length subnetting*), хотя название этой методики нельзя считать удачным, поскольку оно слегка сбивает с толку. Дело в том, что разбивка локальной части IP-адреса, сделанная для сети один раз, больше не меняется. Более того, о выбранном способе разбиения локальной части IP-адреса должно быть известно всем подключенными к сети узлам и маршрутизаторам. В противном случае не будет работать система маршрутизации, что приведет к потере или зациклинию дейтаграмм. Итак,

*чтобы предоставить пользователям максимум свободы при выборе способа адресации подсетей, в стандарте протокола TCP/IP предусмотрена возможность использования подсетей переменной длины. При этом локальная часть IP-адреса может разбиваться независимо для каждой сети. Однако после того, как система адресации определена, о ней должно стать известно всем машинам, подключененным к этим сетям.*

Основным преимуществом использования подсетей переменной длины является гибкость этого метода и большая свобода выбора для пользователя. В результате в одной организации могут существовать как большие, так и малые сети, при этом выделенное организацией адресное пространство будет использоваться максимально эффективно. Несмотря на очевидные преимущества, использование подсетей переменной длины имеет и серьезные недостатки. Очень важно тщательно контролировать процесс назначения адресов подсетям, чтобы избежать *неоднозначности адреса* (*address ambiguity*) — ситуации, при которой адрес интерпретируется по-разному в каждой из физических сетей. В результате

может оказаться так, что один IP-адрес будет соответствовать двум разным подсетям. В результате из-за неправильного использования сетей переменной длины компьютеры не смогут взаимодействовать между собой. Следует отметить, что маршрутизаторы не смогут решить проблему неоднозначности адресов самостоятельно — она может быть решена только путем повторной нумерации сетей. Поэтому сетевые администраторы стараются избегать использования подсетей переменной длины.

## 10.9. Реализация адресации подсетей с помощью масок

Независимо от используемого метода адресации подсетей (с фиксированной или переменной длиной) выполнить его настройку очень легко. В стандарте указано, что способ разделения IP-адреса задается с помощью 32-разрядной маски. Другими словами, для того, чтобы можно было использовать метод адресации подсетей для каждой из физических сетей автономной системы, нужно задать 32-разрядную *маску подсети (subnet mask)*. Мaska формируется на основе очень простого правила: если бит IP-адреса относится к префиксу подсети, то соответствующий ему бит маски полагается равным 1; если же бит IP-адреса относится к идентификатору узла сети, то соответствующий бит маски полагается равным 0. Давайте рассмотрим в качестве примера следующую маску:

11111111 11111111 11111111 00000000

Она указывает, что первые три октета IP-адреса определяют сеть, а четвертый октет — узел в этой сети. Эта маска соответствует IP-адресу сети класса C. Для сети класса B единицы будут находиться в первых двух октетах маски и, возможно, в одном или нескольких битах последних двух октетов.

Один из интересных моментов в адресации подсетей заключается в том, что в стандарте нигде не сказано, как должны располагаться биты, определяющие сетевую часть IP-адреса (т.е. подряд или нет). Например, для некоторой подсети можно задать такую маску:

11111111 11111111 00011000 01000000

Она указывает, что сетевая часть IP-адреса находится в первых двух октетах, двух битах третьего октета и одном бите четвертого октета. Следует отметить, что при таком способе разбиения можно получить весьма интересные результаты при назначении адресов узлам сети. Однако подобная практика затрудняет анализ IP-адресов узлов сети и таблиц маршрутизации. Поэтому рекомендуется использовать только такие маски подсети, в которых биты, определяющие сетевую часть IP-адреса, расположены подряд. Более того, в рамках одной автономной системы с общим пространством IP-адресов желательно, чтобы во всех физических сетях использовались одинаковые маски подсети.

## 10.10. Способы представления масок подсети

В предыдущем разделе маска подсети представлялась в виде 32-разрядного двоичного числа. Однако такой способ представления нельзя назвать удачным из-за неудобства работы с двоичными числами, который к тому же сопряжен с ошибками. Поэтому в программах предусматривают иные формы определения масок. Иногда формат представления масок совпадает с форматом представления двоичных чисел, принятым в конкретной версии операционной системы (например, используется шестнадцатеричная форма записи).

В большинстве программ, работающих с протоколом IP, для представления масок подсетей используется точечная десятичная форма записи. Больше всего

она подходит для тех сетевых центров, в которых для идентификации физической сети и узлов используются целые октеты IP-адреса. Например, как уже было сказано, в большинстве автономных систем, для которых выделен блок адресов класса *B*, третий октет IP-адреса используется для идентификации физической сети, а четвертый — для идентификации узла этой сети. В подобных случаях маска подсети, представленная в точечной десятичной форме записи, будет иметь вид 255.255.255.0. Как видите, такое число легко запомнить и расшифровать.

В литературе можно также встретить примеры записей, когда адрес подсети, ее маска и идентификатор узла задают в виде трех чисел (триад), заключенных в фигурные скобки, как показано ниже. Такая форма записи называется *троичной* (*3-tuple*).

```
{ <Адрес сети> , <адрес подсети> , <адрес узла> }
```

В троичной форме записи число -1 означает “все единицы”. Например, если маска подсети для сети класса *B* — 255.255.255.0, то в троичной форме записи ее можно представить как {-1, -1, 0}.

Основным недостатком троичной формы записи является то, что она не позволяет точно указать, какое количество бит используется в каждой из частей IP-адреса. Достоинством этой формы записи можно считать то, что она абстрагируется от двоичного представления масок и IP-адресов и акцентирует внимание пользователя на трех составляющих частях адреса. Чтобы понять, почему иногда предпочтительнее использовать численные значения адресов, а не их двоичные представления, давайте рассмотрим следующую триаду:

```
{ 128.10, -1, 0 }
```

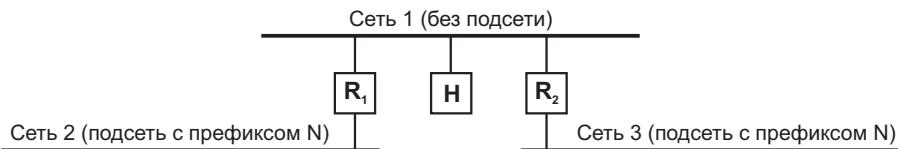
Она представляет IP-адрес, сетевая часть которого равна 128.10, в поле идентификатора подсети находятся все единицы, а в поле идентификатора узла — все нули. Для представления того же адреса в другом виде нам потребуется 32-разрядная маска подсети и 32-разрядный IP-адрес. Кроме того, читателю потребуется какое-то время для декодирования значений битовых полей, прежде чем он сможет их проанализировать. Таким образом, троичная форма записи не зависит от класса IP-адреса и размера битового поля, идентифицирующего подсеть. Поэтому она может использоваться для представления адресов при описании трудных для восприятия понятий. Рассмотрим, например следующую триаду:

```
{ <Адрес сети>, -1, -1 }
```

Она представляет адреса, в которых находится корректный префикс сети, а в полях идентификаторов подсети и узла заданы все единицы. Примеры триад будут приведены далее в этой главе.

## 10.11. Маршрутизация при наличии подсетей

Для того чтобы сделать возможной маршрутизацию при наличии подсетей, стандартный алгоритм маршрутизации протокола IP необходимо немного видоизменить. На всех узлах и маршрутизаторах, подключенных к физическим сетям, в которых используется метод адресации подсетей, должен использоваться видоизмененный алгоритм, называемый маршрутизацией подсетей (*subnet routing*). При этом за кадром остается один момент: если не учитывать ограничений, сделанных для того, чтобы работал механизм адресации подсетей, то придется использовать алгоритм маршрутизации подсетей также на всех других узлах и маршрутизаторах данной автономной системы. Чтобы понять суть проблемы, не вникая в эти ограничения, давайте рассмотрим пример адресации сетей, показанных на рис. 10.6.



*Рис. 10.6. Пример некорректной адресации трех сетей, когда в сетях 2 и 3 используется один префикс IP-адреса N. Если предположить, что такая адресация допустима, то для того, чтобы узел H мог обмениваться дейтаграммами с узлами сетей 2 и 3, на нем должен работать алгоритм маршрутизации подсетей, хотя в сети 1 адресация подсетей не используется*

Как следует из рис. 10.6, в сетях 2 и 3 некорректно используется адресация подсетей, поскольку им назначен один префикс IP-адреса  $N$ . Хотя узел  $H$  не подключен напрямую к сети, в которой используется адресация подсетей, тем не менее на нем должен использоваться алгоритм маршрутизации подсетей, для определения, какому из двух маршрутизаторов —  $R_1$  или  $R_2$  — следует посыпать дейтаграммы, предназначенные для сети  $N$ . На первый взгляд, проблема решается просто — достаточно, чтобы узел  $H$  направлял дейтаграммы одному из маршрутизаторов (они наверняка выполнят маршрутизацию, как надо). Однако такое решение не является оптимальным, поскольку не все потоки данных будут следовать до получателя кратчайшим путем. В случае больших систем отличия между оптимальным и неоптимальным маршрутом будут куда более существенными.

Теоретически существует простое правило, позволяющее определить, когда на узлах сети следует использовать алгоритм маршрутизации подсетей. Оно гласит:

*чтобы оптимизировать процесс маршрутизации, на машине  $H$  должен использоваться алгоритм маршрутизации подсетей для всех узлов с префиксом IP-адреса  $N$ , за исключением случая, когда существует единственный кратчайший маршрут  $P$  между узлом  $H$  и узлом физической сети с префиксом  $N$ .*

К сожалению, применение приведенного теоретического правила мало чем помогает при назначении адресов подсетей. Во-первых, кратчайшие пути могут изменяться в случае отказа сетевого оборудования или переадресации потоков данных из-за возникших в сети перегрузок. Подобные динамические изменения, происходящие в реально действующих сетях, не позволяют применять сформулированное выше правило, за исключением очень простых случаев. Во-вторых, правило маршрутизации подсетей неприменимо за пределами автономной системы, поскольку при этом возникают различного рода сложности в процессе распространения масок подсетей. Информацию о маршрутах к подсетям невозможно распространить за пределы автономной системы, поскольку в используемых для этой цели протоколах (они будут рассмотрены ниже) такая возможность не предусмотрена. На практике бывает крайне трудно распространить информацию о подсети за пределы данной физической сети. По этой причине создатели протокола рекомендуют по возможности упрощать используемую в пределах автономной системы адресацию подсетей. В частности, сетевые администраторы должны всегда придерживаться сформулированных ниже правил.

*Все подсети, которым назначен один префикс IP-адреса, должны быть смежными. Во всех подсетях должны использоваться унифицированные маски, а все машины должны поддерживать алгоритм маршрутизации подсетей.*

Эти правила трудновыполнимы в случае больших организаций, имеющих несколько автономных систем, связанных между собой через Internet, но не

имеющих непосредственного соединения друг с другом. В таких организациях нельзя использовать подсети, которым назначен один префикс IP-адреса, поскольку сами физические сети не являются смежными.

## 10.12. Алгоритм маршрутизации подсетей

Как и в стандартном алгоритме маршрутизации протокола IP, в алгоритме маршрутизации подсетей также используются таблицы. Напомним, что в стандартном алгоритме существует два особых случая, которые должны проверяться явным образом: это маршрутизация отдельных узлов сети и наличие стандартного маршрута. Во всех остальных случаях поиск маршрута осуществляется с помощью таблицы маршрутизации, которая содержит элементы следующего вида:

(адрес сети, адрес ближайшей точки перехода)

В качестве адреса сети указывается IP-адрес сети получателя  $N$ , а адресом ближайшей точки перехода является IP-адрес маршрутизатора, которому следует перенаправлять дейтаграммы, посылаемые в направлении сети  $N$ . При выполнении стандартного алгоритма маршрутизации сетевая часть IP-адреса получателя сравнивается со значением поля адреса сети каждого элемента таблицы маршрутизации, пока не будет найдено соответствие. Поскольку по определению в качестве *адреса ближайшей точки перехода* может быть задан только адрес машины, подключенной напрямую к сети, в которой находится отправитель дейтаграммы, то обычно выполнить поиск в таблице маршрутизации нужно только один раз.

При выполнении стандартного алгоритма маршрутизации легко определить, где заканчивается сетевая часть IP-адреса и начинается локальная, поскольку тип адреса и его формат (т.е. к какому из классов —  $A$ ,  $B$ ,  $C$  или  $D$  — он принадлежит) закодированы в первых трех битах IP-адреса получателя. При использовании подсетей по виду IP-адреса невозможно определить, какие из битов идентифицируют сеть получателя, а какие — узел в ней. Поэтому при использовании видоизмененного алгоритма маршрутизации подсетей в таблицу маршрутизации заносится дополнительная информация. Каждый элемент этой таблицы содержит дополнительное поле. В него помещается маска подсети, используемая в сети получателя, соответствующей этому элементу, как показано ниже:

(маска подсети, адрес сети, адрес ближайшей точки перехода)

При выборе маршрута с помощью видоизмененного алгоритма, с помощью маски подсети из IP-адреса получателя извлекаются биты адреса сети получателя, а затем производится их сравнение со значением поля адреса сети элемента таблицы маршрутизации. Другими словами, вначале выполняется операция по разрядному логического И над 32-разрядным значением IP-адреса получателя и *маской подсети*, извлеченной из текущего элемента таблицы маршрутизации. Затем полученный результат сравнивается с *полем адреса сети* этого же элемента таблицы маршрутизации. Если они совпадают, дейтаграмма пересыпается по адресу, указанному в поле *адреса ближайшей точки перехода*<sup>3</sup> текущего элемента таблицы маршрутизации.

<sup>3</sup> Как и в случае стандартного алгоритма маршрутизации, в качестве адреса ближайшей точки перехода должен быть указан адрес маршрутизатора, с которым отправитель может напрямую связаться по сети.

## 10.13. Унифицированный алгоритм маршрутизации

Скрупулезный читатель наверняка уже догадался, что алгоритм маршрутизации подсетей является более общим случаем стандартного алгоритма маршрутизации, особенно если учесть, что значение маски может быть произвольным. Применяя маскирование, можно выполнить маршрутизацию к отдельным узлам сети, маршрутизацию по стандартному маршруту, а также маршрутизацию к непосредственно подключенным сетям, точно также, как и маршрутизацию к подсетям. Кроме того, с помощью масок можно выполнить маршрутизацию и для традиционных классовых адресов. Подобный универсализм возможен благодаря объединению двух 32-разрядных произвольных значений, находящихся в полях *маски подсети* и *сетевого адреса*. Например, чтобы выполнить маршрутизацию к одиночному узлу сети, необходимо использовать маску, состоящую из всех единиц. При этом адрес сети будет равен IP-адресу узла. Для маршрутизации по стандартному маршруту используется маска, состоящая из всех нулей. При этом адрес сети также будет состоять из всех нулей, поскольку операция логического И с любым значением адреса получателя и нулевой маской даст в результате нулевое значение. Для выполнения маршрутизации к сети класса B, которая не имеет подсетей, используется маска, два первых октета которой равны единице, а два последних — нулю. Поскольку в таблице маршрутизации может содержаться произвольная информация, в универсальном алгоритме маршрутизации возможно меньше частных случаев, чем в стандартном алгоритме маршрутизации (листинг 10.1). По заданному IP-адресу получателя и существующей таблице маршрутизации, содержащей маски подсети, этот алгоритм определяет адрес ближайшей точки перехода, куда следует направить дейтаграмму. Не забывайте, что адрес ближайшей точки перехода должен находиться в той же сети, к которой имеет прямое подключение отправитель дейтаграммы.

### Листинг 10.1. Унифицированный алгоритм маршрутизации протокола IP

Маршрутизация\_Дейтаграммы (Дейтаграмма, Таблица\_Маршрутизации)

- 1: Извлечь из Дейтаграммы IP-адрес конечного получателя  $D$  и определить префикс сети  $N$ .
- 2: Если  $N$  совпадает с префиксом одной из сетей, к которой непосредственно подключена машина, выполнить прямую доставку дейтаграммы получателю  $D$  по соответствующей сети. (При этом нужно определить физический адрес машины  $D$ , инкапсулировать дейтаграмму в сетевой фрейм и отправить фрейм получателю).
- 3: Иначе, выполнить цикл по каждому элементу таблицы маршрутизации.
- 4: Положить  $N$  равным поразрядному логическому И между  $D$  и маской подсети, извлеченной из текущего элемента таблицы маршрутизации.
- 5: Если  $N$  равно значению поля адреса сети текущего элемента таблицы маршрутизации, переслать дейтаграмму по адресу ближайшей точки перехода, указанной в текущем элементе таблицы маршрутизации.
- 6: Повторить цикл для следующего элемента таблицы маршрутизации.
- 7: Если цикл маршрутизации закончен, а нужный элемент таблицы маршрутизации не найден, сгенерировать ошибку маршрутизации.

По сути в большинстве реализаций этого алгоритма не выполняется явная проверка IP-адреса на принадлежность к одной из непосредственно подключенных физических сетей. Вместо этого в таблицу маршрутизации добавляются дополнительные элементы (по одному для каждой из непосредственно подключенных сетей). Как и в других элементах, в них указывается маска подсети, позволяющая выделить префикс IP-адреса.

## 10.14. Работа с масками подсети

Познакомившись с унифицированным алгоритмом маршрутизации перейдем к рассмотрению двух вопросов: как назначаются маски подсети и как информация о них распространяется по сети. Второй вопрос в общих чертах был рассмотрен в главе 9, “Протокол IP: обработка ошибок и управляющие сообщения (ICMP)”. Там шла речь о том, что любой узел сети может определить маску подсети, послав соответствующий ICMP-запрос маршрутизатору, подключенному к этой сети. Если адрес маршрутизатора неизвестен, узел сети может отправить запрос в широковещательном режиме. В следующих главах мы завершим рассмотрение этой темы, описав протоколы, используемые маршрутизаторами для обмена маршрутной информацией, в которой присутствует также адрес сети и ее маска.

Ответить на первую часть вопроса труднее. Это объясняется тем, что администрация каждого сетевого центра вправе самостоятельно выбрать маски для своих сетей. При назначении масок сетевой администратор должен найти компромисс между размером сетей, их количеством и ожидаемым ростом, а также простотой обслуживания. Трудность заключается в том, что, хотя нестандартные маски позволяют достичь наибольшей гибкости, их использование может привести к неоднозначной интерпретации адресов маршрутизаторами. В худшем случае правильно сделанные назначения IP-адресов становятся некорректными при подключении к существующим сетям нескольких новых узлов. Поэтому здесь невозможно дать простые рекомендации. Обычно большинство сетевых центров выбирают проверенные временем решения. Как правило, для всех физических сетей автономной системы назначаются одинаковые сетевые маски, в которых биты, относящиеся к сетевой части адреса, располагаются подряд. Например, во многих автономных системах, которым выделен блок адресов класса B, для идентификации подсети используется один октет.

## 10.15. Широковещательная передача в подсетях

Реализовать режим широковещательной передачи данных при использовании адресации подсетей намного сложнее, чем в обычной сети TCP/IP. Напомним, что в традиционной системе адресации протокола IP, режим широковещательной передачи данных в конкретной сети задается установкой в единицу всех битов IP-адреса, относящихся к идентификатору узла. С точки зрения наблюдателя, находящегося за пределами автономной системы, в которой используется адресация подсетей, режим широковещания по адресу сети, тем не менее имеет смысл<sup>4</sup>. Другими словами, адрес

{ адрес сети, -1, -1 }

означает следующее: “доставить копию дейтаграммы всем машинам, у которых префикс сети IP-адреса совпадает с указанным адресом сети, даже если они

<sup>4</sup> Далее в этой главе будет рассмотрен метод бесклассовой адресации, использование которого делает бессмысленным применение режима широковещательной передачи для всех подсетей.

находятся в отдельных физических сетях". С точки зрения реализации выполнять широковещательную передачу по такому сетевому адресу имеет смысл только в том случае, если маршрутизаторы, соединяющие физические подсети, позволяют распространять подобные дейтаграммы по всем физическим сетям. Естественно, что при этом нужно тщательно проконтролировать весь маршрут прохождения дейтаграмм, дабы избежать их зацикливания. В частности, маршрутизатор не может без негативных последствий распространить пакет, полученный по одному из сетевых интерфейсов, по всем другим сетевым интерфейсам, которым назначен заданный сетевой префикс. Для предотвращения зацикливания дейтаграмм, в маршрутизаторах используется метод *пересылки по обратному маршруту* (*reverse path forwarding*). Суть метода состоит в том, что сначала маршрутизатор извлекает из полученной в широковещательном режиме дейтаграммы IP-адрес отправителя. Затем по этому адресу путем поиска в локальной таблице маршрутизации определяется интерфейс, по которому дейтаграммы должны были бы передаваться к отправителю (т.е. кратчайший путь к отправителю). И если окажется, что дейтаграмма пришла от отправителя не по кратчайшему пути, маршрутизатор ее аннулирует.

При наличии нескольких подсетей с одним префиксом IP-адреса можно организовать режим широковещательной передачи внутри специфической подсети (т.е. передать копию пакета всем машинам, подключенным к физической сети, которой назначен указанный адрес подсети). В стандарте адресации подсетей указанный режим широковещания реализуется путем установки в единицу всех битов IP-адреса, относящихся к идентификатору узла. Другими словами в этом случае широковещательный адрес будет выглядеть так:

{ адрес сети, адрес подсети, -1 }

После рассмотрения типов широковещательных адресов для подсетей и самого принципа широковещательной передачи в подсетях вам должно быть понятно, почему для всех сетей, которым назначен один префикс IP-адреса, рекомендуется выбирать такие маски подсети, чтобы биты, идентифицирующие сетевую часть в них, располагались подряд. Тогда широковещательные адреса для подсетей могут интерпретироваться однозначно. В случае более сложной разбивки адресов подсетей маршрутизаторы могут не разрешить широковещательную передачу дейтаграмм по всем физическим сетям, которым назначен заданный префикс IP-адреса.

## 10.16. Анонимные двухточечные сети

Напомним, что в оригинальной системе IP-адресации каждой физической сети назначается уникальный префикс IP-адреса. В частности, поскольку с точки зрения протокола IP, каждое двухточечное соединение между двумя машинами рассматривается как отдельная сеть, ей должен быть назначен уникальный префикс IP-адреса. При этом оба компьютера должны иметь уникальные суффиксы IP-адреса. В условиях постоянной нехватки адресного пространства применение отдельного префикса IP-адреса для каждого двухточечного соединения выглядит абсурдным. Эта проблема усугубляется для тех организаций, внутренняя структура сети которых построена на большом количестве двухточечных соединений. Например, в организациях, состоящих из нескольких территориально разнесенных филиалов, для формирования магистрального канала, связывающего маршрутизаторы филиалов, обычно используются высокоскоростные цифровые выделенные линии типа T1.

Поэтому, чтобы избежать назначения каждому двухточечному соединению отдельного префикса IP-адреса, для организации связи между двумя машинами

был предложен довольно простой метод, который назвали *анонимным подключением к сети* (*anonymous networking*). Этим методом часто пользуются когда два маршрутизатора соединены между собой цифровым выделенным каналом связи. Суть метода состоит в том, что выделенной линии не присваивается отдельный префикс сети, а двум подключенными к ней машинам не выделяются IP-адреса. При этом становится ненужным использование физических адресов сетевых плат. Изменения вносятся только в драйвер сетевой платы, поскольку при отсылке дейтаграммы адрес ближайшей точки перехода не нужен. Таким образом, при использовании анонимной сети в таблице маршрутизации протокола IP вместо адреса ближайшей точки перехода может находиться произвольное значение.

При организации двухточечных соединений с использованием описанной в этом разделе методики получается сеть, которую называют *ненумерованной* или *анонимной*. На рис. 10.7 показан пример, позволяющий понять принцип маршрутизации в ненумерованных сетях.

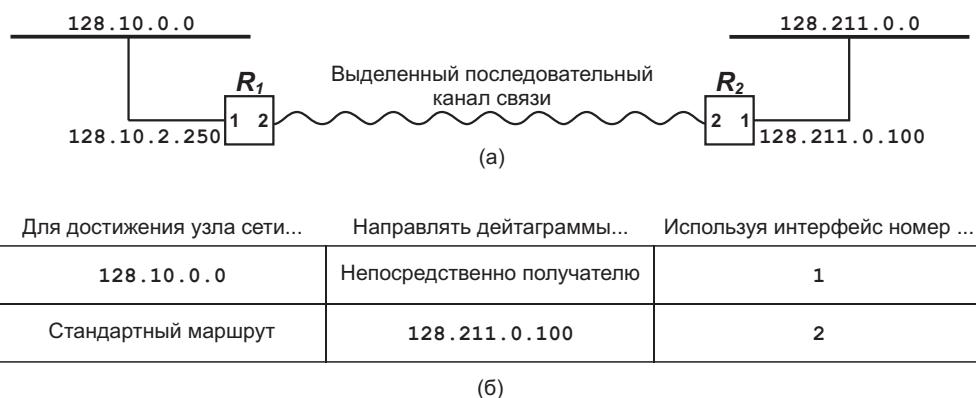


Рис. 10.7. Пример ненумерованного двухточечного соединения между двумя маршрутизаторами (а); таблица маршрутизации устройства  $R_1$  (б)

При анализе ненумерованных сетей нужно иметь в виду, что принцип работы последовательных каналов связи, используемых для организации двухточечных соединений, отличается от сетевых адаптеров, используемых для построения сети с общей средой передачи данных. Поскольку при двухточечном соединении существует только один возможный получатель (т.е. компьютер, находящийся на другом конце канала связи), при передаче фреймов сетевому оборудованию не требуется указывать его физический адрес. Следовательно, когда программы протокола IP передают дейтаграмму модулю сетевого интерфейса, в качестве адреса ближайшей точки перехода можно задать любое значение, поскольку оно не используется при передаче данных. Таким образом, в поле адреса ближайшей точки перехода таблицы маршрутизации можно поместить любое значение, чаще всего нулевое.

Следует отметить, что в таблице маршрутизации, показанной на рис. 10.7(б), в поле адреса ближайшей точки перехода (столбец *Направлять дейтаграммы...*) содержатся отнюдь не произвольные и тем более не нулевые значения. В этом примере показана одна из методик, часто применяемая в ненумерованных сетях. Суть ее заключается в том, что вместо произвольного значения в поле адреса ближайшей точки перехода таблицы маршрутизации помещается IP-адрес ближайшего маршрутизатора (т.е. IP-адрес сетевого интерфейса маршрутизатора, подключенного к другому концу канала связи). В нашем примере там содержится IP-адрес сетевой платы Ethernet маршрутизатора  $R_2$ .

Выше уже говорилось о том, что адрес ближайшей точки перехода не используется в модуле сетевого интерфейса. Поэтому может показаться странным, что в соответствующем поле таблицы маршрутизации находится какое-то значение. Еще более странным является то, что это значение соответствует адресу сетевого интерфейса, который напрямую не подключен к устройству  $R_1$ . На самом деле это значение не используется ни в программах протокола IP, ни в модуле сетевого интерфейса. Единственная причина, по которой поле адреса ближайшей точки перехода таблицы маршрутизации не оставляется пустым, — удобство обслуживания двухточечного соединения сетевым администратором. Это значение будет напоминать об адресе сетевого интерфейса маршрутизатора, подключенного к другому концу двухточечного соединения, а также поможет быстрее понять топологию сети. В нашем примере мы указали IP-адрес сетевой платы Ethernet устройства  $R_2$ , поскольку для платы интерфейса выделенного канала связи адрес не назначен.

## 10.17. Бесклассовая адресация и суперсети

Метод адресации подсетей, описанный в предыдущих разделах, был придуман в начале 1980-х годов и предназначался главным образом для экономного использования 32-разрядного адресного пространства протокола IP. Чуть позже для этой же цели была придумана методика использования ненумерованных сетей. Однако уже к 1993 году стало понятно, что применение только этих методик не решит проблему исчерпания адресного пространства в быстро растущей Internet. Поэтому были начаты работы по созданию совершенно новой версии протокола IP, поддерживающей гораздо большее адресное пространство.

Поскольку на разработку и принятие нового стандарта протокола IP требовалось время, а темпы роста Internet все увеличивались, был найден временный выход, позволивший решить проблему исчерпания старого адресного пространства. Этим выходом стала *бесклассовая адресация (classless addressing)*, *адресация суперсетей (supernet addressing)*, или просто *суперсеть (supernetting)*. Был применен подход, противоположный тому, что использовался при адресации подсетей, т.е. вместо использования одного префикса IP-адреса во всех физических сетях организации в суперсети организации разрешалось применять префиксы адресов, принадлежащие разным классам сетей.

Чтобы понять причины введения бесклассовой адресации необходимо принять во внимание три следующих важных фактора. Во-первых, применение классовой системы адресации приводило к неравномерному использованию адресного пространства внутри классов. Так, максимально возможное количество сетей класса  $B$  составляло порядка 17 000, тогда как сетей класса  $C$  могло быть более двух миллионов. Во-вторых, адресное пространство класса  $C$  исчерпывалось очень медленно. Из максимально возможного количества сетей класса  $C$  было распределено только несколько процентов. В-третьих, исследования показали, что при существующих темпах регистрации сетей класса  $B$ , доступное им адресное пространство довольно быстро исчерпается. Подобное положение вещей называли проблемой исчерпания адресного пространства (*Running Out of Address Space*, или *ROADS*), или проблемой *ROADS*.

Чтобы понять принцип работы суперсети, давайте рассмотрим организацию среднего размера, сети которой нужно подключить к Internet. Для подобных организаций предпочтительно использовать один префикс IP-адреса класса  $B$  по двум причинам. Во-первых, максимальное количество узлов в сети класса  $C$  не может превышать 254; во-вторых, адреса класса  $B$  очень удобно использовать в подсетях, поскольку локальная часть IP-адреса имеет достаточное количество бит. При использовании суперсети, организации, вместо одного блока адресов

класса *B*, выделяется подряд несколько блоков адресов класса *C*. При этом существенно экономиться пространство адресов класса *B*. Количество адресов класса *C* должно быть достаточным для нумерации всех сетей, которые организация планирует в будущем подключить к Internet. Предположим что организация отправила заявку на блок адресов класса *B*, который предполагается использовать для адресации подсетей на основе третьего октета IP-адреса. Однако вместо одного блока адресов класса *B*, организации выделили подряд 256 блоков адресов класса *C*. Таким образом, при использовании суперсети организация может назначить эти адреса всем своим физическим сетям.

Хотя принцип работы суперсети проще понять и описать на примере одной организации, разработчики предусмотрели возможность его использования в более широком масштабе. По их замыслу сеть Internet должна быть иерархической системой, средства доступа к которой предоставляются коммерческими поставщиками услуг, или *провайдерами Internet* (*Internet Service Providers*, или *ISP*). Для подключения своих сетей к Internet администрация предприятия или организации должна заключить договор с одним из поставщиков услуг Internet. При этом организация физического подключения сетей заказчика и назначение им IP-адресов возлагается на провайдера. Разработчики суперсети предложили выделять поставщикам услуг достаточно большую часть адресного пространства (т.е. несколько блоков адресов класса *C*). Тогда поставщики услуг смогут выделить из своего адресного пространства необходимое количество адресов каждому из клиентов.

## 10.18. Влияние суперсети на маршрутизацию

Как уже было сказано, для временного решения проблемы исчерпания адресного пространства и экономного использования IP-адресов класса *B* можно использовать несколько смежных блоков адресов класса *C*. Однако при этом возникает новая проблема — грандиозное увеличение количества передаваемой информации между маршрутизаторами, а также объем хранимых в этих устройствах данных. Например, при назначении организации 256 блоков адресов класса *C* вместо одного блока адресов класса *B* требуется поместить в таблицу маршрутизации 256 дополнительных записей.

Для решения этой проблемы придумали специальную методику, которую назвали *бесклассовой междоменной маршрутизацией*<sup>5</sup> (*Classless Inter-Domain Routing*, или *CIDR*). Идея использования CIDR состоит в том, что несколько непрерывных блоков адресов класса *C* записываются в виде одного элемента, состоящего из пары значений, как показано ниже.

( Начальный адрес сети, количество )

Под *начальным адресом сети* подразумевается минимальное значение адреса в блоке, а параметр *количество* определяет общее число используемых блоков адресов. Например, пара

( 192.5.48.0, 3 )

определяет три адреса сети класса *C*: 192.5.48.0, 192.5.49.0 и 192.5.50.0.

Если несколько поставщиков услуг, договорившись, сформируют ядро Internet и при этом каждому провайдеру будет выделен непрерывный блок IP-адресов, преимущества от использования суперсети становятся очевидными, поскольку при этом существенно сокращается размер таблиц маршрутизации. Рассмотрим содержимое

<sup>5</sup> Это название немного неточное, поскольку кроме маршрутизации в предложенной методике определены также и способы адресации.

таблицы маршрутизации, обслуживаемой поставщиком услуг  $P$ . В ней должны быть корректно прописаны маршруты ко всем клиентам этого провайдера, однако маршруты к клиентам других провайдеров прописывать не нужно — в таблицу маршрутизации следует просто добавить по одному элементу для каждого провайдера. Элемент должен определять маршрут ко всему блоку адресов, выделенному провайдеру.

## 10.19. Блоки адресов CIDR и битовые маски

На практике при использовании бесклассовой междоменной маршрутизации (CIDR) начальный адрес сети может не принадлежать к классу  $C$ , а количество блоков не обязательно должно задаваться целым числом. Однако размер каждого из блоков адресов должен быть кратен  $2^n$ . При этом для указания размера блока можно вместо целого числа использовать битовую маску. Предположим, организации выделен непрерывный блок IP-адресов в количестве 2048, начиная с адреса 128.211.168.0. В табл. 10.2 приведены значения первого и последнего адресов диапазона, представленные в точечной десятичной и в двоичной форме.

Для определения блока адресов в CIDR, показанного в табл. 10.2, необходимо указать 32-разрядное значение начального адреса блока и 32-разрядную маску. Принцип построения маски такой же, как и при определении маски подсети, за одним исключением: биты, соответствующие префиксу адреса, должны быть расположены подряд. В CIDR-маске рассматриваемого примера необходимо установить в единицу первые 21 бит. Это будет означать, что граница раздела между префиксом и суффиксом будет проходить по 21 биту, т.е. префикс занимает первые 21 бит двоичного числа, а суффикс — последние 11 бит, как показано ниже.

11111111 11111111 11111000 00000000

---

**Таблица 10.2. Пример блока CIDR из 2048 адресов**

---

Тип	Адрес	Двоичный эквивалент
Нижний	128.211.168.0	10000000 11010011 10101000 00000000
Верхний	128.211.175.255	10000000 11010011 10101111 11111111

---

## 10.20. Блоки адресов и форма их записи в CIDR

Поскольку для определения блоков адресов в CIDR нужно указать два значения — начальный адрес и маску, — была придумана специальная сокращенная форма записи, позволяющая компактно выразить эти два значения. Официально ее называли *формой записи CIDR (CIDR notation)*, однако чаще всего ее называют *записью через косую (slash notation)*. В сокращенной форме записи вначале указывается начальный адрес блока, выраженный в точечной десятичной форме, а затем, через косую черту, — длина маски в битах, выраженная целым десятичным числом. Таким образом, блок адресов, приведенный в табл. 10.2, в сокращенной форме записи CIDR будет выглядеть так:

128.211.168.0/21

Здесь запись /21 означает, что длина маски префикса равна 21 биту. В табл. 10.3 приведены все возможные CIDR-маски и эквивалентные им значения, выраженные в точечной десятичной форме. Следует отметить, что маски /8, /16 и /24 соответствуют стандартному разделению адресов на классы  $A$ ,  $B$  и  $C$ .

---

**Таблица 10.3. Перечень возможных CIDR-масок и эквивалентные им значения, выраженные в точечной десятичной форме**

---

<i>Маска</i>	<i>Эквивалент</i>	<i>Маска</i>	<i>Эквивалент</i>
/1	128.0.0.0	/17	255.255.128.0
/2	192.0.0.0	/18	255.255.192.0
/3	224.0.0.0	/19	255.255.224.0
/4	240.0.0.0	/20	255.255.240.0
/5	248.0.0.0	/21	255.255.248.0
/6	252.0.0.0	/22	255.255.252.0
/7	254.0.0.0	/23	255.255.254.0
/8	255.0.0.0	/24	255.255.255.0
/9	255.128.0.0	/25	255.255.255.128
/10	255.192.0.0	/26	255.255.255.192
/11	255.224.0.0	/27	255.255.255.224
/12	255.240.0.0	/28	255.255.255.240
/13	255.248.0.0	/29	255.255.255.248
/14	255.252.0.0	/30	255.255.255.252
/15	255.254.0.0	/31	255.255.255.254
/16	255.255.0.0	/32	255.255.255.255

---

## 10.21. Пример бесклассовой адресации

На примере табл. 10.3 показано одно из основных преимуществ бесклассовой системы адресации: речь идет о полной свободе выбора при распределении блоков адресов переменной длины. При использовании CIDR провайдер может выделить каждому из своих клиентов блок адресов требуемого размера. Если провайдеру выделен блок адресов, CIDR-маска которого имеет длину  $N$  битов, то провайдер может выделить своим клиентам блок адресов любой длины, однако его CIDR-маска должна быть больше  $N$  битов. Например, если провайдеру выделен блок адресов 128.211.0.0/16, он вполне может выделить одному из своих клиентов блок из 2048 адресов 128.211.168.0/21, как показано в табл. 10.2. Если же одному из мелких клиентов нужно всего несколько IP-адресов, чтобы подключить пару компьютеров к Internet, провайдер может выделить ему другой блок адресов 128.211.176.212/29, диапазон которых приведен в табл. 10.4. Очевидно, что использование битовой маски переменной длины позволяет более гибко выделять блоки IP-адресов, чем это было при использовании классовой системы адресации.

---

**Таблица 10.4. Пример блока CIDR 128.211.176.212/29**

---

<i>Тип</i>	<i>Адрес</i>	<i>Двоичный эквивалент</i>
Нижний	128.211.176.212	10000000 11010011 10110000 11010100
Верхний	128.211.176.215	10000000 11010011 10110000 11010111

---

Есть прекрасный способ пояснить принцип бесклассовой адресации: представьте себе, что провайдер выделяет каждому клиенту из своего блока адресов подсеть переменной длины. Таким образом, выделенный блок адресов может быть разбит произвольным образом на префикс сети и суффикс узла. При этом для каждого варианта разделения можно прописать отдельный маршрут следования дейтаграмм. Несмотря на то что группе подключенных к одной сети компьютеров могут быть назначены последовательные адреса, диапазон этих адресов не обязательно должен принадлежать одному из предопределенных классов. Предложенный метод разделения дает системному администратору определенную свободу при назначении адресов, поскольку он может точно определить количество битов, занимаемых префиксом IP-адреса. Из всего изложенного выше следует такой вывод.

*Использование бесклассовой адресации позволяет провайдерам Internet выделять своим клиентам непрерывный блок IP-адресов нужного размера. При этом сами IP-адреса рассматриваются как обычные целые числа, а количество адресов в блоке всегда кратно  $2^n$ .*

## 10.22. Структуры данных и алгоритмы для бесклассового поиска

Основным критерием выбора алгоритмов и структур данных, используемых для работы с таблицами маршрутизации, является скорость. При этом существует два разных подхода. В первом из них преимущество отдается скорости поиска адреса ближайшей точки перехода для заданного адреса конечного получателя. Во втором — скорости внесения изменений в элементы таблицы маршрутизации.

Введение бесклассовой адресации сильно повлияло на методы маршрутизации в объединенной сети, поскольку изменился основополагающий принцип, заложенный при разработке адресации протокола IP. В отличие от классовой системы адресации, при использовании бесклассовой междоменной маршрутизации (CIDR) IP-адреса перестают быть *автоматически опознаваемыми*. Это означает, что маршрутизатор не сможет по виду IP-адреса определить границу раздела между префиксом сети и суффиксом узла. Указанное отличие очень важно, поскольку используемые при классовой адресации структуры данных и алгоритмы поиска нельзя применять в том случае, если в таблице маршрутизации указаны бесклассовые адреса. После короткого обзора методов классового поиска будут рассмотрены структуры данных и алгоритмы, применяемые при бесклассовой системе адресации.

### 10.22.1. Хеширование и классовая адресация

Все алгоритмы поиска, используемые при маршрутизации, оптимизированы по скорости. Если в протоколе IP разрешена только классовая адресация, то для достижения необходимого уровня оптимизации достаточно использовать только одну методику: хеширование. При помещении классового адреса в таблицу маршрутизации маршрутизатор извлекает сетевую часть IP-адреса  $N$  и использует ее в качестве параметра хеш-функции, или ключа. Точно так же обрабатывается и адрес конечного получателя. Маршрутизатор извлекает из него сетевую часть  $N$ , вычисляет по ней значение хеш-функции  $h(N)$  и использует полученное число как индекс при поиске в хеш-таблице.

Поскольку классовые IP-адреса являются автоматически опознаваемыми, алгоритм их хеширования работает очень хорошо. Более того, его эффективность практически не снижается, даже если некоторые элементы таблицы

маршрутизации определяют маршруты к подсетям. Это достигается благодаря легкости выделения из IP-адреса сетевой части и использованию ее в качестве ключа. Если для нескольких IP-адресов получено одинаковое значение хеш-функции (т.е. эти адреса должны быть помещены в одну и ту же ячейку хеш-таблицы), элементы, соответствующие одной ячейке хеш-таблицы, сортируются в порядке, обратном их значимости. При этом элементы, определяющие маршруты к подсетям, предшествуют элементам, определяющим маршруты к сетям. В результате, если для некоторого адреса конечного получателя в таблице маршрутизации будет указан и маршрут к сети и маршрут к подсети, описываемый алгоритм позволяет найти корректный элемент, определяющий маршрутизацию к подсети.

Однако при использовании бесклассовых адресов метод хеширования работает хуже. Причина в том, что бесклассовые адреса не являются автоматически опознаваемыми. Поэтому маршрутизатор не сможет определить для произвольного адреса границу раздела между префиксом и суффиксом и вычислить ключ для хеш-функции. Таким образом, для бесклассовых адресов нужно использовать другой метод поиска.

### 10.22.2. Поиск по длине маски

Простейший алгоритм поиска, учитывающий бесклассовую природу IP-адресов, состоит в простом переборе всех возможных значений масок (т.е. выполняется перебор префиксов и суффиксов IP-адреса различной длины). Для заданного адреса конечного получателя  $D$ , сначала берется маска, длина которой составляет 32 бита, затем 31, и т.д. вплоть до 0. Для каждого из возможных размеров маски,  $M$ , маршрутизатор извлекает  $M$  битов из адреса  $D$ , считает их префиксом сети, по которому выполняется поиск в таблице. Работа алгоритма завершается, когда в таблице будет найдено соответствие с одним из элементов, определяющими маршрут следования дейтаграммы. Таким образом, в результате поиска будет найден префикс подсети, имеющий максимальную длину.

Недостаток описанного выше метода очевиден. Поскольку выполняется перебор всех возможных значений маски и многократный просмотр содержимого таблицы, скорость работы данного алгоритма во много раз ниже, чем стандартного алгоритма поиска, используемого при классовой адресации. В худшем случае, т.е. когда в таблице не найден подходящий элемент маршрутизации, просмотр будет выполняться 32 раза. И даже если элемент маршрутизации будет найден, при использовании алгоритма последовательного перебора просмотр таблицы маршрутизации будет многократно выполняться вхолостую. Например, для поиска маршрута к традиционной сети класса  $B$  (т.е. CIDR-маска равна /16), необходимо 16 раз просмотреть таблицу маршрутизации. Хуже всего то, что для поиска стандартного маршрута следования дейтаграммы необходимо выполнить просмотр таблицы маршрутизации аж 31 раз! И это при том, что элемент стандартной маршрутизации чаще всего используется во многих таблицах маршрутизации.

### 10.22.3. Структуры с использованием двоичных деревьев

Для повышения эффективности поиска при использовании бесклассовых адресов в реально работающих программах необходимо отказаться от алгоритма последовательного перебора. Поэтому чаще всего таблицы маршрутизации организованы в виде иерархической структуры данных, а процесс поиска выполняется по иерархии сверху вниз. Самой популярной иерархической структурой является разновидность *двоичного дерева* (*binary trie*), в которой значе-

ние последующих битов адреса определяет направление поиска в двоичном дереве, начиная с корня.

Двоичное дерево представляет собой древовидную иерархическую структуру, в которой пути к хранящимся в ней элементам определяются значениями хранящихся в ней данных более высокого уровня. Чтобы понять принцип работы двоичного дерева, представьте себе, что набор 32-битовых адресов записан в виде двоичных строк, в которых удалены избыточные суффиксы. Оставшаяся часть, т.е. набор префиксов, уникальным образом определяет каждый из элементов. В качестве примера в табл. 10.5 приведены семь адресов, записанных в двоичном виде, и соответствующие им уникальные префиксы. Как видно из табл. 10.5, для уникальной идентификации значений адресов в наборе требуется разное количество бит. Например, значение первого адреса в таблице можно идентифицировать тремя битами, поскольку в наборе нет других адресов, начинающихся с 001. Однако для идентификации двух последних адресов набора требуется уже пять бит, поскольку 4-битовый префикс 1011 имеют несколько элементов набора.

Таблица 10.5. Набор 32-битовых адресов и соответствующие им уникальные префиксы

32-битовый адрес	Уникальный префикс
00110101 00000000 00000000 00000000	001
01000110 00000000 00000000 00000000	0100
01010110 00000000 00000000 00000000	0101
01100001 00000000 00000000 00000000	011
10101010 11110000 00000000 00000000	1010
10110000 00000010 00000000 00000000	10110
10111011 00001010 00000000 00000000	10111

Определив значения уникальных префиксов, их можно использовать для идентификации ветвей двоичного дерева. На рис. 10.8 изображено двоичное дерево для семи значений префикса, приведенных в табл. 10.5.

Каждый внутренний узел двоичного дерева (на рис. 10.8 они показаны в виде кружочка) соответствует двум или более значениям префиксов, а каждый внешний узел (на рис. 10.8 они показаны в виде квадратика) — одному уникальному префиксу. Алгоритм поиска завершает работу при достижении внешнего узла, или в том случае, когда для указанного значения префикса не существует путей в двоичном дереве. Например, поиск в таблице адреса

10010010 11110000 00000000 00000001

завершится неудачей, поскольку не существует ветви со значением 0, исходящей из узла 10. Таким образом, можно сделать следующий вывод.

*Для повышения эффективности поиска маршрутов следования пакетов при использовании бесклассовых адресов в программах маршрутизации следует применять структуры данных и алгоритмы, отличные от тех, что используются для поиска классовых адресов. Чаще всего для этих целей применяются методы, основанные на двоичных деревьях.*

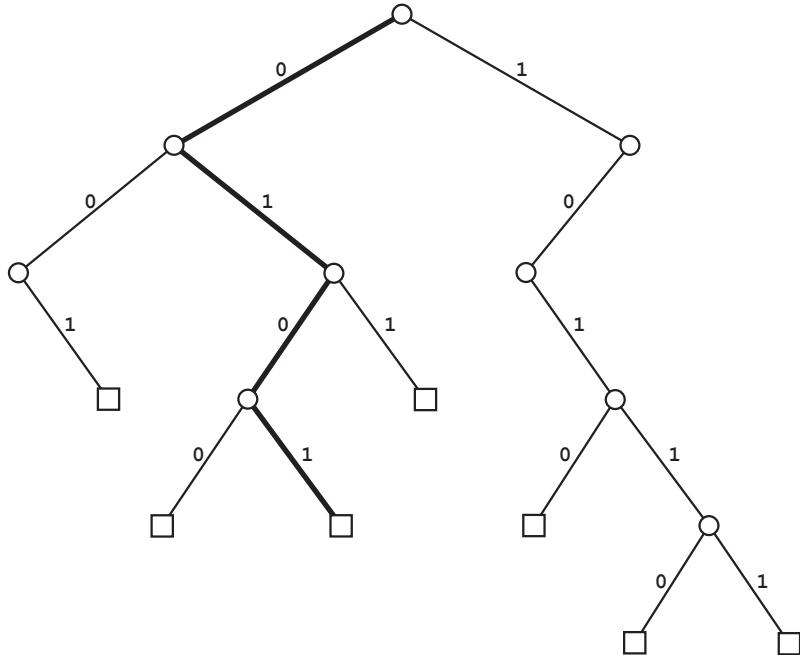


Рис. 10.8. Двоичное дерево для семи значений префикса, приведенных в табл. 10.5. Путь, соответствующий префиксу 0101, выделен жирной линией

### 10.23. Метод наилучшего совпадения и смешанный тип маршрутизации

Приведенное краткое описание двоичных деревьев может служить лишь общим обзором структур данных, которые используются на практике. Например, было сказано о том, что для идентификации каждого элемента таблицы маршрутизации в двоичном дереве необходимо задать уникальное значение префикса. При этом ничего не говорилось о том, что каждый префикс должен охватывать сетевую часть маршрута. Для гарантии того, что маршрутизатор будет пересыпать дейтаграммы только по правильному маршруту (т.е. префикс сети адреса конечного получателя должен совпадать с префиксом сети маршрута, указанного в таблице), каждый внешний узел двоичного дерева должен содержать 32-битовый адрес сети  $A$  и 32-битовую маску  $M$ , соответствующую адресу сети  $A$ . Если при работе алгоритма поиска будет достигнут внешний узел двоичного дерева, то вначале выполняется операция поразрядного логического И между значением IP-адреса конечного получателя и маской  $M$ . Полученный результат сравнивается с адресом сети  $A$  (так же, как и при использовании традиционных алгоритмов поиска в таблице маршрутизации). Если значения не совпадут, маршрутизация дейтаграммы не производится (как и в случае традиционных алгоритмов маршрутизации). Другими словами, двоичное дерево можно рассматривать как некий механизм, позволяющий быстро идентифицировать элементы, находящиеся в таблице маршрутизации, которые являются возможными кандидатами на выполнение маршрутизации, а не как механизм, позволяющий находить точное соответствие элементов.

Даже если предположить, что двоичное дерево является механизмом, позволяющим идентифицировать возможные соответствия элементов, в приведенном описании упущена одна важная деталь. Мы предположили, что каждый элемент в таблице маршрутизации имеет уникальный двоичный префикс. Однако на практике элементы в большинстве таблиц маршрутизации не имеют таких уникальных префиксов, поскольку, чаще всего, в таблицах, кроме общего маршрута следования дейтаграмм, обычно указывается несколько специфичных маршрутов к одному и тому же конечному получателю. В качестве примера достаточно рассмотреть таблицу маршрутизации, в которой для одной и той же подсети указан как специфичный для сети (в которую входит подсеть) маршрут, так и другой маршрут для подсети. Или таблицу, в которой указан как специфичный для сети маршрут, так и специальный маршрут к одному из узлов этой сети. Из приведенных примеров видно, что двоичный префикс, идентифицирующий элемент маршрутизации к сети, будет также идентифицировать элементы маршрутизации к подсети и к отдельному узлу этой сети. Пример таблицы маршрутизации, не содержащей уникальных префиксов, приведен в табл. 10.6. Подобная ситуация встречается в реальной жизни не так уж редко.

**Таблица 10.6. Пример таблицы маршрутизации, не содержащей уникальных префиксов**

Префикс	Адрес ближайшей точки перехода
128.10.0.0/16	10.0.0.2
128.10.2.0/24	10.0.0.4
128.10.3.0/24	10.1.0.5
128.10.4.0/24	10.0.0.6
128.10.4.3/32	10.0.0.3
128.10.5.0/24	10.0.0.6
128.10.5.1/32	10.0.0.3

При применении неуникальных префиксов необходимо видоизменить описанную выше древовидную структуру данных так, чтобы при выборе маршрута следования дейтаграммы можно было воспользоваться методом *наилучшего совпадения* (*longest-match*). Для этого нужно поместить во внутренние узлы дерева пары адрес/маска и изменить алгоритм поиска, чтобы проверка на совпадение адресов выполнялась в каждом узле дерева. При этом более поздние совпадения адресов на двоичном дереве (т.е. те, что соответствуют индивидуальным маршрутам) должны иметь преимущество перед всеми ранними совпадениями, поскольку они соответствуют более длинному префиксу.

### 10.23.1. Двоичное дерево PATRICIA и сжатие уровней

При описании двоичных деревьев мы не коснулись методов оптимизации поиска. Очень важно, чтобы при выполнении поиска были пропущены те уровни двоичного дерева, которые не определяют маршруты. Рассмотрим двоичное дерево для набора маршрутов, указанных в табл. 10.6. Поскольку первые 16 бит всех адресов в списке совпадают (т.е. они равны 10000000 00001010), то на первых 16 уровнях (начиная с корня) двоичного дерева, построенного для указанных адресов, будет содержаться по одному узлу.

В описанном выше случае гораздо эффективнее сразу проверить значение первых 16 бит адреса конечного получателя, а не извлекать их по одному за проход,

последовательно перемещаясь вниз по уровням двоичного дерева. Поэтому были разработаны два измененных варианта двоичного дерева, в которых используются элементарные средства оптимизации. Первое из них называется *двоичным деревом PATRICIA*. В каждом узле этого дерева указывается значение для проверки, а также количество битов, которые следует пропустить. Второе называется *двоичным деревом со сжатыми уровнями*. Дополнительная оптимизация в нем осуществляется за счет того, что из сравнения исключается один или несколько уровней, которые могут быть пропущены при обходе дерева по любому пути.

Вполне естественно, что оптимизация структур данных является примером принятия компромиссного решения. Несмотря на то, что оптимизация позволяет ускорить поиск маршрута в таблице, тем не менее ее использование приводит к дополнительным нагрузкам на центральный процессор при создании или изменении содержимого таблиц маршрутизации. Однако в большинстве случаев подобная оптимизация оправдана, поскольку изменение содержимого таблиц маршрутизации выполняется гораздо реже, чем поиск в них.

## 10.24. Блоки CIDR, выделенные частным сетям

В главе 4, “Классовая адресация”, уже упоминалось о том, что группой IETF был выделен специальный набор префиксов IP-адреса, предназначенных для использования в частных сетях. При этом гарантируется, что указанные зарезервированные префиксы никогда не будут назначены физическим сетям, которые подключены к глобальной сети Internet. Весь набор зарезервированных префиксов называют *частными (private)* или *немаршрутизуемыми (nonroutable)* *адресами*. Термин *немаршрутизуемые* означает, что маршрутизаторы в глобальной сети Internet “знают”, что эти адреса зарезервированы для локального использования. Поэтому, если по какой-либо причине в глобальную сеть Internet попадет дейтаграмма, посланная по одному из частных адресов, маршрутизатор сможет обнаружить проблему.

В группу зарезервированных префиксов протокола IPv4 входят как классовые адреса, так и блоки CIDR, охватывающие разные классы адресов. В табл. 10.7 приведены зарезервированные адреса, записанные в виде блоков CIDR, а также значения начального и конечного адресов блока, выраженные в точечной десятичной форме. Указанный в табл. 10.7 последним блок адресов 169.254/16 не применяется в частных сетях, поскольку он используется в системах с *автоматическим* назначением IP-адресов.

**Таблица 10.7. Список префиксов, зарезервированных для использования в частных локальных сетях, не подключенных к Internet**

Префикс	Начальный адрес блока	Конечный адрес блока
10/8	10.0.0.0	10.255.255.255
172.16/12	172.16.0.0	172.31.255.255
192.168/16	192.168.0.0	192.168.255.255
169.254/16	169.254.0.0	169.254.255.255

## 10.25. Резюме

В первоначальной системе IP-адресации каждой физической сети назначается уникальный префикс адреса. Поэтому в данной главе были рассмотрены пять методов, специально придуманных для экономного использования выделенного

адресного пространства. Первый метод позволяет расширить адресное пространство одной (как правило, глобальной) сети и подключить к ней через специальное устройство, называемое “прозрачным” маршрутизатором, несколько компьютеров локальной сети. Второй метод, предусматривающий использование агента ARP, позволяет маршрутизатору отвечать на ARP-запросы, адресованные компьютерам, подключенными к другой физической сети. Агенты ARP применяются только в тех сетях, где для преобразования адресов используется протокол ARP. Кроме того, имеется еще одно важное требование: программы протокола ARP не должны реагировать на ситуацию, когда нескольким IP-адресам соответствует один физический адрес.

Третий метод, являющийся одним из стандартов протокола TCP/IP, называется адресацией подсетей. Он позволяет назначать общий префикс IP-адреса нескольким физическим сетям. При этом на всех узлах и маршрутизаторах, подключенных к сетям, где используется адресация подсетей, должны быть запущены программы, поддерживающие видоизмененный алгоритм маршрутизации. Он заключается в том, что программы должны корректно обрабатывать элементы таблиц маршрутизации, содержащие маску подсети. Алгоритм маршрутизации при наличии подсетей можно рассматривать как обобщение первоначального алгоритма маршрутизации, поскольку он позволяет обрабатывать особые случаи, такие как наличие стандартных маршрутов следования дейтаграмм и маршрутов к конкретному узлу сети. Четвертый метод, связанный с использованием анонимных или ненумерованных сетей, позволяет не назначать префикс IP-адреса двухточечным соединениям.

Пятый метод, называемый бесклассовой адресацией и бесклассовой междоменной маршрутизацией (CIDR), является большим достижением в развитии IP-технологии. В отличие от традиционной классовой системы адресации, использование этого метода позволяет разделить IP-адрес на префикс и суффикс произвольным образом. Технология CIDR позволяет разделить доступное адресное пространство на блоки, размер которых кратен  $2^n$ . Одной из сильных сторон CIDR является возможность объединения нескольких блоков адресов класса C в один блок адресов суперсети. В отличие от оригинальной классовой системы адресации, бесклассовые адреса не являются автоматически опознаваемыми. Поэтому применение CIDR требует существенного изменения алгоритмов поиска маршрутов и используемых при этом структур данных для их хранения. Во многих реализациях CIDR для ускорения поиска используются иерархические структуры данных, называемые двоичными деревьями.

## Материал для дальнейшего изучения

Стандарт адресации подсетей был описан Могулом (Mogul) в [RFC 950] и дополнен Браденом (Braden) в [RFC 1122]. Первоначальные предложения по адресации подсетей были описаны Кларком (Clark) в [RFC 932], Карелсом (Karels) в [RFC 936], Гадсом (Gads) [RFC 940] и Могулом в [RFC 917]. Анализ широковещательной передачи данных при наличии подсетей сделан Могулом в [RFC 922]. Использование агентов ARP в подсетях описано Постелом (Postel) в [RFC 925]. В статье Аталаха (Atallah) и Камера [4] приведено доказательство существования оптимального алгоритма распределения адресов в подсетях переменной длины. Использование агентов ARP для реализации “прозрачных” маршрутизаторов, работающих в подсетях, описано Карл-Митчеллом (Carl-Mitchell) и Квотерманом (Quaterman) в [RFC 1027]. Распределение бесклассовых IP-адресов описано Рехтером (Rekhter) и Ли (Li) в [RFC 1518]. Бесклассовая междоменная маршрутизация (CIDR) и суперсети описаны Фаллером (Fuller), Ли (Li), Ю (Yu) и Варадханом (Varadhan) в [RFC 1519]. Префиксы адресов, зарезервированные

для использования в частных сетях, приведены Рехтером и др. в [RFC 1918]. Описание двоичных деревьев, в частности структуры данных PATRICIA, можно найти в третьем томе Кнута [79].

## Упражнения

- 10.1. Предположим, что на маршрутизаторе запущен агент ARP, который использует таблицу адресов узлов сети для принятия решения об ответе на поступивший ARP-запрос. Очевидно, что содержимое этой таблицы должно изменяться каждый раз, когда к одной из физических сетей добавляется новый узел. Продумайте механизм назначения IP-адресов, при использовании которого не нужно изменять содержимое таблицы адресов узлов сети в случае добавления к ней нового узла. (*Подсказка.* Воспользуйтесь подсетями.)
- 10.2. Хотя в стандарте не запрещается использовать в качестве номера подсети число, состоящее из всех нулей, тем не менее в некоторых реализациях протокола IP это работает некорректно. Попытайтесь присвоить одной из подсетей вашего центра нулевой номер и проверьте, будет ли при этом корректно распространяться маршрутная информация.
- 10.3. Можно ли использовать “прозрачный” маршрутизатор в локальных сетях, наподобие Ethernet? Поясните свой ответ.
- 10.4. Покажите, что агенты ARP могут использоваться в трех физических сетях, соединенных между собой двумя маршрутизаторами.
- 10.5. Предположим, что локальная часть блока адресов класса *B* разбита на две части фиксированного размера. При этом размер идентификатора физической сети достаточен для назначения адресов как минимум 76 физическим сетям. Какое максимальное количество узлов может быть при таком разбиении в каждой из физической сети?
- 10.6. Имеет ли смысл разбивать на подсети блок адресов класса *C*? Поясните свой ответ.
- 10.7. В некотором сетевом центре решили разбить блок адресов класса *B* так, чтобы каждую физическую сеть можно было идентифицировать по значению третьего октета IP-адреса. Каково же было удивление администрации центра, когда они узнали, что при таком разбиении нельзя адресовать 256 и даже 255 физических сетей. Поясните, почему.
- 10.8. Какой бы вы использовали метод адресации подсетей в организации, с учетом того, что ей выделен один блок адресов класса *B*.
- 10.9. Имеет ли смысл в рамках одного маршрутизатора использовать и агента ARP, и адресацию подсетей? Если да, то объясните, как. Если нет, то поясните, почему.
- 10.10. Докажите, что любая сеть, в которой используются агенты ARP, не защищена от подлога IP-адресов (т.е. когда компьютер сети может выдавать себя за другой компьютер).
- 10.11. Можете ли вы придумать алгоритм нестандартной реализации протокола ARP, в котором запрещено использование агентов ARP?
- 10.12. Один из производителей сетевого программного обеспечения решил добавить возможность адресации подсетей в свой модуль протокола

IP. При этом предполагалось, что для IP-адресов всех физических сетей будет использоваться одинаковое значение маски подсети. Производитель внес изменение в стандартный алгоритм IP-маршрутизации и сделал так, чтобы проверка на наличие подсети выполнялась в виде отдельной ветки программы (т.е. как частный случай алгоритма маршрутизации). Докажите, что в некоторых случаях такой алгоритм будет работать некорректно. (*Подсказка.* Рассмотрите работу многоадресного узла.)

- 10.13.** При каких условиях алгоритм, описанный в предыдущем упражнении, будет работать корректно?
- 10.14.** Обратитесь за дополнительной информацией, касающейся широковещательного режима передачи данных в подсетях, к соответствующему стандарту. Можете ли вы назвать метод назначения адресов подсетям, при котором для всех возможных подсетей будет существовать один широковещательный адрес?
- 10.15.** В стандарте допускается использование произвольного значения маски подсети в тех физических сетях, где применена адресация подсетей. Нужно ли в стандарте вводить ограничение на порядок расположения битов в маске (т.е. непрерывный или нет), идентифицирующих сетевую часть IP-адреса? Поясните свой ответ.
- 10.16.** Приведите пример адреса узла при использования адресации подсетей переменной длины, когда возникает проблема неоднозначности адресов.
- 10.17.** Подробно рассмотрите принцип организации стандартного маршрута следования дейтаграмм в условиях подсетей. Что произойдет при получении дейтаграммы, посланной по несуществующему адресу подсети?
- 10.18.** Сравните структуру систем, в которых для объединения нескольких физических сегментов сети Ethernet используются маршрутизаторы и адресация подсетей, а также мосты, описанные в главе 2, “Обзор основных сетевых технологий”. Назовите условия, при которых предпочтительнее та, или иная система.
- 10.19.** Предположим, что некоторому сетевому центру выделен блок адресов класса B. Администрация центра решила для идентификации нескольких физических сетей выделить 6 бит локальной части IP-адреса, а для идентификации остальных сетей — 8 бит. При каком распределении IP-адресов узлов компьютеры в сети будут идентифицироваться неоднозначно?
- 10.20.** В унифицированном алгоритме маршрутизации протокола IP, приведенном в листинге 10.1, применен метод простого перебора элементов таблицы маршрутизации. Это позволяет расположить в таблице маршруты к конкретным узлам сети перед маршрутами к сетям или подсетям. Создайте структуру данных, позволяющую достичь такой же универсальности при выполнении маршрутизации, и которую можно было бы использовать совместно с методом хеширования для повышения эффективности поиска. (Это упражнение было предложено Дэйвом Миллсом.)
- 10.21.** Хотя разработчики потратили немало усилий на повышение эффективности алгоритмов маршрутизации, программы бесклассовой маршрутизации работают медленнее своих аналогов для классовой маршрутизации. Основная причина — неэффективность алгоритмов

поиска бесклассовых адресов по сравнению с применяемыми методами хеширования для классовых адресов. Исследуйте возможность создания структур данных и алгоритмов поиска бесклассовых адресов, эффективность которых была бы выше, чем при использовании двоичных деревьев.

- 10.22.** В алгоритмах на основе двоичных деревьев в зависимости от значения соответствующего бита IP-адреса выбирается одно из двух направлений дальнейшего движения относительно текущего узла. Рассмотрите возможность использования деревьев, в которых выбиралось бы одно из четырех направлений дальнейшего движения относительно текущего узла по значению двух битов IP-адреса. При каких условиях использование подобных деревьев позволяет повысить эффективность поиска, а при каких снижает?
- 10.23.** Представьте себе, что у всех провайдеров Internet используется бесклассовая система адресации. При этом они выделяют своим заказчикам IP-адреса из собственного блока адресов. Что произойдет, если один из заказчиков захочет поменять провайдера?

# 11

## Уровни протоколов

### 11.1. Введение

В предыдущих главах был рассмотрен основной принцип построения объединенной сети, описан процесс передачи по ней дейтаграмм узлами и маршрутизаторами, и показан механизм преобразования IP-адресов в физические адреса. В этой главе внимание будет сосредоточено на структуре протокольного программного обеспечения, работающего на узлах сети и маршрутизаторах, которое предназначено для организации сетевого обмена. Кроме того, будут представлены основные принципы разбиения протоколов на уровни и показано, как благодаря такому разбиению облегчается создание программ, работающих с протоколом IP, а также рассмотрено прохождение дейтаграммы через уровни протоколов при передаче ее через объединенную сеть на основе протокола TCP/IP.

### 11.2. Необходимость нескольких протоколов

Выше мы уже говорили, что протоколы позволяют понять и описать процесс сетевого взаимодействия без знания подробностей реализации сетевого оборудования. Для процесса сетевого взаимодействия протоколы играют ту же роль, что и языки программирования для компьютерных вычислений. Это станет очевидно при более подробном рассмотрении аналогии. Подобно языку ассемблера, некоторые протоколы регламентируют процесс передачи данных по физической сети. Например, в стандарте Ethernet оговариваются формат фрейма, механизмы доступа к сети и обработки ошибок. Подобно языкам программирования высокого уровня, протокол IP определяет абстракции высокого уровня (т.е. IP-адресацию, формат дейтаграмм, концепцию негарантированной доставки и обмен данными без установки соединения с получателем).

Коммуникационные системы не используют один протокол для решения всех задач, связанных с передачей данных, — они применяют несколько совместно используемых протоколов, иногда называемых *семейством протоколов*, или *стеком протоколов*. Чтобы понять это, сначала необходимо разобраться с проблемами, возникающими при передаче данных по сети.

- *Аппаратные сбои.* Сбои в работе узла сети или маршрутизатора могут быть следствием неполадок в работе оборудования или зависания операционной системы. Кроме того, может выйти из строя канал передачи данных, или пользователь случайно отключит компьютер от сети. Подобные неполадки должны обнаруживаться программами поддержки сетевого протокола, и, если это возможно, устраняться.

- *Перегрузка сети.* Поскольку любая сеть имеет ограниченную пропускную способность, то рано или поздно этот предел может быть достигнут, даже если аппаратное и программное обеспечение функционирует в штатном режиме. Поэтому в программах поддержки протоколов должны быть предусмотрены меры, предотвращающие дальнейшее увеличение трафика в сети.
- *Задержка при передаче и потеря пакетов.* Иногда при передаче пакетов возникают чрезвычайно большие задержки, нередки также случаи потери пакетов. Поэтому программы протокола должны определять эти ошибки и приспособливаться к большим задержкам при передаче пакетов.
- *Повреждение данных.* Причиной повреждения передаваемых данных могут быть электромагнитные помехи, а также неполадки оборудования. Программы протоколов должны определять и устранять эти ошибки.
- *Дублирование данных и нарушение порядка доставки пакетов.* Если в объединенной сети существует несколько маршрутов к конечному получателю, то при передаче пакетов данных может нарушаться порядок их доставки, а также появляться дубликаты пакетов.

Как видите, проблем при передаче данных по сети возникает довольно много. Трудно представить, как можно написать одну программу протокола, чтобы в ней решались все эти задачи. По аналогии с языками программирования, решение должно быть комплексным. Трансляция программы разбивается на четыре концептуальные подзадачи, каждая из которых обрабатывается отдельным модулем — компилятором, ассемблером, компоновщиком или загрузчиком. Подобное разделение позволяет разработчику сосредоточиться на решении одной подзадачи, а конструктору — построить и протестировать каждую часть независимо. Ниже будет показано, что создание программ поддержки протоколов очень напоминает описанный выше процесс.

В заключение проследим сходство с языками программирования — это поможет понять структуру протокольного программного обеспечения. Во-первых, должен быть согласован точный формат передаваемых между модулями транслятора данных. Например, данные, передаваемые от компилятора ассемблеру, должны состоять из команд, написанных на языке ассемблера. В процессе трансляции может использоваться несколько форм представления данных. Этот принцип распространяется и на протокольное программное обеспечение — сначала нужно определить формат данных, передаваемых между модулями разных протоколов. Во-вторых, четыре модуля транслятора выполняются в линейной последовательности, в которой результат работы одного модуля (например, компилятора) является входными данными для следующего модуля (в данном случае — ассемблера) и т.д. В протоколах также используется линейная последовательность выполнения.

### 11.3. Концепция уровней протоколов

Семейство протоколов можно представить в виде расположенных друг над другом *уровней*, или *модулей*, каждый из которых отвечает за решение одной части общей задачи (рис. 11.1).

При передаче сообщения от одной машины к другой пакеты последовательно проходят через все уровни протокольного программного обеспечения отправителя, следуют по сети, после чего выполняется их последовательная обработка всеми уровнями протокольных программ получателя.

На самом деле структура семейства протоколов намного сложнее, чем показано на рис. 11.1. На каждом уровне проверяется корректность сообщения и выполняется действие в соответствии с типом сообщения или адресом получателя.

Например, программа протокола первого уровня, запущенная на машине получателя, должна решить, что нужно сделать с полученной дейтаграммой — принять ее или переслать другой машине. Программа следующего уровня решает, какое из приложений должно обработать принятое сообщение.

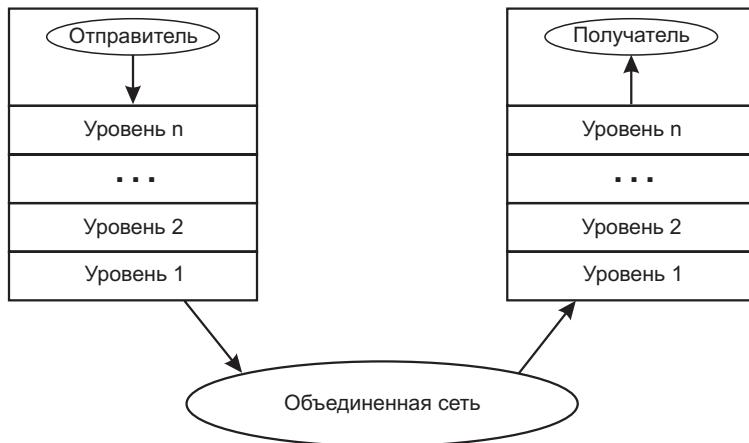


Рис. 11.1. Многоуровневая структура сетевого программного обеспечения

Чтобы вы ощущали разницу между абстрактной моделью семейства протоколов и программной реализацией, рассмотрим схемы, показанные на рис. 11.2. На обобщенной структурной схеме (рис. 11.2(а)) уровень протокола IP находится посередине между верхним уровнем, где расположены прикладные протоколы, и нижним уровнем сетевого интерфейса. На программной модели (рис. 11.2(б)) показано, что модуль протокола IP связан с несколькими протоколами высокого уровня и несколькими сетевыми интерфейсами.

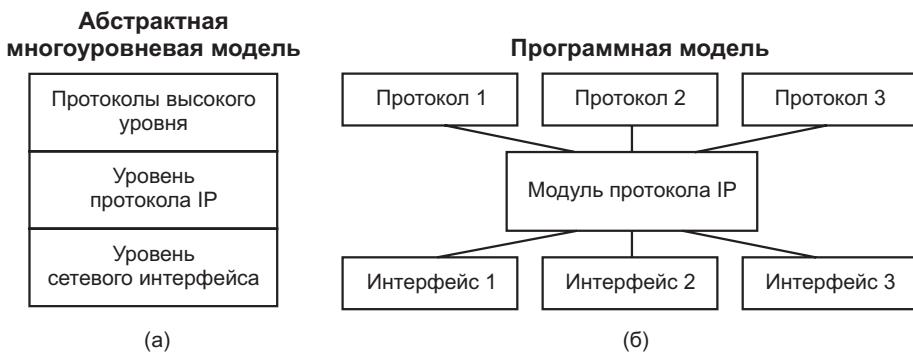


Рис. 11.2. В отличие от абстрактной (а), в программной (б) модели семейства протоколов модуль IP связан с несколькими сетевыми интерфейсами и несколькими протоколами высокого уровня

Хотя на обобщенной структурной схеме не показаны подробно все уровни протоколов, тем не менее она дает возможность понять основную идею. Например, на рис. 11.3 изображены уровни протоколов, используемые для передачи сообщения через три сети. На диаграмме показаны только уровни сетевого интерфейса и протокола IP, потому что только эти уровни необходимы для приема, маршрутизации и отправки дейтаграмм. Выше уже шла речь о том, что для

подключения компьютера к двум физическим сетям он должен иметь два интерфейсных модуля. Это нужно всегда иметь в виду при анализе схемы абстрактной многоуровневой модели, на которой изображен только один интерфейсный модуль для каждого компьютера (рис. 11.3)

Как показано на рис. 11.3, отправитель передает сообщение, которое помещается в дейтаграмму на уровне протокола IP и отправляется по сети 1. На промежуточных маршрутизаторах дейтаграмма обрабатывается протоколами уровня IP и передается в следующую сеть. И только когда она достигнет пункта назначения, протокол уровня IP извлечет сообщение, которое затем будет обработано программами протоколов верхнего уровня.

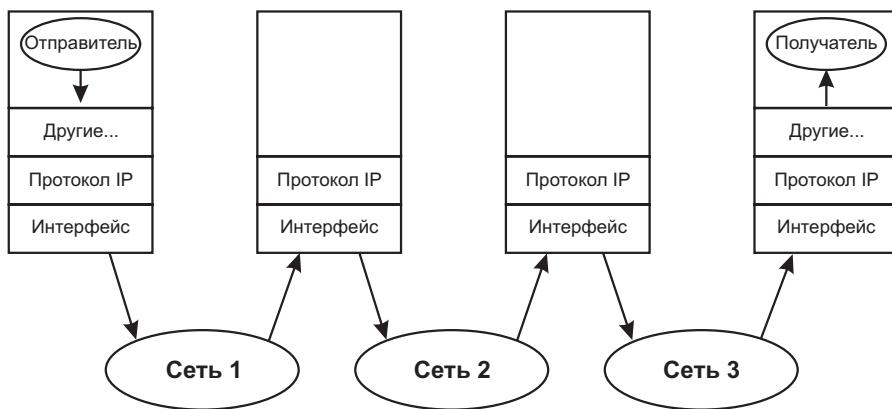


Рис. 11.3. Процесс передачи сообщения через объединенную сеть от отправителя через два промежуточных маршрутизатора до конечного получателя. Промежуточные маршрутизаторы обрабатывают дейтаграмму только на уровне протокола IP

## 11.4. Разделение функциональных обязанностей между уровнями

После того как решение одной большой коммуникационной проблемы разбито на части и написаны соответствующие программные модули, каждый из которых решает одну из небольших задач, возникает вопрос: какие функции должны выполняться в каждом из модулей? На этот вопрос нелегко ответить по ряду причин. Во-первых, на основании поставленных целей и с учетом того, что решение общей задачи сводится к решению отдельных коммуникационных проблем, можно выбрать такую структуру протокольного программного обеспечения, которая была бы оптимизирована для решения поставленной задачи. Во-вторых, даже если принять во внимание службы общего назначения сетевого уровня, такие как надежная транспортная система, существует несколько принципиально разных подходов к решению проблемы. В-третьих, структура сети (или объединенной сети) и семейство протоколов взаимосвязаны; одно не может быть создано без другого.

### 11.4.1. Семиуровневая модель ISO

При разбиении протоколов на уровни учитывалась *эталонная модель взаимодействия открытых систем* (*Reference Model of Open System Interconnection*), или *модель ISO*<sup>1</sup>, созданная Международной организацией по стандартизации

<sup>1</sup> Модель ISO также называют моделью OSI (*Open System Interconnection*)

(International Organization for Standardization, или ISO). На рис. 11.4 показаны 7 уровней модели ISO.

Следует учитывать, что модель ISO была разработана для описания структуры протоколов в рамках одной сети. В ней не предусмотрен уровень, выполняющий маршрутизацию пакетов в объединенной сети, как в семействе протоколов TCP/IP.

Уровень	Выполняемые функции
7	Приложений
6	Представлений
5	Сеансовый
4	Транспортный
3	Сетевой
2	Канальный (плата сетевого интерфейса)
1	Физический (среда передачи данных)

Рис. 11.4. Семиуровневая модель ISO, описывающая принцип построения протокольного программного обеспечения

## 11.5. Протокол X.25 и модель ISO

Хотя модель ISO разрабатывалась в виде набора абстрактных определений и не содержала конкретных правил по ее реализации, она была положена в основу реализации некоторых семейств протоколов. Среди протоколов, построенных на ее основе, широко используемым является X.25. Протокол X.25 основан на рекомендациях *Международного телекоммуникационного союза*<sup>2</sup> (International Telecommunications Union, или ITU) — организации, разрабатывающей международные стандарты для телефонии. Протокол X.25 адаптирован для применения в *сетях общего пользования* (Public Data Networks, или PDN), и был особенно популярен в Европе. Его рассмотрение поможет прояснить назначение уровней модели ISO.

Функционирование сети на основе протокола X.25 во многом напоминает работу обычной телефонной сети. Основу сети X.25 составляют сложные коммутаторы пакетов, используемые для маршрутизации пакетов. Компьютеры пользователей не подключаются к сети X.25 непосредственно. Каждый компьютер подключается к коммутатору пакетов с помощью последовательного канала связи.

<sup>2</sup> Ранее эта организация называлась *Международным консультативным комитетом по телеграфии и телефонии* (International Consultative Committee for Telegraphy and Telephony, или CCITT).

В каком-то смысле соединение компьютера с коммутатором пакетов является миниатюрной сетью, построенной на основе одного последовательного канала связи. Для передачи пакетов по сети компьютер пользователя должен выполнить ряд сложных процедур. Ниже мы вкратце рассмотрим особенности каждого из уровней модели ISO.

- **Физический уровень.** В протоколе X.25 определены стандарты физического подключения компьютера к коммутатору пакетов, а также процедура передачи пакета от одной машины до другой. В эталонной модели ISO уровень 1 определяет параметры физического подключения, в том числе электрические характеристики передаваемых сигналов (такие как величина напряжения и сила тока). Детали работы с сетями общего пользования описаны в протоколе X.21, связанном с X.25.
- **Канальный уровень.** Уровень 2 протокола X.25 определяет способ передачи данных между компьютером пользователя и коммутатором пакетов. Единица данных, передаваемых между этими устройствами, называется *фреймом* (следует отметить, что в стандарте X.25 понятие *фрейм* несколько отличается от данного нами в предыдущих главах определения). Поскольку для передачи данных на физическом уровне (т.е. по каналу связи) используется поток битов, то на канальном уровне необходимо определить формат фрейма и указать процедуру определения его границ. Так как ошибки, возникающие при передаче данных, приводят к искажению информации, то на уровне 2 должна быть предусмотрена возможность детектирования ошибок (т.е. выполнен подсчет контрольной суммы данных фрейма). И, наконец, поскольку процедура побитовой передачи данных является по своей природе ненадежной, на втором уровне требуется определить порядок обмена подтверждениями, который позволит двум машинам убедиться в успешной доставке фрейма.
- Один из широко используемых протоколов семейства X.25, относящийся ко второму уровню, называется *высокоуровневым протоколом управления каналом передачи данных* (High Level Data Link Communication, или HDLC). Существует несколько версий протокола HDLC, последняя из которых называется *HDLC/LAPB*. Важно отметить, что удачное завершение процедуры обмена данными на уровне 2 означает передачу фрейма коммутатору пакетов для дальнейшей доставки. Однако сам факт этого еще не гарантирует, что коммутатор примет фрейм или сможет выполнить его маршрутизацию.
- **Сетевой уровень.** Функцией третьего уровня эталонной модели ISO, который иногда называют уровнем *коммуникационной подсети* (*communication subnet*), является определение взаимодействия между узлом и сетью. На этом уровне оговаривается единица передачи данных по сети, вводятся концепции адреса получателя и маршрутизации. Напомним, что в семействе протоколов X.25 процесс взаимодействия между узлом и коммутатором пакетов принципиально не связан с потоком передаваемых данных. Это позволяет программному обеспечению третьего уровня формировать пакеты, размер которых превосходит размер фреймов, формируемых на уровне 2. На уровне 3 выполняется сборка пакетов в соответствии со стандартом используемого сетевого протокола, после чего с помощью программ уровня 2 они передаются (возможно, по частям) коммутатору пакетов. Уровень 3 также отвечает за предотвращение перегрузки сети.
- **Транспортный уровень.** Уровень 4 обеспечивает надежную транспортировку пакетов между двумя конечными точками сети благодаря установке соединения между узлами отправителя и получателя. Несмотря на то

что протоколы нижнего уровня проверяют правильность выполнения каждой операции при передаче данных, назначение этого уровня состоит в дополнительной проверке правильности передаваемых данных и подтверждения того, что все промежуточные звенья коммуникационной системы работают в штатном режиме.

- **Сеансовый уровень.** Верхние уровни модели ISO определяют такой способ организации протокольного программного обеспечения, благодаря которому прикладные программы обеспечивались бы необходимыми функциональными возможностями. Поскольку раньше работа с сетью осуществлялась в основном с помощью терминалов, подключенных к узловым компьютерам, комитет ISO не мог проигнорировать проблему доступа к узлам сети с удаленных терминалов. К решению проблемы подошли настолько серьезно, что поддержка сеансов связи с терминалами была поручена программам пятого уровня. Основная услуга, предлагаемая ранее в сетях общего пользования, фактически состояла в поддержке соединений между терминалами и узловыми компьютерами. Связь с сетью через коммутируемую линию обеспечивал специальный компьютер, называемый *сборщик/разборщик пакетов* (Packet Assembler and Disassembler, или PAD). Абонентами сети чаще всего являлись мобильные пользователи, которые возили с собой компьютер с модемом. Для работы в сети они должны были с помощью модема подключиться к PAD, установить сетевое подключение с нужным им компьютером, после чего войти в систему (*log in*). Большинство абонентов для связи на большие расстояния предпочитали использовать сеть, а не прямое модемное соединение, исходя из соображений стоимости.
- **Уровень представлений.** Уровень 6 отвечает за реализацию функций, необходимых программам при работе в сети. Типичным примером является сжатие текста или преобразование графического изображения в поток битов для передачи по сети. Следует отметить, что данные, используемые прикладными программами, представляются в одном из стандартных форматов. В качестве примера можно назвать разработанный ISO стандарт *абстрактной синтаксической записи версии 1*, или ASN.1 (Abstract Syntax Notation). Этот стандарт используется для представления данных семейства протоколов TCP/IP, который называется SNMP.
- **Уровень приложений.** К седьмому уровню относятся прикладные программы, предназначенные для работы в сети. В качестве примера можно назвать программу для работы с электронной почтой или программу передачи файлов FTP. В частности, ITU был разработан специальный стандарт для работы с электронной почтой, который называли X.400. Фактически, организации ITU и ISO работали вместе над системой обработки сообщений; версия стандарта ISO называется MOTIS.

### 11.5.1. Пятиуровневая модель протокола TCP/IP

Второй вариант многоуровневой модели не был разработан одним из комитетов по стандартизации, а появился в результате исследований при создании семейства протоколов TCP/IP. С небольшими изменениями модель ISO может применяться и для описания уровней протокола TCP/IP. Однако следует помнить о том, что в основе модели ISO и семейства протоколов TCP/IP лежат совершенно разные допущения.

Семейство протоколов TCP/IP разбито на пять абстрактных уровней: четыре программных и нижний — аппаратный. На рис. 11.5 показаны абстрактные уровни протокола TCP/IP и передаваемые между ними данные.



*Рис. 11.5. Четыре абстрактных программных уровня семейства протоколов TCP/IP и пятый — аппаратный уровень, а также передаваемые между уровнями данные. Уровень сетевого интерфейса иногда называют канальным уровнем*

- **Уровень приложений.** Самый верхний уровень предназначен для предоставления пользовательским программам доступа к службам Internet. Программы взаимодействуют с одним из протоколов транспортного уровня для приема или передачи данных. Каждая программа выбирает нужный ей транспортный протокол, передача по которому может осуществляться либо в виде отдельных сообщений, либо в виде последовательного потока байтов. Пользовательская программа передает данные в нужном формате транспортному уровню для доставки.
- **Транспортный уровень.** Основной задачей транспортного уровня является обеспечение передачи данных между программами. Подобный тип связи часто называют *сквозным (end-to-end)*. На транспортном уровне может выполняться управление потоками данных. Кроме того, на этом уровне реализована надежная система доставки, гарантирующая поступление данных получателю без ошибок и в нужной последовательности. При этом используется режим подтверждения получения данных и повторная передача потерянных пакетов. Программное обеспечение транспортного уровня разделяет входной поток данных на небольшие части, называемые *пакетами*, и, дополнив каждый пакет адресом получателя, передает их программам протокола нижнего уровня для пересылки.

Хотя на рис. 11.5 уровень приложений показан в виде одного блока, на компьютере пользователя обычно запущено несколько программ, одновременно работающих с объединенной сетью. Поэтому программы транспортного уровня должны получать данные от нескольких пользовательских программ и пересыпать их программам нижнего уровня. Для этого к каждому пакету добавляется специальный код, идентифицирующий программу, которая отправила пакет, и ту, которая должна его получить на приемной стороне, а также контрольная сумма. После получения пакета по контрольной сумме проверяется его целостность, а затем по идентификационному коду определяется программа, которой данный пакет адресован.

- *Уровень протокола IP.* Как уже было сказано, программное обеспечение этого уровня обеспечивает взаимодействие двух узлов сети. Оно получает пакет с транспортного уровня, содержащий идентификационные данные той машины, которой он должен быть отослан. Пакет помещается в IP-дейтаграмму, после чего заполняются поля ее заголовка. Далее, с помощью алгоритма маршрутизации определяется, можно ли доставить дейтаграмму непосредственно получателю или она должна быть переслана маршрутизатору. После этого дейтаграмма передается программам поддержки сетевого интерфейса для дальнейшей передачи. Программное обеспечение уровня протокола IP обрабатывает также входящие дейтаграммы. При этом сначала проверяется их целостность, а затем с помощью алгоритма маршрутизации принимается решение о том, должна ли дейтаграмма обрабатываться на локальном компьютере или ее следует перенаправить другой машине. Из дейтаграммы, адресованной локальной машине, программа протокола IP удаляет заголовок и выбирает нужный протокол транспортного уровня для ее обработки. И наконец, на уровне протокола IP принимаются и отправляются ICMP-сообщения, извещающие об ошибках или выполняются различные управляющие действия.
- *Уровень сетевого интерфейса.* Программы самого нижнего уровня семейства протоколов TCP/IP обеспечивают работу сетевого оборудования, в частности платы сетевого интерфейса. Они принимают IP-дейтаграммы и передают их по физической сети. Сетевой интерфейс может состоять из драйвера устройства (если машина непосредственно подключена к локальной сети) или сложной подсистемы со своими протоколами канального уровня (когда в сети имеются коммутаторы пакетов, которые взаимодействуют с компьютерами пользователей с помощью протокола HDLC).

## 11.6. Отличие модели ISO от семейства протоколов TCP/IP

Между уровнями протокола TCP/IP и модели ISO/X.25 имеется два основных отличия: это различные подходы к надежности, а также к управлению системой и распределению вычислительных средств.

### 11.6.1. Различие в надежности

Основное отличие между протоколами TCP/IP и X.25 заключается в различном подходе к обеспечению надежности служб передачи данных. В модели X.25 обнаружение и обработка ошибок выполняется протокольным программным обеспечением на всех уровнях. Совокупность протоколов канального уровня гарантирует корректную передачу данных между компьютером пользователя и коммутатором пакетов. Для этого каждый передаваемый блок данных дополняется контрольной суммой, а после его получения отправителю посыпается подтверждение о доставке. Программы канального уровня устанавливают максимальный интервал времени для завершения передачи блока данных. Для предотвращения потери данных и обеспечения автоматического восстановления работоспособности сети при отказе и последующем ремонте оборудования в программы канального уровня включен специальный алгоритм повторной передачи.

Каждый последующий уровень семейства протоколов X.25 обеспечивает требуемую степень надежности. Так, на третьем уровне протокола X.25 производится обнаружение ошибок и восстановление передаваемых по сети пакетов данных с помощью контрольных сумм, отслеживания тайм-аутов и выполнения повторной передачи. На четвертом уровне должна обеспечиваться надежность сквозного

соединения. Для этого отправитель просит получателя проверить правильность полученных данных.

В отличие от описанной выше схемы, в многоуровневой модели семейства протоколов TCP/IP подразумевается, что проблема надежности должна решаться комплексно на всем пути следования дейтаграмм от отправителя до конечного получателя. Подход к проектированию был очень простым: объединенная сеть должнаправляться с ожидаемой нагрузкой. При этом допускались потери или искажение данных в отдельных каналах связи или на некоторых компьютерах, причем их немедленное восстановление не требовалось. Фактически в большинстве программ протокола TCP/IP, работающих на уровне сетевого интерфейса, проблема надежности решена частично или не решена вовсе. Обнаружение ошибок и их восстановление выполняется в основном на транспортном уровне.

Отсутствие проверок на уровне сетевого интерфейса позволило упростить реализацию программ протокола TCP/IP, сделать их более легкими для восприятия, а значит, уменьшить в них количество ошибок. Так, промежуточным маршрутизаторам разрешено аннулировать дейтаграммы, которые не могут быть доставлены получателю или целостность которых была нарушена в результате возникновения ошибок при передаче. Потеря дейтаграмм может происходить и в том случае, если у компьютера не хватает ресурсов для обработки входящего потока данных. Маршрутизаторы могут также перенаправлять дейтаграммы по другому пути с меньшим или большим временем задержки, не ставя при этом в известность отправителя или получателя.

Существование ненадежных каналов связи означает, что отдельные дейтаграммы могут не достичь конечного получателя. Обнаружение ошибок и восстановление утерянных дейтаграмм производится на всем пути следования от отправителя до конечного получателя. Поэтому такой тип контроля иногда называют *сквозным (end-to-end verification)*. Для управления процессом передачи в программах протокола TCP/IP транспортного уровня используются контрольные суммы, подтверждения и тайм-ауты. Таким образом, в отличие от систем, требующих установки соединения с получателем, в которых применяется иерархия протоколов X.25, в семействе протоколов TCP/IP основное внимание надежности уделяется только на одном уровне.

### **11.6.2. Распределение вычислительных средств и принятие решений**

Еще одно различие между иерархическими моделями X.25 и TCP/IP появляется при рассмотрении места в них органов управления. Обычно сети X.25 используются исключительно как средство для транспортировки данных. При этом управлением доступом к сети, наблюдением за трафиком, обслуживанием учетных записей пользователей и выставлением счетов за оплату занимается фирма, предоставляющая это транспортное средство. Эта же фирма для обеспечения надежности передачи пакетов самостоятельно решает проблемы маршрутизации, управления потоками данных и подтверждения их доставки. При таком подходе от узлов сети практически не требуется выполнение каких-либо дополнительных действий. Короче говоря, сеть представляет собой сложную независимую систему, к которой могут подключаться относительно простые компьютеры пользователей. При этом сами пользовательские компьютеры практически не участвуют в работе сети.

В отличие от этого в семействе протоколов TCP/IP требуется, чтобы узлы сети участвовали в работе практически всех сетевых протоколов. Выше уже шла речь о том, что узлы сети активно участвуют в процессе сквозного обнаружения ошибок и восстановления данных. Кроме того, они непосредственно участвуют в маршрутизации данных, поскольку перед отправкой дейтаграммы узел сети должен определить адрес маршрутизатора, которому следует переслать пакет. Узлы сети участвуют также в управлении сетью, поскольку они должны

обрабатывать ICMP-сообщения. Таким образом, по сравнению с X.25, объединенная сеть на основе протокола TCP/IP может рассматриваться как относительно простая система доставки пакетов, к которой подключены компьютеры пользователей, обладающие достаточно развитой логикой.

## 11.7. Принцип разделения протоколов на уровни

Независимо от используемой схемы разделения на уровни или функциональных возможностей каждого уровня, в основу работы работы многоуровневого семейства протоколов положена одна замечательная идея. Эта идея, называемая *принципом разделения на уровни*, может быть сформулирована следующим образом.

*Многоуровневые протоколы проектируются так, чтобы программы протокола, находящиеся на уровне n машины получателя, поступали на обработку точно такие же объекты, какие были сформированы аналогичной программой, находящейся на уровне n машины отправителя.*

Принцип разделения на уровни позволяет разработчикам при проектировании программ сосредоточиться на одном из уровней, не заботясь о других. Например, при создании программы передачи файлов программисту нужно продумать только взаимодействие копий программы, запущенной на двух компьютерах, и сосредоточиться только на сообщениях, необходимых для передачи файлов. При этом разработчик предполагает, что программа, работающая на приемном узле получит те же данные, которые были посланы с машины отправителя. Принцип разделения на уровни в действии продемонстрирован на рис. 11.6.

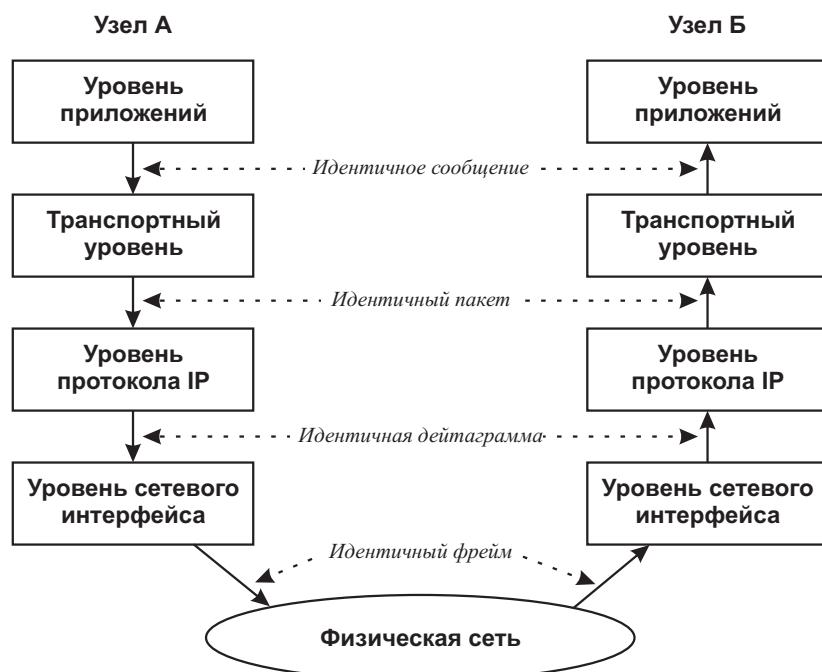


Рис. 11.6. Схема прохождения сообщения от прикладной программы узла А до аналогичной прикладной программы узла Б. Уровню n узла Б передаются те же объекты, которые были отосланы с уровня n узла А

### 11.7.1. Иерархия уровней протоколов в объединенной сети TCP/IP

Приведенный в предыдущем разделе принцип разделения протоколов на уровни кажется несколько туманным. Не внес ясности и рис. 11.6, поскольку на нем не отражена принципиальная разница между прохождением пакетов по одной или нескольким сетям. Эта разница продемонстрирована на рис. 11.7, где показан процесс передачи сообщения через маршрутизатор от прикладной программы, запущенной на одной машине, до прикладной программы, работающей на другой машине.

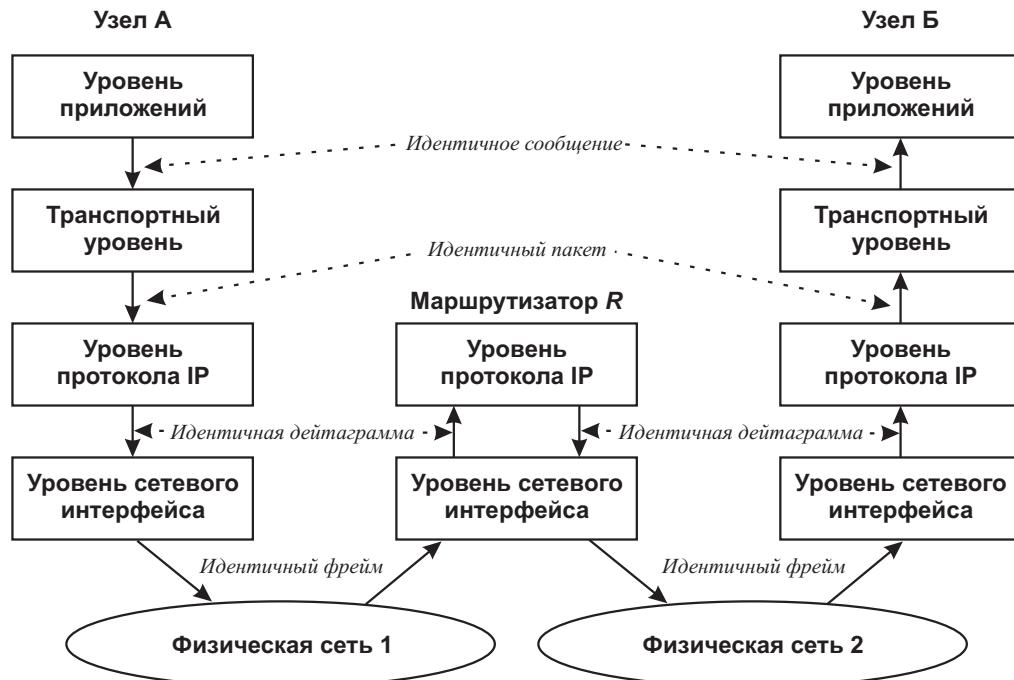


Рис. 11.7. Иллюстрация принципа разделения протоколов на уровни при использовании маршрутизатора. Фрейм, полученный маршрутизатором R, в частности соответствует фрейму, отправленному узлом А, но отличается от фрейма, передаваемого между узлами R и Б

Как видно из рис. 11.7, сообщения доставляются с помощью фреймов двух изолированных сетей. Один из них используется для передачи сообщения от машины А до маршрутизатора R, а другой — от маршрутизатора R до машины Б. Согласно принципам разбиения на уровни фрейм, полученный устройством R должен быть идентичен фрейму, отосланному машиной А. В отличие от уровня сетевого интерфейса, прикладной и транспортный уровни отправителя и конечного получателя непосредственно связаны между собой. Другими словами программное обеспечение, запущенное на машине отправителя взаимодействует с программами соответствующего уровня, работающими на машине конечного получателя. Таким образом, сформулированный выше принцип разделения протоколов на уровни прекрасно работает и в объединенной сети, но только для транспортного и прикладного уровней. То есть пакет, посланный программой транспортного уровня отправителя, идентичен пакету, переданному на обработку программе аналогичного уровня конечного получателя.

Нетрудно понять, что на верхних уровнях протоколов принцип разделения распространяется на всю цепочку взаимодействия от отправителя до конечного получателя. В то же время на самом нижнем уровне этот принцип можно применить только к операциям взаимодействия между машинами, подключенными к одной физической сети. Остается неясным, как быть с уровнем протокола IP. С одной стороны, мы уже говорили о том, что для любого подключенного к объединенной сети узла она является одной большой виртуальной сетью, поскольку IP-дейтаграммы при передаче инкапсулируются в сетевые фреймы. Дейтаграммы передаются от отправителя до конечного получателя, а принцип разделения протоколов на уровня гарантирует, что конечному получателю будет доставлена точно такая же дейтаграмма, которая была послана в объединенную сеть отправителем. С другой стороны, известно, что в заголовке дейтаграммы есть поля (например, счетчик *времени жизни*), значение которых изменяется при прохождении дейтаграммы через маршрутизатор. Исходя из этого можно сказать, что конечный получатель *никогда* не получит *точную* копию дейтаграммы, посланной отправителем. Таким образом, подводя итоги можно сказать, что хотя при передаче по объединенной сети большая часть дейтаграммы остается неизменной, принцип разделения протоколов на уровня применим только к машинам, подключенными к одной физической сети. То есть уровень протокола IP не обеспечивает сквозной режим обслуживания.

## 11.8. Многоуровневая структура сложной сети

В главе 2, “Обзор основных сетевых технологий”, уже шла речь о том, что в некоторых глобальных сетях может использоваться несколько коммутаторов пакетов. Например, типичная глобальная сеть может состоять из ряда маршрутизаторов, территориально расположенных в сетевых центрах и соединенных между собой выделенными последовательными каналами связи. Каждый маршрутизатор связывает локальную сеть центра с глобальной сетью. При поступлении дейтаграммы маршрутизатор должен либо доставить ее получателю в локальной сети, либо передать другому маршрутизатору по последовательному каналу связи. При этом возникает вопрос: как программы протоколов, обеспечивающие работу последовательного канала связи, вписываются в многоуровневую структуру семейства протоколов TCP/IP? Ответ зависит от того, какая у разработчика была точка зрения на структуру объединения последовательных каналов.

С точки зрения протокола IP совокупность двухточечных соединений между маршрутизаторами может функционировать либо как ряд независимых физических сетей, либо как единая физическая сеть. В первом случае, каждое физическое соединение рассматривается как любая другая сеть, входящая в структуру объединенной сети. Ему назначается уникальный номер сети, а для обеспечения связи между двумя узлами, находящимся на обоих концах соединения, им присваиваются уникальные IP-адреса<sup>3</sup>. В таблицы маршрутизации, расположенные на устройствах маршрутизации добавляется соответствующая информация о двухточечной сети, точно также как и для любой другой физической сети. И хотя для управления последовательным каналом связи в структуру протокольного программного обеспечения маршрутизатора добавляется новый модуль, относящийся к уровню сетевого интерфейса, это не вносит существенных изменений в общую многоуровневую структуру семейства протоколов TCP/IP. Очевидно, что если каждую двухточечную сеть рассматривать как независимую физическую

<sup>3</sup> За исключением случаев использования анонимной сети, описанной в главе 10, “Бесклассовая адресация и подсети (CIDR)”. Однако следует отметить, что ненумерованное соединение не влияет на многоуровневую структуру сети.

сеть, то это приведет к увеличению общего количества номеров сетей (поскольку, по определению, каждому соединению между двумя компьютерами должен быть назначен уникальный номер сети) и, как следствие, — к увеличению размеров таблиц маршрутизации. Несмотря на этот недостаток, в обоих протоколах — *SLIP* (*Serial Line IP*, или *межсетевой протокол для последовательного канала связи*) и *PPP* (*Point to Point Protocol*, или *протокол двухточечного соединения*), — используемых для связи по последовательным каналам, каждое соединение рассматривается как отдельная сеть.

Второй подход к проблеме двухточечных соединений не предусматривает назначение каждому физическому соединению отдельного IP-адреса сети. Совокупность всех соединений рассматривается как единая и независимая IP-сеть со своим форматом фрейма, системой физической адресации и протоколами канального уровня. При таком подходе для обеспечения связи по всем двухточечным сетям маршрутизатору достаточно знать только номер одной IP-сети.

Для реализации второго подхода необходимо расширить существующую многоуровневую структуру семейства протоколов TCP/IP и добавить в нее новый уровень *внутрисетевой маршрутизации*, который следует поместить между уровнем сетевого интерфейса и сетевым оборудованием. В случае машины, имеющей только одно двухточечное соединение, введение дополнительного уровня кажется излишним. Однако, чтобы понять необходимость дополнительного уровня, следует рассмотреть машину, имеющую несколько двухточечных соединений, и вспомнить программную модель семейства протоколов TCP/IP, показанную на рис. 11.2. Мы уже говорили о том, что уровень сетевого интерфейса составляют несколько программных модулей, каждый из которых управляет работой одной физической сети. Таким образом, нам необходимо добавить еще один модуль сетевого интерфейса, обслуживающего новую двухточечную сеть, причем этот модуль должен управлять работой нескольких сетевых устройств. Более того, перед отправкой дейтаграммы модуль сетевого интерфейса должен выбрать физический канал, по которому ее следует послать (рис. 11.8).

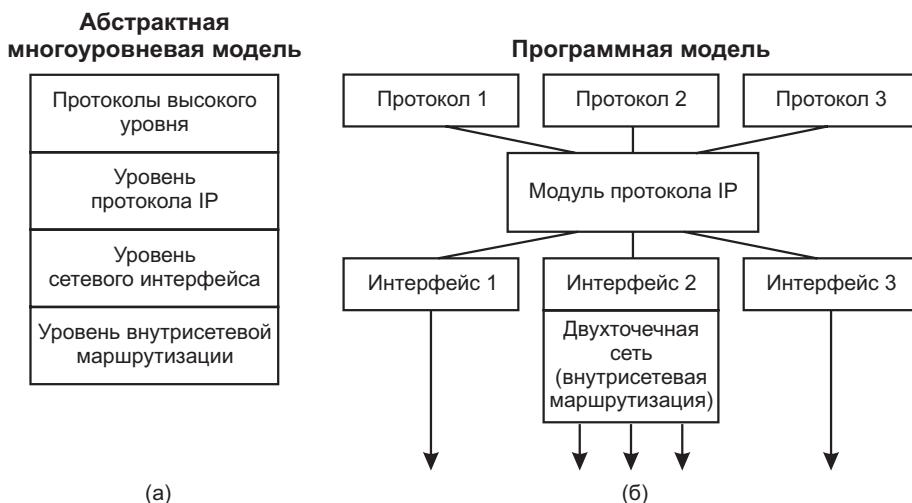


Рис. 11.8. Место уровня внутрисетевой маршрутизации в абстрактной многоуровневой модели, когда отдельные двухточечные соединения рассматриваются как одна IP-сеть (а); детализация построения соответствующих программных модулей (б). Каждая стрелка соответствует одному физическому устройству

Все дейтаграммы, которые должны быть посланы через одно из двухточечных соединений, передаются программами протокола IP на уровень сетевого интерфейса. В свою очередь программы уровня сетевого интерфейса передают дейтаграммы модулю внутрисетевой маршрутизации, где они анализируются и определяется физическое соединение, через которое дейтаграммы должны быть посланы во внешний мир.

При разработке модуля внутрисетевой маршрутизации программисту нужно знать, как выбрать соответствующее физическое соединение. Как правило, все алгоритмы основаны на таблице внутрисетевой маршрутизации. Она напоминает обычную таблицу маршрутизации, используемую в объединенной сети, в которой указываются соответствия между адресом конечного получателя и маршрутом следования дейтаграмм. В таблице внутрисетевой маршрутизации содержатся пары элементов ( $D, L$ ), где  $D$  — адрес конечного получателя, а  $L$  — номер физического соединения, через которое можно достичь этого получателя.

В отличие от таблицы маршрутизации объединенной сети, таблица внутрисетевой маршрутизации имеет гораздо меньший размер. В ней содержится информация о маршрутах только к тем узлам, которые непосредственно подключены к одному из двухточечных соединений. Причина, должно быть, уже понятна. Перед передачей дейтаграммы уровню сетевого интерфейса, программы протокола IP определяют по IP-адресу ее конечного получателя адрес ближайшего узла маршрутизации. Таким образом, на уровне внутрисетевой маршрутизации остается только определить какому из компьютеров, подключенных к одной из двухточечных сетей, следует отправить дейтаграмму.

## 11.9. Две основные границы раздела многоуровневой модели TCP/IP

В абстрактной многоуровневой модели семейства протоколов TCP/IP имеются две границы раздела, которые не столь очевидны. Одна из них разделяет системы адресации (на логическую и физическую), используемые в семействе протоколов TCP/IP, а другая отделяет программы операционной системы от прикладных программ.

### 11.9.1. Граница раздела логических адресов

После рассмотрения многоуровневой модели семейства протоколов TCP/IP пришла пора подробнее рассмотреть понятие, описанное в главе 8, “Протокол IP: маршрутизация дейтаграмм”. Речь идет об абстрактной границе, разделяющей программное обеспечение, которое использует низкоуровневые (физические) адреса, и программы, использующие высокоуровневые (логические, или IP) адреса. Как показано на рис. 11.9, граница раздела проходит между уровнем сетевого интерфейса и протокола IP. Таким образом,

*в прикладных программах, а также во всех программах поддержки протоколов, расположенных в абстрактной модели выше уровня протокола IP, используются только IP-адреса. Программы, относящиеся к уровню сетевого интерфейса, оперируют физическими адресами.*

Таким образом, протоколы наподобие ARP, оперирующие физическими адресами, относятся к уровню сетевого интерфейса. Они не являются частью протокола IP.

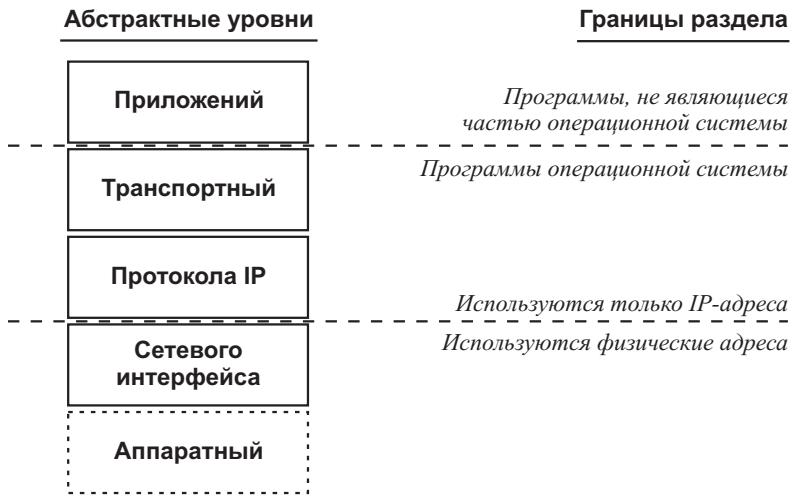


Рис. 11.9. Две границы раздела абстрактной многоуровневой модели семейства протоколов TCP/IP

### 11.9.2. Граница раздела программ операционной системы и пользовательских приложений

На рис. 11.9 показана еще одна важная граница, которая разделяет программы, являющиеся частью операционной системы, и пользовательские приложения. В некоторых реализациях протокола TCP/IP эта граница может быть проведена по-другому, однако в большинстве реализаций придерживаются предложенной выше схемы. Поскольку программы протоколов нижнего уровня входят в состав операционной системы, данные между ними передаются с большей эффективностью, чем между пользовательскими приложениями и программами протоколов транспортного уровня. Эта проблема подробно описана в главе 20, “Взаимодействие частных сетей (NAT, VPN)”; там же приводится пример интерфейса, который должна поддерживать операционная система.

## 11.10. Недостатки многоуровневой модели

Уже было сказано о том, что многоуровневая модель является основополагающим принципом при разработке сетевых протоколов. Это позволяет разработчику разделить сложную проблему на несколько подзадач и решить каждую из них отдельно. К сожалению, программы, при разработке которых строго придерживались принципа многоуровневой модели, часто оказываются крайне неэффективными. В качестве примера рассмотрим функции программного обеспечения протокола транспортного уровня. Оно должно принимать поток байтов, поступающих от прикладной программы, разбивать его на пакеты и передавать каждый пакет по объединенной сети. Чтобы оптимизировать процесс передачи данных, программа транспортного уровня должна выбрать максимально возможный размер пакета, но чтобы он целиком помещался в одном сетевом фрейме. В частности, когда машина получателя напрямую подключена к одной сети с машиной отправителя, данные будут передавать только по одной физической сети. Поэтому отправитель всегда может выбрать оптимальный размер пакета для этой сети. Если же при разработке программного

обеспечения строго придерживаться принципа многоуровневой модели, то программы транспортного уровня не будут “знать”, как маршрутизируется трафик в модуле протокола IP и к каким сетям эта машина подключена напрямую. Более того, на транспортном уровне не должны определяться форматы дейтаграммы и фрейма, поскольку это — прерогатива протоколов более низкого уровня. Таким образом, программы транспортного уровня не смогут определить, какое количество октетов данных будет добавлено к пакету в качестве заголовков программами нижнего уровня. Становится очевидным, что строгое следование принципу многоуровневой модели не позволяет программам транспортного уровня оптимизировать процесс передачи данных.

Обычно при написании программного обеспечения разработчики строго не следуют принципам многоуровневой модели. Они предусматривают возможность определения параметров низкоуровневых протоколов (таких как маршруты следования или значение MTU) программами старших уровней. При выделении памяти под буферы в них часто резервируется место для размещения заголовков, которые будут добавлены к пакету программами протоколов низкого уровня. Кроме того, при передаче полученного пакета на обработку программам протоколов старшего уровня иногда сохраняются заголовки входящих фреймов. Подобные методы оптимизации могут существенно повысить эффективность работы программ протоколов, не нарушая основную многоуровневую структуру.

## 11.11. Основная идея мультиплексирования и демультиплексирования

Технологии *мультиплексирования* (*multiplexing*) и *демультиплексирования* (*demultiplexing*) используются на всех уровнях иерархии протоколов. К отправляемому сообщению компьютер добавляет несколько битов, которые определяют тип сообщения, прикладную программу и используемый протокол. Перед передачей по сети сообщение помещается в физический фрейм, а затем на приемном конце восстанавливается в поток пакетов. Благодаря дополнительным битам машина может корректно обработать полученные данные. Рассмотрим пример демультиплексирования входящих фреймов, показанный на рис. 11.10.

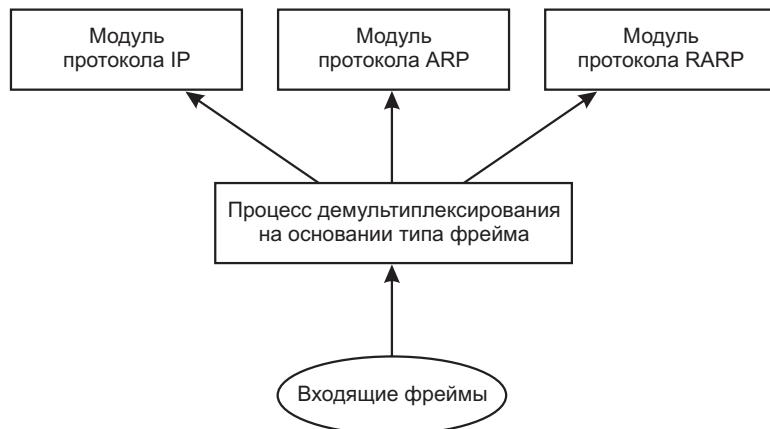
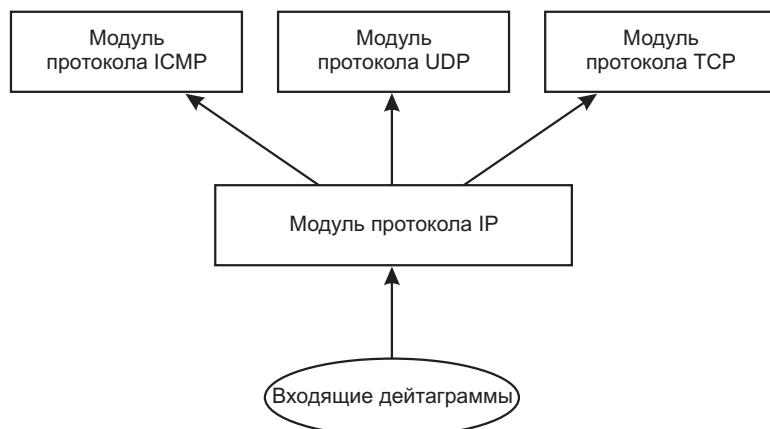


Рис. 11.10. Демультиплексирование входящих фреймов на основании информации, находящейся в поле типа их заголовков

На рис. 11.10 проиллюстрирован процесс выбора модуля для обработки входящих фреймов, который выполняют программы уровня сетевого интерфейса в зависимости от типа фрейма. Говорят, что программы сетевого интерфейса *демультиплексируют* входящие фреймы на основании их типа. Для того чтобы этот процесс стал возможен, перед отправкой фрейма, программа устанавливает значение поля типа в его заголовке. Таким образом, любой программный модуль, который отправляет фрейм, определяет его содержимое, установив соответствующее значение поля типа в его заголовке.

Процесс мультиплексирования и демультиплексирования выполняется почти на всех уровнях протоколов. Например, после демультиплексирования фреймов модулем сетевого интерфейса полученные IP-дейтаграммы передаются на обработку модулю протокола IP. Программы протокола IP восстанавливают из дейтаграмм пакеты данных и выполняют дальнейшее их демультиплексирование на основании типа протокола транспортного уровня (рис. 11.11).



*Рис. 11.11. Демультиплексирование дейтаграмм на уровне протокола IP. Модуль IP выбирает нужную процедуру для обработки дейтаграммы на основании типа протокола, указанного в ее заголовке*

Для определения способа обработки дейтаграмм программы протокола IP анализируют их заголовки и в зависимости от типа дейтаграммы выбирают соответствующий модуль протокола транспортного уровня. Например, входящие дейтаграммы могут относиться к одному из протоколов: *ICMP* (он был рассмотрен в главе 9, “Протокол IP: обработка ошибок и управляющие сообщения (ICMP)”), *UDP* и *TCP* (последние два протокола будут рассмотрены в следующих главах).

## 11.12. Резюме

Протоколы представляют собой стандарты, которые определяют способ представления данных при их передаче от одной машины к другой. В протоколах оговариваются способы передачи данных, подтверждений и обнаружения ошибок. Для упрощения разработки и реализации протокольного программного обеспечения общая проблема обмена данными между компьютерами разделяется на отдельные подзадачи, которые могут быть решены независимо. Для решения каждой подзадачи разрабатывается отдельный протокол.

Идея многоуровневого подхода при решении общей коммуникационной задачи является основополагающей, поскольку она обеспечивает концептуальную основу для разработки протоколов. В многоуровневой модели на каждом уровне

решается одна из частей коммуникационной проблемы; обычно один уровень соответствует одному протоколу. При разработке программ протоколов соблюдается многоуровневый подход. Это означает, что на вход программного обеспечения, функционирующего на уровне  $n$  машины получателя, поступает точно такая же информация, которая была на выходе программы уровня  $n$  машины отправителя.

В этой главе была рассмотрена базовая пятиуровневая модель семейства протоколов TCP/IP и старая семиуровневая модель ISO. В обоих случаях, многоуровневая модель является только концептуальной основой для разработки программ поддержки протоколов. В основу семейства протоколов X.25, которые являются примером надежных коммуникационных служб, используемых в коммерческих целях, положена базовая модель ISO. В то же время в семействе протоколов TCP/IP применена совершенно другая многоуровневая схема.

На практике для выбора модуля нужного уровня, который должен обработать входящие пакеты, используются принципы мультиплексирования и демультиплексирования. Все это несколько усложняет реализацию алгоритма протокольного программного обеспечения по сравнению с тем алгоритмом, который описан в многоуровневой модели.

## Материал для дальнейшего изучения

В [RFC 791] Постел (Postel) предложил набросок многоуровневой структуры протокола IP, а Кларк (Clark) в [RFC 817] описал эффект применения многоуровневого подхода при реализации протокольного программного обеспечения. В статье [115] Зальцер (Saltzer), Рид (Reed) и Кларк (Clark) доказали важность сквозного контроля при передаче данных. В работе [21] Чессон (Chesson) выдвинул сомнительную идею по поводу того, что строгое следование принципам многоуровневой модели при создании протокольного программного обеспечения существенно снижает пропускную способность сети. Более подробно многоуровневый подход будет рассмотрен во втором томе этой книги. Там же будет продемонстрирован пример реализации семейства протоколов, в котором эффективность работы достигается за счет использования разумного компромисса между строгим соблюдением принципов многоуровневой модели и передачи указателей между программами разных уровней.

Протокол ISO описан в документе [59], а язык ASN.1 подробно рассматривается в документе [60]. Стандарт XDR фирмы Sun описывается в [RFC 1014]. Его можно рассматривать как пример протокола, который вызывает один из протоколов TCP/IP уровня представлений. Передача информации в восходящем порядке между программами протоколов разного уровня описана в работе Кларка [23].

## Упражнения

- 11.1.** Внимательно изучите многоуровневую модель ISO. Насколько точно в этой модели описан процесс передачи данных по локальной сети, такой как Ethernet?
- 11.2.** Опишите случай, когда в многоуровневую модель семейства протоколов TCP/IP можно добавить шестой уровень (представлений). (*Подсказка.* Учтите, что разные программы могут использовать протоколы XDR и ASN.1)
- 11.3.** Как вы считаете, может ли один протокол уровня представлений в конечном счете заменить все остальные протоколы? Обоснуйте свой ответ.

- 11.4.** Сравните теговый формат представления данных, используемый в протоколе ASN.1, от нетегового формата протокола XDR. Чем отличаются эти форматы представления данных? Опишите ситуацию, когда один из форматов может иметь преимущество.
- 11.5.** Выясните, как в системе UNIX для повышения эффективности работы многоуровневого протокольного программного обеспечения используется структура `mbuf`.
- 11.6.** Изучите *потоковый* (*streams*) механизм системы UNIX System V. Как его использование может упростить реализацию многоуровневого семейства протоколов? Назовите его основные недостатки?

# 12

## Передача пользовательских дейтаграмм (UDP)

### 12.1. Введение

В предыдущей главе был рассмотрен способ передачи IP-дейтаграмм через объединенную сеть TCP/IP. Напомним, что маршрутизация каждой дейтаграммы выполняется на основе IP-адреса конечного получателя. С точки зрения протокола IP, этот адрес определяет только компьютер пользователя в сети. При этом не уточняется, какому именно пользователю или прикладной программе следует доставить дейтаграмму. В этой главе продолжается изучение семейства протоколов TCP/IP. Будет рассмотрен механизм, позволяющий получателю различать входящие дейтаграммы и определять, какой из прикладных программ они адресованы. Этот же механизм позволяет нескольким программам отправлять и получать дейтаграммы независимо друг от друга.

### 12.2. Идентификация конечного получателя

Операционные системы большинства компьютеров поддерживают так называемый *мультипрограммный* (*multiprogramming*) режим работы, при котором обеспечивается одновременное и независимое выполнение нескольких пользовательских программ. В различных операционных системах выполняющиеся программы могут называться по-разному: *процесс* (*process*), *задача* (*task*), *приложение* (*application program*) или *пользовательский процесс* (*user level process*). При этом сама операционная система называется *многозадачной* (*multitasking*). Исходя из этого вполне правомерно сказать, что конечным получателем дейтаграмм является процесс, запущенный на узле сети с указанным IP-адресом. Однако подобное утверждение может ввести в заблуждение. Во-первых, поскольку процессы создаются и уничтожаются динамически, отправитель редко имеет достаточное количество информации о процессе, запущенном на другой машине. Во-вторых, все процессы, принимающие дейтаграммы на конкретном компьютере, могут быть заменены без уведомления отправителей. Например, перезагрузка компьютера вызывает замену всех запущенных на нем процессов, однако стоит ли об этом уведомлять всех потенциальных отправителей дейтаграмм? В-третьих, получатель дейтаграмм должен идентифицироваться по выполняемым им функциям, а не по процессу, в котором эти функции реализованы. Другими словами, отправитель должен иметь возможность подключиться к файловому серверу, не выясняя, в каком именно процессе на машине получателя реализованы функции файлового сервера. Кроме того, в системах, позволяющих в одном

процессе обрабатывать две и более функций, необходимо обеспечить для процесса возможность определять, какая именно функция запрошена отправителем.

Из всего сказанного уже, должно быть, понятно, что процессы нельзя рассматривать в качестве конечного получателя дейтаграмм. Необходимо сделать так, чтобы на каждой машине имелся набор абстрактных получателей, называемых *портами протокола (protocol ports)*. Каждому порту протокола поставлено в соответствие целое положительное число. При этом в локальной операционной системе должны быть предусмотрены средства идентификации портов и доступа к ним.

В большинстве операционных систем используется синхронный доступ к портам. С точки зрения процесса это означает, что на время доступа к порту, операционная система приостанавливает его выполнение. Например, если процесс попытается считать данные из порта до того, как они будут получены, операционная система приостановит (заблокирует) его до поступления данных. Когда данные будут получены, операционная система передаст их процессу и возобновит его выполнение. Обычно в портах используется *буферизация данных*, поэтому данные не будут потеряны, даже если поступят раньше, чем процесс сможет их принять. При буферизации протокольное программное обеспечение, находящееся внутри операционной системы помещает полученные для определенного порта пакеты в очередь конечного размера, откуда процесс сможет их извлечь.

Для связи с внешним портом отправителю необходимо знать IP-адрес машины получатели и номер порта на этой машине. В каждом сообщении должен быть указан *номер порта получателя*, находящегося на машине, которой послано сообщение, а также *номер порта отправителя*, которому должно быть адресовано ответное сообщение. Это позволяет любому процессу получать сообщение и отвечать отправителю.

### 12.3. Протокол передачи пользовательских дейтаграмм (UDP)

В семействе протоколов TCP/IP *протокол UDP (User Datagram Protocol)* обеспечивает основной механизм передачи дейтаграмм между двумя прикладными программами. Выбор нужной программы на компьютере получателя в протоколе UDP выполняется по номеру порта. Поэтому, кроме передаваемых данных, в каждом UDP-сообщении содержатся номера портов получателя и отправителя. То есть для программ протокола UDP, запущенных на компьютере конечного получателя, обеспечивается механизм доставки сообщений до нужной прикладной программы, а для самой прикладной программы — возможность ответить отправителю.

Для транспортировки сообщений от одной машины к другой в протоколе UDP используется протокол нижнего уровня — IP. Поэтому для протокола UDP (так же, как и для IP) характерна ненадежная доставка дейтаграмм без установки соединения с получателем. В нем не используется механизм подтверждения доставки сообщений. В протоколе UDP не отслеживается порядок получения пакетов, а также не обеспечивается обратная связь для управления скоростью информационного потока между машинами. В результате сообщения UDP могут быть утеряны, переданы повторно или получены в неправильном порядке. Более того, в течение некоторого промежутка времени может быть получено больше пакетов, чем получатель сможет их обработать. Таким образом, можно подвести итог.

*Протокол передачи пользовательских дейтаграмм (User Datagram Protocol, или UDP) обеспечивает ненадежную доставку сообщений без установки соединения с получателем. При этом для транспортировки сообщений между*

машинами используется протокол IP. В отличие от IP, в протоколе UDP предусмотрена возможность выбора заданного получателя на конкретной машине.

При использовании протокола UDP за надежность доставки отвечает прикладная программа. Это подразумевает обработку ситуаций потери сообщений, их дублирования, возникновения задержек, нарушения порядка, а также потери связи с получателем. К сожалению, программисты, разрабатывающие прикладные программы, часто не уделяют должного внимания перечисленным проблемам. Более того, поскольку для тестирования сетевых программ, как правило, используется локальная сеть, отличающаяся высокой надежностью и малым временем задержки при передаче пакетов, на этапе тестирования все потенциальные проблемы могут быть не выявлены. Поэтому, даже если программа, использующая протокол UDP, хорошо работает в локальной сети, она может некорректно работать в большой объединенной сети на основе протокола TCP/IP.

## 12.4. Формат UDP-сообщения

Любое сообщение протокола UDP называется *пользовательской дейтаграммой* (*user datagram*). Пользовательская дейтаграмма состоит из двух частей: UDP-заголовка и области данных. Как показано на рис. 12.1 заголовок разделен на четыре 16-битовых поля, в которых указываются номера портов отправителя и получателя, длина сообщения и контрольная сумма UDP-сообщения.

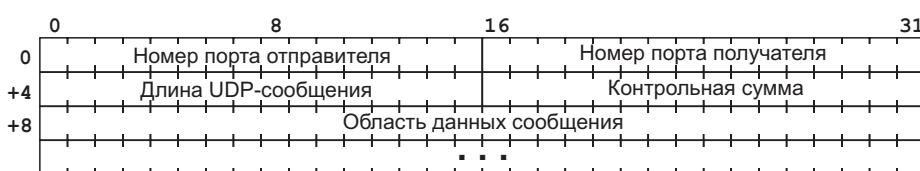


Рис. 12.1. Формат пользовательской дейтаграммы

Как видно из рис. 12.1, в первых двух 16-битовых полях заголовка пользовательской дейтаграммы указываются *номера портов отправителя и получателя*. Последний используется программой протокола UDP для распределения поступивших дейтаграмм между ожидающими их процессами. Номер *порта отправителя* указывать необязательно. Если порт отправителя не указан, в этом поле должно содержаться нулевое значение. В противном случае в этом поле задается номер порта, по которому получатель может послать ответ отправителю.

В поле *длины UDP-сообщения* указывается размер дейтаграммы в октетах, включая заголовок и пользовательские данные. Поэтому, минимальный размер UDP-дейтаграммы равняется восьми октетам (в том случае, если в сообщении отсутствуют пользовательские данные).

Поле *контрольной суммы* является необязательным и может вообще не использоваться. Если его значение равно нулю, значит, контрольная сумма пользовательской дейтаграммы не вычислялась. По замыслу разработчиков такой подход позволяет передавать UDP-дейтаграммы через высоконадежные локальные сети, с минимальными накладными расходами, связанными с вычислением и последующей проверкой контрольных сумм. Напомним, что в протоколе IP контрольная сумма вычисляется только для заголовка дейтаграммы и не вычисляется для поля данных. Таким образом, по значению контрольной суммы UDP-дейтаграммы можно судить о том, что в процессе доставки данные не были изменены.

Новички часто удивляются, узнав о том, что вычисленная контрольная сумма UDP-сообщения может равняться нулю. В этом нет ничего странного, поскольку в протоколе UDP используется тот же алгоритм вычисления контрольной суммы, что и в протоколе IP. То есть при подсчете контрольной суммы данные пользовательской дейтаграммы разбиваются на последовательности 16-разрядных целых чисел, которые затем суммируются. При сложении используется двоичная арифметика с представлением отрицательных чисел в инверсном коде (так называемое сложение с учетом знака). Затем полученный результат инвертируется, чтобы получилось положительное значение контрольной суммы. Самое удивительное то, что при использовании подобной арифметики проблема нуля, полученного в результате вычисления значения контрольной суммы, решается очень просто. Все дело в том, что при использовании инверсного кода для представления отрицательных чисел существует два нулевых значения: положительное (когда все биты числа равны нулю) и отрицательное (когда все биты числа равны единице). Поэтому, если при вычислении контрольной суммы пользовательской дейтаграммы окажется, что она равна нулю, используется отрицательное представление нуля.

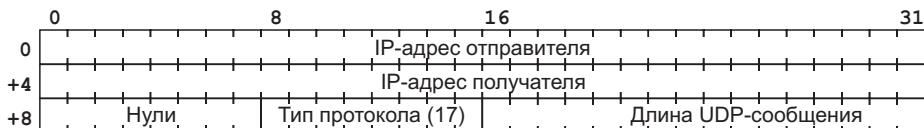
## 12.5. Псевдозаголовок пользовательской дейтаграммы

При вычислении контрольной суммы UDP-дейтаграммы используются не только данные, хранящиеся в дейтаграмме. Перед вычислением к UDP-дейтаграмме добавляется псевдозаголовок и общая длина полученной конструкции выравнивается на 16-битовую границу (в конец дейтаграммы добавляется необходимое количество нулевых октетов). После этого вычисляется контрольная сумма полученного объекта. Следует учесть, что нулевые октеты, добавленные к дейтаграмме для выравнивания ее длины, и сам псевдозаголовок *не* передаются конечному получателю. Они также не учитываются при вычислении длины дейтаграммы. При вычислении контрольной суммы значение поля контрольной суммы в заголовке дейтаграммы полагается равным нулю. После суммирования 16-битовых значений псевдозаголовка, заголовка UDP-дейтаграммы и области данных, полученное значение инвертируется, чтобы результат был неотрицательным.

Псевдозаголовок предназначен для проверки корректности доставки UDP-дейтаграммы заданному получателю. Для понимания происходящих процессов необходимо вспомнить, что конечный получатель дейтаграммы определяется IP-адресом компьютера и номером порта на этом компьютере. В заголовке UDP-дейтаграммы указывается только номер порта конечного получателя. Поэтому, чтобы гарантировать корректность доставки дейтаграммы, программы протокола UDP, работающие на машине отправителя, вычисляют контрольную сумму с учетом IP-адреса машины и данных UDP-дейтаграммы. На приемном конце программы протокола UDP проверяют значение контрольной суммы, используя значение IP-адреса, полученное из заголовка IP-дейтаграммы, в которой было доставлено сообщение UDP. Если контрольные суммы совпадают, значит, с высокой долей вероятности можно сказать, что дейтаграмма была корректно доставлена не только указанному компьютеру сети, но и заданному порту этого компьютера.

При вычислении контрольной суммы UDP-дейтаграммы используется псевдозаголовок размером 12 октетов, формат которого показан на рис. 12.2. В первых двух полях псевдозаголовка указываются IP-адреса отправителя и конечного получателя, которые помещаются в заголовок IP-дейтаграммы, содержащей UDP-сообщение. В поле типа протокола указывается код семейства протоколов TCP/IP (для протокола UDP он равен 17). В последнем поле псевдозаголовка указывается длина UDP-дейтаграммы без учета длины самого псевдозаголовка.

При проверке контрольной суммы получатель должен извлечь значения этих полей из заголовка IP-дейтаграммы, поместить их в соответствующие поля псевдо-заголовка и заново вычислить контрольную сумму.



*Рис. 12.2. Формат псевдо-заголовка, размером 12 октетов, который используется для вычисления контрольной суммы UDP-дейтаграммы*

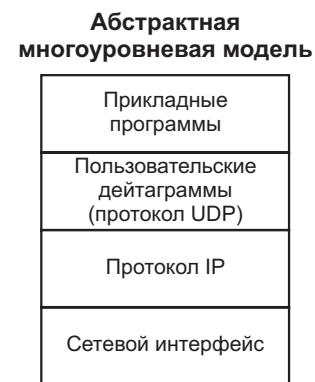
## 12.6. Инкапсуляция и разделение на уровни

В многоуровневой модели семейства протоколов TCP/IP, транспортный протокол UDP находится над протоколом IP. Теоретически обращаться к протоколу UDP должны пользовательские программы, которые применяют протокол IP для передачи и приема дейтаграмм (рис. 12.3).

Помещение протокола UDP над протоколом IP означает, что готовое к отправке UDP-сообщение, состоящее из UDP-заголовка и передаваемых данных, инкапсулируется в IP-дейтаграмму для последующей передачи по объединенной сети (рис. 12.4).

Для рассмотренных протоколов процесс инкапсуляции означает следующее. Пользовательская программа передает блок данных программе протокола UDP. Последняя добавляет к ним заголовок и формирует UDP-дейтаграмму, которая затем передается программам протокола IP. На уровне протокола IP полученная UDP-дейтаграмма рассматривается как обычные пользовательские данные. Поэтому к ней добавляется заголовок и формируется IP-дейтаграмма. На заключительном этапе IP-дейтаграмма передается программам сетевого интерфейса, которые перед передачей по участку физической сети от одной машины до другой помещают ее в физический фрейм. Формат физического фрейма зависит от используемой для передачи данных сетевой технологии. Как правило, физический фрейм содержит дополнительный заголовок.

После доставки пакета конечному получателю происходит процесс, обратный инкапсуляции. Т.е. вначале пакет поступает на обработку программам нижнего сетевого уровня, а затем “поднимается” вверх по иерархии протоколов. На каждом уровне из пакета удаляется один заголовок, после чего сообщение передается для дальнейшей обработки программам протоколов более высокого уровня. Процесс продолжается до тех пор, пока из сообщения не будут удалены все заголовки. В результате прикладной программе — получателю сообщения данные поступают без служебных заголовков. Таким образом, самый внешний заголовок передаваемого сообщения относится к протоколу самого нижнего уровня, соответственно, самый внутренний заголовок сообщения соответствует протоколу самого верхнего уровня. При рассмотрении процесса добавления и удаления заголовков важно не забывать о принципе разделения протоколов на абстрактные уровни. В частности, соблюдение многоуровневого принципа в применении к



*Рис. 12.3. В абстрактной многоуровневой модели протокол UDP занимает место между пользовательскими программами и протоколом IP*

протоколу UDP гарантирует, что пользовательская дейтаграмма, переданная из программы протокола IP на машине получателя, будет точно такой же, как дейтаграмма, которая была передана программам протокола IP на машине отправителя. Поэтому данные, которые были доставлены программой протокола UDP прикладной программе на машине получателя, будут такими же, как были переданы прикладной программой на уровень протокола UDP на машине отправителя. Таким образом, разделение обязанностей между различными протоколами строгое и четкое.

*Программы протокола IP отвечают только за передачу данных между двумя узлами объединенной сети. Программы протокола UDP выполняют разделение потоков данных между различными источниками или получателями в пределах одного компьютера.*

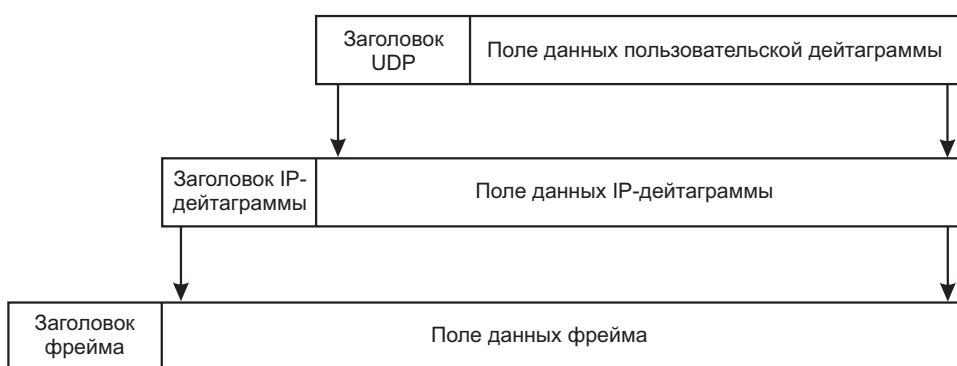


Рис. 12.4. Для передачи по объединенной сети UDP-дейтаграмма инкапсулируется в IP-дейтаграмму. В дальнейшем каждый раз перед передачей по заданному участку физической сети IP-дейтаграмма инкапсулируется в физический фрейм

Таким образом, информация об узлах отправителя и получателя находится только в IP-заголовке. Порты отправителя и получателя, с помощью которых разделяются потоки данных в пределах одной машины, указываются только в UDP-заголовке.

## 12.7. Вычисление контрольной суммы UDP-дейтаграммы при разделении на уровни

Внимательный читатель наверняка уже заметил явное противоречие между принципом разделения на уровни и вычислением контрольной суммы UDP-дейтаграммы. Напомним, что при вычислении контрольной суммы используется псевдозаголовок, в который помещаются IP-адреса отправителя и конечного получателя. Поэтому при передаче сообщения программам протокола UDP прикладной программе должен быть известен IP-адрес конечного получателя. Более того, прикладная программа должна его передать вместе с сообщением на уровень протокола UDP. Следовательно, программы протокола UDP могут определить IP-адрес получателя, не обращаясь к программам протокола IP. Однако IP-адрес отправителя зависит от принятого метода маршрутизации IP-дейтаграмм. Напомним, что IP-адрес идентифицирует сетевой интерфейс, через который передается IP-дейтаграмма. Поэтому для определения IP-адреса отправителя программы протокола UDP должны обратиться к программам протокола IP.

Исходя из сказанного можно предположить, что программы протокола UDP обращаются к уровню протокола IP для определения IP-адреса отправителя и, возможно, получателя. Затем эти адреса помещаются в псевдозаголовок, после чего вычисляется контрольная сумма UDP-дейтаграммы. Далее псевдозаголовок отбрасывается, и пользовательская дейтаграмма передается программам протокола IP для отправки ее по сети. Для повышения эффективности работы программ протокола UDP был придуман альтернативный подход. Суть его заключается в том, что пользовательская дейтаграмма инкапсулируется в IP-дейтаграмму на уровне протокола UDP. После определения IP-адреса отправителя с помощью программ протокола IP оба адреса (отправителя и получателя) заносятся в соответствующие поля заголовка IP-дейтаграммы. После вычисления контрольной суммы UDP-дейтаграммы, IP-дейтаграмма передается на уровень протокола IP. Программам протокола IP нужно заполнить только оставшиеся поля заголовка IP-дейтаграммы.

Как вы могли убедиться, между уровнями протоколов UDP и IP существует очень тесная связь. При этом возникает вопрос, не нарушает ли она нашу исходную предпосылку насчет жесткого разделения функциональных обязанностей между уровнями? Конечно нарушает! Протокол UDP тесно связан с протоколом IP. Разработчики пошли на явный компромисс из практических соображений. Они сознательно нарушили принцип разделения протоколов на уровне, поскольку невозможно указать координаты прикладной программы, которой предназначены пакеты, без указания адреса машины конечного получателя, на которой она запущена. Кроме того, желательно иметь эффективный механизм преобразования адресов, используемых как на уровне протокола UDP, так и на уровне протокола IP. В качестве упражнения читателю предлагается рассмотреть описанный подход с другой точки зрения: когда протокол UDP полностью изолирован от протокола IP.

## 12.8. Мультиплексирование и демультиплексирование UDP-дейтаграмм с помощью портов

В главе 11, “Уровни протоколов”, уже шла речь о том, что протокольное программное обеспечение каждого из уровней выполняет мультиплексирование и демультиплексирование объектов, предназначенных для обработки на соседнем уровне. В этом плане протокол UDP не является исключением. В его задачу входит прием пользовательских дейтаграмм, поступающих от различных прикладных программ, и пересылка их расположенному ниже уровню протокола IP для передачи по сети. Кроме того, протокол UDP получает пользовательские дейтаграммы, поступившие от протокола IP, и перенаправляет их соответствующей прикладной программе.

Теоретически мультиплексирование и демультиплексирование дейтаграмм между программами протокола UDP и прикладными программами должно выполняться исключительно на основе номеров портов. На практике же, прежде чем прикладная программа сможет посыпать UDP-дейтаграммы, она должна обратиться к операционной системе и получить стандартный номер порта для данного протокола и ассоциированный с ним номер порта<sup>1</sup>. После назначения номера порта прикладная программа может отправлять через этот порт пользовательские дейтаграммы. При этом номер выделенного порта будет автоматически занесен в поле *номера порта отправителя* заголовка UDP-дейтаграммы.

---

<sup>1</sup> Пока мы опишем механизм использования портов в общих чертах. В главе 22, “Интерфейс сокетов”, будет приведен пример стандартных вызовов операционной системы, используемых для создания и использования портов.

При обработке поступающих от протокола IP дейтаграмм, протокол UDP демультиплексирует их на основе номера порта получателя, как показано на рис. 12.5.

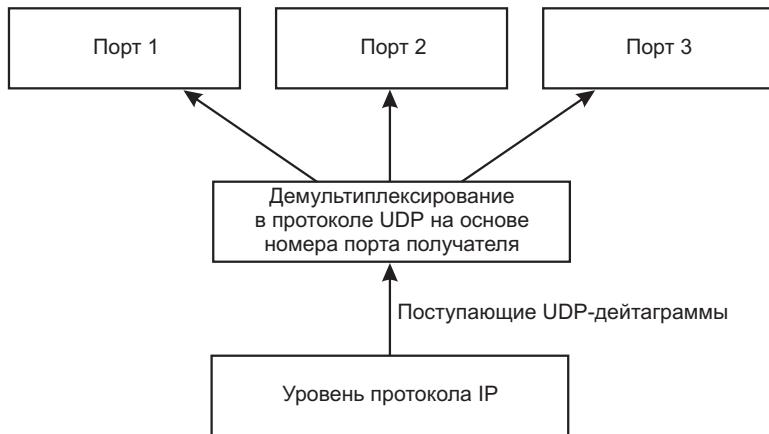


Рис. 12.5. Пример демультиплексирования на следующем после протокола IP уровне. При выборе соответствующего номера порта для поступивших дейтаграмм в программе протокола UDP используется поле номера порта получателя заголовка UDP-дейтаграммы

Проще всего представить UDP-порт в виде очереди. В большинстве реализаций протокола при обращении прикладной программы к операционной системе за выделением порта последняя создает внутреннюю очередь, в которую помещаются прибывающие в порт сообщения. Часто прикладной программе разрешается указывать операционной системе размер очереди сообщений или изменять его в процессе работы. При поступлении дейтаграммы в модуль протокола UDP проверяется номер порта получателя. Программа проверяет, был ли выделен указанный в заголовке UDP-дейтаграммы номер порта одной из запущенных прикладных программ. Если указанный номер порта не найден, программа протокола UDP посыпает отправителю дейтаграммы ICMP-сообщение о недоступности указанного порта (*port unreachable*), после чего аннулирует дейтаграмму. Если номер порта найден, прибывшая дейтаграмма помещается в ассоциированную с ним очередь, откуда ее сможет прочитать прикладная программа. В случае переполнения очереди сообщений отправляется соответствующее сообщение об ошибке, и дейтаграмма аннулируется.

## 12.9. Зарезервированные и свободные номера портов UDP

Теперь пришло время поговорить о назначении номеров портов. Эта проблема крайне важна, поскольку, прежде чем два компьютера смогут взаимодействовать между собой, они должны согласовать номера используемых портов. Например, когда компьютер *A* хочет переписать файл с компьютера *B*, ему нужно знать, какой номер порта назначен программе передачи файлов, запущенной на компьютере *B*. При распределении номеров портов возможны два подхода. Первый состоит в использовании центрального органа управления. Все полномочия по назначению номеров портов передаются этому органу, который по запросу выделяет новый номер и публикует список всех назначенных номеров портов. Программное обеспечение разрабатывается с учетом этого списка портов. Такой подход иногда называют методом *универсального назначения* номеров портов. При

этом сами порты, номера которым назначены центральным органом управления, называют *стандартными*, или *широко известными* (*well-known*).

Второй подход состоит в динамической привязке номеров портов. При этом не существует никакого списка стандартных назначений. Когда прикладной программе понадобится порт для передачи данных, она обращается с соответствующим запросом к операционной системе, которая выделяет его из списка свободных номеров. Для того чтобы узнать, какой номер порта назначен некоторой программе, запущенной на другом компьютере, необходимо отправить соответствующий запрос на этот компьютер. То есть на компьютер отправляется запрос о номере порта, используемого программой передачи файлов. В ответ компьютер присыпает отправителю текущий номер порта, который используется указанной программой.

**Таблица 12.1. Номера портов, используемые в протоколе UDP**

<i>Номер порта</i>	<i>Идентификатор в стандарте</i>	<i>Идентификатор в UNIX</i>	<i>Описание</i>
0	-	-	Зарезервировано
7	ECHO	echo	Эхо
9	DISCARD	discard	Аннулирование
11	USERS	systat	Список активных пользователей
13	DAYTIME	daytime	Текущая дата
15	-	netstat	Программа выдачи состояния сети
17	QUOTE	qotd	Цитата дня
19	CHARGEN	chargen	Генератор символов
37	TIME	time	Текущее время
42	NAMESERVER	name	Сервер имен хостов
43	NICNAME	whois	Информация о пользователе
53	DOMAIN	nameserver	Сервер доменных имен (DNS)
67	BOOTPS	bootps	Сервер BOOTP или DHCP
68	BOOTPC	bootpc	Клиент BOOTP или DHCP
69	TFTP	tftp	Простейший протокол передачи файлов (TFTP)
88	KERBEROS	kerberos	Служба аутентификации Kerberos
111	SUNRPC	sunrpc	Дистанционный вызов процедур в системе Sun
123	NTP	ntp	Протокол синхронизации сетевого времени (NTP)
161	-	snmp	Простой протокол сетевого управления (SNMP)
162	-	snmp-trap	Прерывания протокола SNMP
512	-	biff	Сервер UNIX comsat
513	-	who	Демон UNIX rwho
514	-	syslog	Системный журнал
525	-	timed	Демон службы времени timed

Создатели протокола TCP/IP применили смешанный подход, при котором части портам заранее назначаются стандартные номера. Все остальные номера портов остаются свободными и могут использоваться прикладными программами в рамках локальных систем. Назначение стандартных номеров портов происходило по принципу “снизу вверх”, т.е. от меньших номеров к большим. При этом большие номера зарезервированы для динамического выделения. Список некоторых номеров портов, используемых в настоящее время программами протокола UDP, приведен в табл. 12.1. Во втором столбце указаны идентификаторы портов, используемые в принятых для Internet стандартах. В третьем столбце приведены идентификаторы, используемые в большинстве систем UNIX. Учтите, что данный список далеко не полон и приводится лишь в познавательных целях. Если в других транспортных протоколах реализованы функции, аналогичные протоколу UDP, то по возможности в них используются такие же номера портов.

## 12.10. Резюме

Большинство операционных систем позволяет одновременно запустить несколько прикладных программ. На языке операционной системы выполняющиеся программы называются *процессами*. В протоколе передачи пользовательских дейтаграмм (User Datagram Protocol, или UDP) предусмотрен механизм, позволяющий опознавать поступающие дейтаграммы и передавать их на обработку одному из процессов, запущенных на машине конечного получателя. Для этого в заголовок UDP-дейтаграммы помещается два 16-битовых целых числа, называемых номерами портов отправителя и получателя. Номера портов служат для идентификации отправителя и получателя дейтаграмм. Часть номеров портов (их называют *стандартными*, или *широко известными*) выделена для использования стандартными службами глобальной сети Internet. Например, номер порта 69 зарезервирован для использования службой простейшего протокола передачи файлов, TFTP, о которой речь пойдет в главе 26, “Приложения: передача файлов и удаленный доступ к ним (FTP, TFTP, NFS)”. Остальные номера портов могут свободно использоваться в прикладных программах.

Протокол UDP является примитивным протоколом в том смысле, что по сравнению с протоколом IP разработчики добавили в него совсем немного возможностей. Единственное его назначение — предоставить прикладным программам возможность взаимодействия с помощью ненадежной и не требующей соединения службы доставки пакетов протокола IP. Поэтому UDP-сообщения могут быть потеряны, продублированы, задержаны или доставлены получателю в произвольном порядке. Все эти проблемы должны самостоятельно решаться прикладными программами. Как следствие, очень многие программы, использующие протокол UDP, не всегда корректно работают в объединенной сети, поскольку их создатели не всегда предусматривают обработку возможных ошибочных ситуаций.

В абстрактной многоуровневой модели протокол UDP находится на транспортном уровне, т.е. занимает промежуточный уровень между протоколом IP и прикладными программами. Теоретически протокол транспортного уровня не должен зависеть от протокола IP (т.е. протокола более низкого уровня). Однако на практике между ними существует очень тесная связь. Для вычисления контрольной суммы UDP-сообщения нужно знать IP-адреса отправителя и конечного получателя. А это означает, что перед отправкой дейтаграммы программы протокола UDP должны обратиться к программам протокола IP и выяснить эти адреса.

## Материал для дальнейшего изучения

В книге Таненбаума (Tanenbaum) [127] приведены сравнительные характеристики двух моделей сетевых коммуникаций: на основе передачи дейтаграмм и создания виртуального канала. В статье Болла (Ball) и др. [5] описаны системы, управляемые при помощи механизма сообщений, но не рассмотрен сам протокол передачи сообщений. Протокол UDP, описанный в этой главе, является одним из стандартов семейства протоколов TCP/IP и впервые был предложен Постелем в [RFC 768].

## Упражнения

- 12.1. Поэкспериментируйте с протоколом UDP в вашей локальной сети. Измерьте среднюю скорость передачи сообщений длиной 256, 512, 1024, 2048, 4096 и 8192 байт. Можете ли вы объяснить результаты изменений? (*Подсказка.* Определите значение MTU в вашей локальной сети).
- 12.2. Почему контрольная сумма UDP-сообщения вычисляется независимо от контрольной суммы протокола IP? Назовите недостатки протокола, в котором вычислялось бы общее значение контрольной суммы для всей IP-дейтаграммы, включая UDP-сообщение?
- 12.3. Известно, что передача дейтаграмм без вычисления контрольных сумм сопряжена с определенным риском. При каких условиях одиночный ARP-пакет, содержимое которого искажено в результате передачи по каналам связи, посланный в широковещательном режиме машиной  $P$ , может никогда не достичь машины  $Q$ .
- 12.4. Должна ли в протоколе IP реализовываться идея доставки дейтаграмм нескольким получателям, идентифицируемым по номеру порта? Поясните свой ответ.
- 12.5. *Регистрация имен.* Предположим нужно сделать так, чтобы любые две прикладные программы могли взаимодействовать между собой посредством протокола UDP, однако крайне нежелательно назначать им фиксированные номера портов. Возможные получатели должны идентифицироваться с помощью текстовой строки, длина которой не может превышать 64 символа. Другими словами прикладная программа, запущенная на машине  $A$ , может взаимодействовать с программой “здесь-должно-быть-указано-ее-имя”, запущенной на машине  $B$ . (Предполагается, что процессу-отправителю всегда известен IP-адрес машины, на котором запущен процесс-получатель.) Кроме того, процесс, запущенный на машине  $B$ , может свободно взаимодействовать с программой “некоторое-имя”, выполняющейся на машине  $A$ . Покажите, что для взаимодействия между программами необходимо назначить только один UDP-порт. При этом программное обеспечение, выполняющееся на каждой из машин, должно позволять:
  - а) локальному процессу выбирать неиспользуемый UDP-порт, через который будет осуществляться взаимодействие;
  - б) локальному процессу регистрировать 64-символьное имя, на которое он должен отвечать;
  - в) удаленному процессу устанавливать соединение с локальным процессом посредством протокола UDP, используя для этого только 64-символьное имя и IP-адрес машины получателя.

- 12.6.** Реализуйте программу регистрации имен, о которой шла речь в предыдущем упражнении.
- 12.7.** Назовите преимущества и недостатки использования заранее назначенных номеров UDP-портов.
- 12.8.** Почему при выборе программы-получателя на заданной машине предпочтительней использовать номера портов вместо идентификаторов процессов?
- 12.9.** Поскольку в протоколе UDP не гарантируется доставка дейтаграмм, процесс взаимодействия между двумя приложениями нельзя назвать надежным. Опишите работу надежного протокола, в котором для гарантирования доставки отслеживаются временные интервалы и используются подтверждения. Насколько при этом увеличится поток данных в сети и возрастут задержки при передаче дейтаграмм?
- 12.10.** Отправьте по глобальной сети с помощью протокола UDP поток дейтаграмм и определите, какое их количество (в процентах) будет потеряно и переупорядочено. Будет ли результат зависеть от времени суток и загрузки сети?

# 13

## Надежная потоковая транспортная служба (TCP)

### 13.1. Введение

В предыдущей главе была описана ненадежная служба доставки пакетов, не требующая установки соединения с получателем, которая лежит в основе процесса взаимодействия по объединенной сети. Кроме того, вы уже познакомились с протоколом IP, посредством которого IP-дейтаграммы доставляются на машину конечного получателя. В этой главе речь пойдет о еще одной важной и популярной сетевой службе — надежной потоковой транспортной службе — и лежащем в ее основе протоколе управления передачей (*Transmission Control Protocol*, или *TCP*). Ниже мы покажем, что по сравнению с описанными ранее протоколами протокол TCP обладает обширными функциональными возможностями. Однако следует заметить, что и реализовать его значительно сложнее.

Хотя выше мы уже говорили о том, что протокол TCP является частью семейства протоколов TCP/IP, используемого в Internet, он по сути является независимым протоколом общего назначения, который можно адаптировать для использования практически с любыми средствами доставки. В частности, поскольку протокол TCP не зависит от применяемого сетевого оборудования, его можно использовать как в пределах одной небольшой локальной сети, наподобие Ethernet, так и в сложной объединенной сети. Более того, протокол TCP стал настолько популярным, что Международная организация по стандартизации (ISO) создала на его основе свой протокол TP-4.

### 13.2. Необходимость в потоковой доставке дейтаграмм

На самом нижнем уровне компьютерные сетевые коммуникационные системы обеспечивают лишь ненадежную доставку пакетов. Это означает, что при передаче по сети пакеты могут быть утеряны, а их содержимое искажено в результате возникновения перекрестных помех, отказов сетевого оборудования или его перегрузки. В сетях, где выполняется динамическая маршрутизация пакетов, возможно нарушение порядка их доставки, возникновение значительных задержек или дубликатов. Более того, от используемого сетевого оборудования может зависеть оптимальный размер пакета, а также другие ограничения, налагаемые для достижения максимальной пропускной способности канала передачи данных.

В тоже время прикладным программам, находящимся на верхнем уровне сетевой иерархии, часто требуется передавать большие массивы данных от одного компьютера к другому. С точки зрения программирования использование для

этих целей низкоуровневых ненадежных сетевых протоколов, не требующих установки соединения с получателем, является довольно скучным, однообразным и раздражающим занятием. Причина заключается в том, что в каждой прикладной программе должен быть предусмотрен код для обнаружения и коррекции ошибок, возникающих в процессе передачи данных. Создать такой код, понять как он функционирует, а тем более внести изменения в корректно работающую программу не так-то просто, поэтому для выполнения подобной работы требуются высококвалифицированные программисты, обладающие необходимым багажом знаний. Вследствие этого одной из целей, ставившихся при разработке семейства сетевых протоколов, был поиск общего метода решения проблемы надежности доставки больших потоков данных. Экспертам поручили создать единую универсальную программу потокового сетевого протокола, которая могла бы использоваться всеми прикладными программами. При таком подходе удалось добиться независимости прикладных программ от используемых сетевых технологий и стандартизовать единый интерфейс для служб потоковой передачи данных.

### 13.3. Особенности надежной службы доставки пакетов

Интерфейс между прикладными программами и надежной службой доставки пакетов протокола TCP/IP характеризуется пятью основными особенностями.

- *Обработка потоков данных.* При пересылке между двумя прикладными программами (пользовательскими процессами) большого массива данных он рассматривается как поток битов, разделенных на октеты, состоящие из 8 битов. Октеты часто по традиции называют *байтами*. Служба потоковой доставки данных, запущенная на машине получателя, должна передать прикладной программе такую же последовательность октетов, которая была передана этой же службе на машине отправителя.
- *Подключение по виртуальному каналу.* Передачу потока данных можно сравнить с обычным телефонным звонком. Перед началом обмена данными отправляющая и принимающая программы должны, вступив во взаимодействие со своими локальными операционными системами, проинформировать их о том, что необходимо запустить процесс потоковой передачи данных. Далее, как и в телефонии, одна из программ “звонит” другой программе, которая должна этот звонок принять. Эти действия выполняют специальные модули сетевого протокола двух операционных систем. Для того чтобы проверить возможность передачи данных и готовность сторон, эти модули обмениваются по объединенной сети специальными сообщениями. Выполнив все формальности, модули протокола информируют свои прикладные программы о том, что *соединение* с получателем (отправителем) установлено и можно начинать передачу данных. Во время передачи данных модули сетевого протокола, работающие на компьютерах, продолжают взаимодействовать, проверяя правильность переданных данных. Если передача данных по какой-либо причине завершается неудачей (например, вышло из строя сетевое оборудование одного из компьютеров, находящегося на пути следования пакетов от отправителя до конечного получателя), это сразу же становится известно вовлеченым в процесс сторонам (точнее, прикладным программам, выполняющим передачу данных). Далее для описания подобного типа соединения мы будем использовать термин *виртуальный канал* (*virtual circuit*), поскольку с точки зрения прикладных программ оно функционирует наподобие физического выделенного канала связи. Таким образом, при использовании потоковой

службы доставки пакетов для прикладных программ создается иллюзия абсолютно надежной передачи данных.

- **Буферизованная передача.** При передаче потоков данных по виртуальным каналам прикладные программы периодически посылают нижележащему модулю протокола октеты данных. При этом количество октетов данных может быть произвольным — в зависимости от типа используемых прикладных программ (вплоть до одного байта!). На приемном конце модуль протокола доставляет прикладной программе октеты данных из полученного потока в том же порядке, как они были посланы. При этом данные становятся доступными прикладным программам сразу после их получения и проверки на правильность. Отметим, что модулю протокола ничто не мешает разделить поток данных на пакеты, не привязываясь к фрагментам, полученным от прикладных программ. Обычно во время реализации стараются повысить эффективность передачи данных и минимизировать ненужный сетевой трафик. Поэтому сначала модуль протокола собирает необходимое количество данных для формирования дейтаграммы достаточно большого размера и только после этого передает ее по объединенной сети. Таким образом, даже если прикладная программа генерирует поток данных, состоящий из последовательности одиночных октетов, это никак не скажется на эффективности передачи по объединенной сети. Точно так же, если прикладная программа генерирует блоки данных слишком большого размера, перед передачей модуль протокола разобьет их на меньшие части.

Если генерируемые приложением данные должны отправляться сразу (без задержки, связанной с заполнением буфера), в службе потоковой доставки предусмотрен специальный механизм *выталкивания*, немедленно запускающий процесс передачи данных. На передающей стороне это вызовет немедленную передачу накопленных в буфере данных, даже если он еще не полон. На машине получателя механизм выталкивания также заставляет модуль протокола TCP доставить поступившие данные прикладной программе без задержки. Однако следует заметить, что использование механизма выталкивания гарантирует только немедленную отправку всех накопленных данных. При этом первоначальная структура пакетов не сохраняется. Другими словами, после того как механизм выталкивания приведен в действие, модуль протокола может разделить поток данных самым неожиданным образом.

- **Неструктурированные потоки.** Следует отметить, что потоковая служба доставки протокола TCP/IP не поддерживает обработку структурированных потоков данных. Например, не существует средств, с помощью которых приложение баз данных смогло бы заставить потоковые службы доставки каким-либо образом отслеживать границы записей. Нельзя также идентифицировать содержимое потока на принадлежность к какой-либо прикладной программе. Поэтому при использовании потоковой службы доставки прикладные программы должны сами определять содержимое потока и заранее согласовывать форматы передаваемых данных.
- **Дуплексное соединение.** При установке соединения между потоковыми службами протокола TCP/IP отправителя и получателя, возможна параллельная передача данных в обеих направлениях. Подобное соединение называют *дуплексным (full duplex)*. Прикладная программа воспринимает дуплексное соединение как два независимых потока данных, текущих в противоположных направлениях. При этом между потоками отсутствует какое-либо взаимодействие. В потоковой службе доставки

предусмотрен также *полудуплексный режим* (*half duplex*), когда данные прикладных программ могут передаваться только в одном направлении в некоторый момент времени. Преимуществом дуплексного режима является то, что модуль протокола может передавать управляющую информацию о пересылаемом потоке данных отправителю (т.е. в обратном направлении). Подобное совмещение передаваемых потоков позволяет уменьшить сетевой трафик.

### 13.4. Обеспечение надежности

Выше уже отмечалось, что надежная потоковая служба гарантирует доставку потока данных от одной машины к другой без потери или дублирования данных. При этом у вас может возникнуть вопрос: как модулю протокола удается достичь надежной передачи данных, если все задействованные при этом протоколы нижнего уровня обеспечивают только ненадежную доставку пакетов? Ответить на этот вопрос очень сложно, однако в двух словах можно сказать, что в большинстве протоколов для обеспечения надежности используется технология *подтверждения приема с повторной передачей* (*positive acknowledgement with retransmission*). Суть ее заключается в том, что после приема данных, получатель должен связаться с отправителем и отправить ему специальное сообщение о *подтверждении приема* (*acknowledgement*, или *ACK*). Отправитель сохраняет информацию о каждом посланном пакете. При этом следующий пакет посыпается только после получения подтверждения о приеме предыдущего пакета. Кроме того, посылая пакет, отправитель устанавливает значение таймера, по истечении которого выполняется *повторная передача* (*retransmit*) пакета, если не будет получено подтверждение о его успешном приеме. Упрощенная схема передачи данных с помощью механизма подтверждения приема с повторной передачей приведена на рис. 13.1. События, происходящие на стороне отправителя и получателя, изображены, соответственно, слева и справа. Диагональные линии в центре рисунка соответствуют передаче одного сообщения по сети.

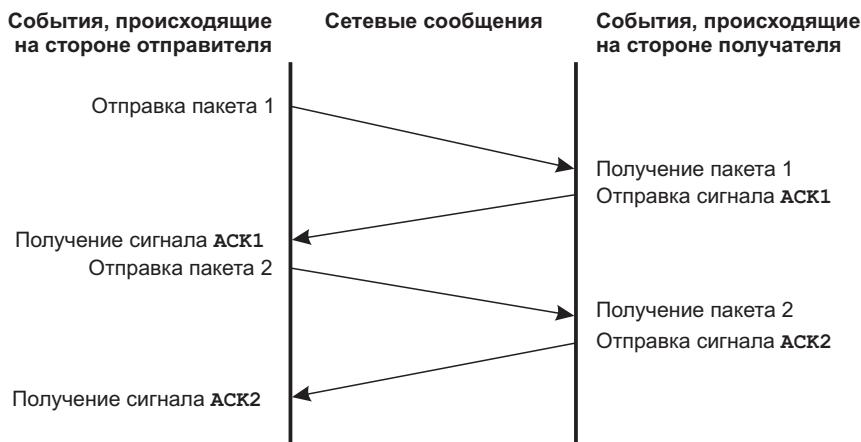


Рис. 13.1. Иллюстрация механизма подтверждения приема с повторной передачей, при котором отправитель ждет уведомления об успешном получении каждого пакета. Величина временной задержки отложена на этой схеме по вертикали, а диагональные линии в центре рисунка представляют процесс передачи пакета по сети

На рис. 13.2 показано, что произойдет, если в процессе передачи пакет будет утерян или поврежден. Передав пакет, отправитель устанавливает значение таймера, по истечении которого (если не подтвержден прием) считается, что пакет не достиг получателя, и выполняется его повторная передача.

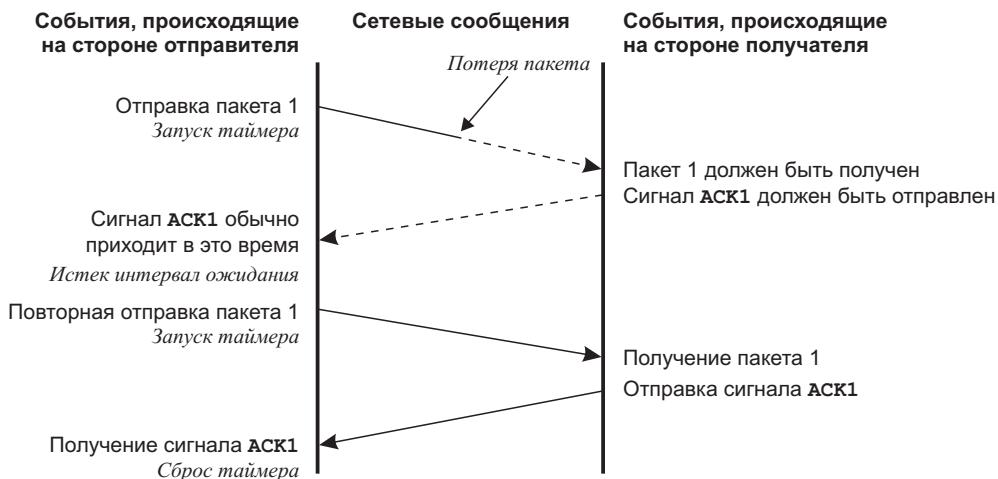


Рис. 13.2. При потере пакета через определенный интервал времени выполняется его повторная передача. Пунктирной линией показан процесс нормальной передачи пакета и получения подтверждения

Еще одна проблема обеспечения надежности возникает при получении системой доставки нижнего уровня дублей пакетов. Одна из причин появления дублей — перегрузка сети, в результате которой увеличивается время доставки пакетов и, как следствие, преждевременно выполняется повторная передача. Решение этой проблемы требует тщательно продуманного подхода, поскольку продублированы могут быть как сами пакеты, так и сигналы об их получении. Обычно для обнаружения дубликатов в надежных протоколах доставки используется нумерация пакетов. Каждому посылаемому пакету присваивается порядковый номер. При этом получатель должен запоминать порядковые номера всех полученных пакетов. Чтобы избежать путаницы, вызванной дубликатами сигналов о получении пакетов, в механизме подтверждения приема с повторной передачей порядковые номера посылаются обратно отправителю вместе с сигналом об успешном получении пакета. В результате отправитель сможет корректно сопоставить сигнал о получении пакета с самим пакетом.

### 13.5. Использование движущихся окон

Прежде чем приступить к рассмотрению потоковой службы доставки протокола TCP, необходимо познакомиться еще с одним методом, используемым в процессе потоковой передачи данных. Речь идет о *движущихся (скользящих) окнах (sliding window)*, позволяющих повысить эффективность потоковой передачи данных. Чтобы понять принцип их работы, необходимо проанализировать последовательность событий, происходящих при надежной передаче пакета (рис. 13.1). Передав пакет, отправитель ждет подтверждения его получения, а затем передает следующий пакет. Как показано на рис. 13.1 в определенный момент поток данных между машинами течет только в одном направлении, и это при том, что в сети возможна одновременная передача потоков данных в

обоих направлениях! Очевидно, что при возникновении задержек между отправкой пакетов (например, когда компьютеры вычисляют маршруты следования пакетов или контрольные суммы), сеть будет загружена не полностью. А если представить себе сеть, в которой возникают большие задержки при передаче пакетов, проблема становится понятной.

*В простейшем случае использование технологии подтверждения приема с повторной передачей приводит к существенному снижению пропускной способности сети, поскольку отправка нового пакета откладывается до момента получения подтверждения о приеме предыдущего пакета.*

Метод движущихся окон является более сложным вариантом технологии подтверждения приема с повторной передачей, по сравнению с описанным выше упрощенным вариантом. Она позволяет эффективно использовать полосу пропускания сети, поскольку отправитель может послать несколько пакетов сразу, не дожидаясь подтверждения приема каждого пакета. Проще всего описать работу метода движущихся окон на примере последовательности передаваемых пакетов, изображенных на рис. 13.3. Суть метода состоит в том, что небольшое окно фиксированного размера накладывается на последовательность предназначенных для передачи пакетов. При этом передаются все пакеты, попавшие внутрь окна.



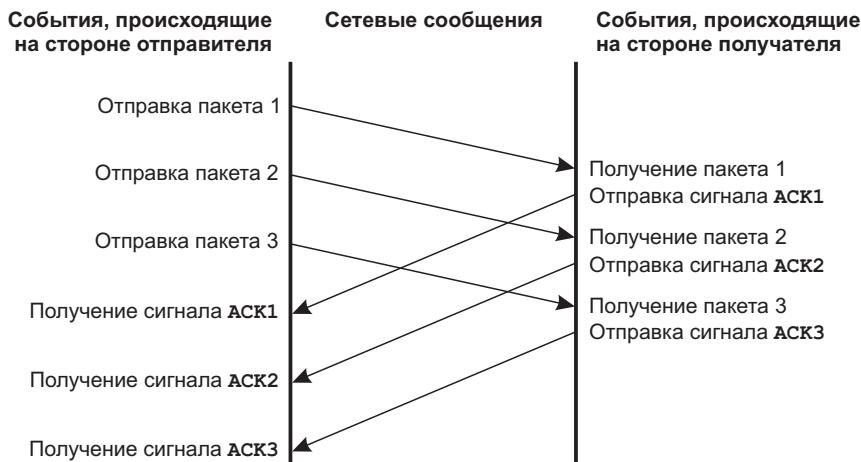
Рис. 13.3. Движущееся окно, внутрь которого помещено 8 пакетов (а); при получении подтверждения о приеме пакета 1 окно сдвигается на один пакет вправо, и в сеть отправляется 9-й пакет (б). Если для какого-либо пакета, попавшего в окно, не получен сигнал подтверждения, то выполняется повторная передача этого пакета

Пакет считается *непринятым (unacknowledged)*, если он был послан в сеть, но не было получено подтверждение о его приеме. Формально количество непринятых пакетов в любой момент времени не может превышать *размеров окна* и всегда является небольшим целым числом. Например, если размер движущегося окна равен 8, отправитель может отправить в сеть до 8 пакетов и только после этого перейти в состояние ожидания подтверждений об их приеме.

Как показано на рис. 13.3, получив подтверждение о приеме первого из пакетов, находящихся внутри окна, отправитель сдвигает окно на один пакет вправо и посыпает в сеть следующий пакет. По мере получения новых подтверждений окно все время передвигается вправо.

Производительность метода движущихся окон зависит как от размера самого окна, так и от скорости пересылки пакетов по сети и задержек, возникающих

при их обработке. На рис. 13.4 этот метод продемонстрирован на примере окна, в которое помещается три пакета. Обратите внимание, что отправитель успевает послать все три пакета в сеть до того, как будет получено хотя бы одно сообщение о подтверждении их приема.



*Рис. 13.4. Пересылка трех пакетов с использованием метода движущихся окон. Идея заключается в том, что отправитель может послать в сеть сразу все три пакета, не дожидаясь сообщений о подтверждении их приема*

Если принять размер окна равным единице, то метод движущихся окон ничем не будет отличаться от рассмотренного в предыдущем разделе метода подтверждения приема с повторной передачей. Увеличивая размер окна, можно полностью исключить простой в сети. Это произойдет при достижении равновесия между количеством посланных в сеть пакетов, скоростью их передачи и задержками, возникающими при их обработке на узлах сети.

*При использовании метода движущихся окон правильный выбор размера окна позволяет полностью исключить простой в сети, достичь большей производительности по сравнению с методом подтверждения приема с повторной передачей*

При использовании метода движущихся окон в модуле протокола необходимо отслеживать сигналы подтверждения приема для каждого находящегося в окне пакета. А это означает, что для каждого непринятого пакета должен быть установлен отдельный таймер. Тогда по исчерпании значения соответствующего таймера легко определить, какой из пакетов утерян, и передать его повторно. Получив подтверждение приема, отправитель должен переместить окно и вытеснить из него все принятые пакеты, идущие подряд. В модуле протокола на машине получателя создается аналогичное окно. Это позволяет получателю отслеживать принятые пакеты и отправлять подтверждения об их приеме. Таким образом, окно разделяет последовательность пакетов на три части. Те, что находятся слева от окна, уже переданы в сеть и успешно достигли получателя. Пакеты, внутри окна передаются, а пакеты, находящиеся справа от окна, ожидают отправки. Самый левый из находящихся в окне пакетов имеет наименьший порядковый номер и является первым в списке непринятых пакетов.

## 13.6. Протокол управления передачей (TCP)

Рассмотрев принцип работы движущихся окон, можно переходить к изучению надежной потоковой службы, являющейся составной частью семейства протоколов TCP/IP. Эта служба называется *протоколом управления передачей* (*Transmission Control Protocol*, или *TCP*). Она настолько важна, что все семейство протоколов назвали в ее честь — TCP/IP. Однако следует помнить, что

*TCP является коммуникационным протоколом, а не просто частью программы.*

Разница между протоколом и программой примерно такая же, как между описанием языка программирования и его реализацией (т.е. компилятором). Как и при разработке компиляторов, иногда трудно провести грань между спецификацией протокола и его реализацией. Пользователям чаще всего приходится сталкиваться именно с реализацией протокола TCP, а не с его спецификацией. Поэтому неудивительно, что конкретную реализацию протокола TCP часто путают со стандартом. Тем не менее необходимо четко различать эти две вещи.

Итак, какие же функции выполняет протокол TCP? Их так много и протокол так сложен, что ответить на этот, казалось бы, простой вопрос не так-то просто. В спецификации протокола определен точный формат данных и подтверждающих сообщений, которыми обмениваются два компьютера при организации надежного канала передачи данных. Кроме того, в спецификации протокола описаны процедуры, которые должны выполнять машины в сети для обеспечения корректной доставки данных. В стандарте протокола TCP оговорено, как различать получателей пакетов на конкретной машине, а также указаны способы восстановления данных взаимодействующими сторонами при возникновении различного рода ошибок, таких как потеря или дублирование пакетов. Кроме того в стандарте протокола TCP указаны способы установки и завершения сеансов потоковой передачи данных.

Несколько слов о том, что не включено в стандарт протокола. Хотя в спецификации протокола описаны методы использования протокола TCP прикладными программами, в ней не оговорены детали реализации конкретного интерфейса между прикладной программой и модулем протокола TCP. Другими словами, в стандарте протокола зафиксированы только действия, выполняемые конкретной реализацией модуля TCP, но не описаны процедуры, которые должна вызывать прикладная программа для выполнения этих действий. Описание прикладного интерфейса не включено в стандарт протокола TCP по одной причине — стандарт должен быть универсальным. В частности, поскольку разработчики обычно встраивают модуль реализации протокола TCP в операционную систему, им нужно поддерживать все виды интерфейсов, предоставляемые системой. Кроме того, у разработчиков появляется возможность создавать программное обеспечение, выполняющее одинаковые функции согласно спецификации протокола TCP, для различных компьютерных платформ.

Поскольку протокол TCP не привязан к системе передачи данных, он может использоваться совместно с различными системами доставки пакетов (протоколами транспортного уровня), включая широко используемую службу доставки дейтаграмм протокола IP. Например, реализации протокола TCP могут быть созданы для связи по коммутируемым телефонным каналам, локальным сетям, высокоскоростным оптоволоконным каналам или низкоскоростным сетям большой протяженности. Фактически, одним из самых больших преимуществ протокола TCP является возможность его совместной работы практически с любым средством доставки пакетов.

## 13.7. Порты, соединения и конечные точки

Как и протокол передачи пользовательских дейтаграмм (UDP), описанный в главе 12, “Передача пользовательских дейтаграмм (UDP)”, протокол TCP располагается в системе иерархии выше уровня протокола IP. Абстрактная многоуровневая модель протоколов семейства TCP/IP приведена на рис. 13.5. Протокол TCP позволяет нескольким, выполняющимся на компьютере прикладным программам параллельно и независимо обмениваться данными с прикладными программами, запущенными на других машинах. Так же, как и UDP, протокол TCP демультиплексирует свой входной трафик между несколькими прикладными программами. Поэтому по аналогии с протоколом UDP, в протоколе TCP также используется понятие *номеров портов*, позволяющих идентифицировать получателя на конкретной машине. Каждому из портов для идентификации присваивается небольшое целое число<sup>1</sup>.

Абстрактная многоуровневая модель



Рис. 13.5. Абстрактная многоуровневая модель протоколов семейства TCP/IP. Из рисунка видно, что оба протокола (как UDP, так и TCP) расположены выше уровня протокола IP. При этом протокол TCP обеспечивает надежную потоковую службу доставки пакетов, тогда как UDP выполняет только ненадежную доставку пакетов получателю. Несмотря на это оба протокола успешно используются в прикладных программах

При описании портов протокола UDP мы говорили о том, что каждый порт можно представить в виде очереди, в которую сетевое программное обеспечение помещает все полученные дейтаграммы. Порты протокола TCP — намного более сложное понятие, поскольку номеру порта нельзя поставить в соответствие какой-либо один объект. Поэтому в протокол TCP введено абстрактное понятие *соединения* (*connection abstraction*), в котором идентифицируемыми объектами являются соединения, осуществляемые по виртуальным каналам, а не отдельные порты. Таким образом, ключевой в протоколе TCP является идея использования соединений. Понимание этой идеи поможет вам понять назначение и принцип использования номеров портов протокола TCP.

<sup>1</sup> Хотя и в протоколе TCP и в протоколе UDP для идентификации портов используются целые числа начиная с 1, никакого конфликта по номерам портов двух протоколов не происходит. Причина состоит в том, что во входящей дейтаграмме указывается как используемый тип протокола, так и номер порта, относящийся к этому протоколу.

*В качестве основной абстракции в протоколе TCP используется понятие соединения, а не номера порта. Соединения идентифицируются с помощью пары конечных точек.*

Осталось ответить на вопрос, что же такое конечные точки соединения? Как уже было сказано, соединением называется виртуальный канал, проложенный между двумя прикладными программами. Поэтому вполне естественно предположить, что конечными точками соединения являются сами прикладные программы. Однако это не так. В протоколе TCP *конечная точка* определяется с помощью 2-мерного кортежа, или пары целых чисел (*узел, порт*), где параметр *узел* определяет IP-адрес узла сети, а параметр *порт* является номером TCP-порта на этом узле. Например, запись (128.10.2.3, 25) определяет в качестве конечной точки TCP-порт с номером 25 на машине, IP-адрес которой 128.10.2.3.

Теперь, после определения понятия конечной точки, несложно понять, что такое само соединение. Напомним, что соединение определяется парой своих конечных точек. Поэтому если установлено соединение между машиной с IP-адресом 18.26.0.36, расположенной в Массачусетском технологическом институте, и машиной с IP-адресом 128.10.2.3, расположенной в университете Пердью, оно идентифицируется парой конечных точек:

(18.26.0.36, 1069) и (128.10.2.3, 25).

Если в это же время к тому же самому компьютеру, находящемуся в университете Пердью, поступит запрос на установку соединения от другой машины с IP-адресом 128.9.0.32, расположенной в институте информационных исследований (Information Sciences Institute), то соединение будет идентифицироваться двумя конечными точками следующим образом:

(128.9.0.32, 1184) и (128.10.2.3, 53).

Приведенные примеры соединений были довольно простыми, поскольку номера портов, используемые во всех конечных точках, имели уникальные значения. Однако в концепции абстрактного соединения допускается совместное использование одной конечной точки несколькими соединениями. Например, можно добавить к двум рассмотренным выше соединениям с машиной университета Пердью еще одно, установленное с машиной 128.2.254.139, находящейся в университете Карнеги-Меллона:

(128.2.254.139, 1184) и (128.10.2.3, 53).

На первый взгляд может показаться странным, что два соединения могут одновременно использовать TCP-порт 53 на машине 128.10.2.3, однако никакой неопределенности при этом не возникает. Все дело в том, что в протоколе TCP все входящие сообщения привязываются не к заданному номеру порта протокола, а к открытому соединению, которое в свою очередь идентифицируется двумя конечными точками. Итак, необходимо запомнить следующее.

*Поскольку в протоколе TCP соединение идентифицируется по двум конечным точкам, один номер TCP-порта может использоваться несколькими соединениями, открытыми на одной и той же машине.*

Концепция абстрактного соединения играет очень важную роль с точки зрения программирования. Благодаря ей программист может создать программу, обеспечивающую одни и те же функциональные возможности для нескольких одновременно открытых соединений. При этом нет необходимости использовать для каждого соединения уникальный номер локального порта. Например, в большинстве операционных систем поддерживается возможность одновременного доступа нескольким клиентам к службе электронной почты. Это позволяет некоторым

компьютерам параллельно отправлять почтовые сообщения. Поскольку программа, принимающая входящие сообщения, использует для связи протокол TCP, для ее работы достаточно выделить только один локальный порт, несмотря на то, что она может параллельно обрабатывать несколько соединений.

### 13.8. Пассивное и активное открытие соединения

В отличие от UDP, протокол TCP требует установки соединения с получателем. А это означает, что обе конечные точки должны предварительно дать согласие на участие в соединении. Другими словами, прежде чем поток данных протокола TCP сможет проследовать по объединенной сети, прикладные программы, расположенные на концах соединения, должны разрешить установку соединения. Для этого одна из прикладных программ вызывает функцию *пассивного открытия соединения* операционной системы. Вызов такой функции является сигналом о том, что данная сторона готова к приему входящих соединений. После вызова описываемой функции операционная система назначает номер TCP-порта для своей конечной точки данного соединения. Для установки соединения, прикладная программа, находящаяся на другом конце виртуального канала должна обратиться к операционной системе с запросом на *активное открытие соединения*. При этом два модуля протокола TCP, запущенные на подключающихся компьютерах, взаимодействуют между собой и проверяют возможность установки соединения. После того как соединение установлено, прикладные программы могут обмениваться данными. В процессе работы модули протокола TCP, находящиеся по обе стороны соединения, незаметно для прикладных программ обмениваются служебными сообщениями, гарантирующими надежность доставки данных. Подробнее процесс установки соединения будет рассмотрен чуть ниже после изучения формата сообщения протокола TCP.

### 13.9. Сегменты, потоки и порядковые номера

В протоколе TCP перед передачей по сети поток данных разбивается на *сегменты*, состоящие из последовательности октетов, или байтов. Как правило, каждый сегмент данных передается по объединенной сети в виде одной IP-дейтаграммы.

Для решения двух важных задач — повышения эффективности передачи данных и управления потоком данных — в протоколе TCP применяется специализированный механизм движущихся окон, наподобие того, что был описан выше. Он позволяет модулю протокола TCP отправлять сразу несколько сегментов данных до получения сообщений об их доставке. Подобный подход позволяет увеличить суммарную пропускную способность сети за счет сокращения времени ееостояния. Вариант движущихся окон, реализованный в протоколе TCP, позволяет также решить еще одну проблему — сквозного управления потоком данных. В распоряжение получателя предоставлены средства, позволяющие ограничить поток передаваемых данных в случае исчерпания буферного пространства для приема данных.

Механизм движущихся окон протокола TCP определяет октетами данных, а не сегментами или пакетами. При этом октеты потока данных нумеруются последовательно, а отправитель запоминает три указателя для каждого открытого соединения. Эти указатели определяют движущееся окно, как показано на рис. 13.6. Первый указатель обозначает левую границу движущегося окна. Он разделяет октеты на те, что были успешно доставлены получателю, и те, которые отправлены в сеть, но подтверждение их доставки еще не получено. Второй указатель обозначает правую границу движущегося окна. Он определяет номер

старшего октета последовательности, который может быть послан в сеть до поступления сигналов о подтверждении доставки других октетов, находящихся в окне. Третий указатель обозначает границу внутри окна, разделяющую последовательность октетов на те, что уже были посланы в сеть, и те, которые еще только предстоит отправить. Модуль протокола отправляет в сеть все октеты, находящиеся в окне, без задержки. Поэтому граница внутри окна, определяемая третьим указателем, обычно перемещается слева направо очень быстро.



*Рис. 13.6. Пример движущегося окна протокола TCP. Октеты 1 и 2 были успешно доставлены получателю; октеты 3–6 посланы в сеть, но подтверждение об их доставке еще не получено; октеты 7–9 еще не отправлены, но могут быть отправлены без всяких задержек; октеты с номерами 10 и выше не могут быть посланы в сеть до тех пор, пока не попадут внутрь окна*

Выше было описано, как модуль протокола TCP, работающий на машине отправителя, перемещает окно вдоль последовательности октетов. Кроме того, было отмечено, что модуль протокола TCP, запущенный на машине получателя, должен создать точно такое же окно для того, чтобы собрать поток получаемых данных. Однако не стоит забывать, что устанавливаемые протоколом TCP соединения являются дуплексными. А это означает, что одновременно по каждому из виртуальных каналов можно передавать два потока данных в противоположных направлениях. Таким образом, передачу этих двух потоков можно считать полностью независимыми процессами, поскольку в любой момент времени данные могут течь как в одном направлении, так и в обоих. Поэтому программы реализации протокола TCP, запущенные на обеих концах соединения, должны поддерживать для каждого соединения по два окна (всего окон четыре). Одно окно используется для передачи потока данных, а другое — для приема.

### 13.10. Окна переменного размера и управление потоком данных

Еще одно различие между механизмом движущихся окон протокола TCP и упрощенным механизмом, рассмотренным выше, состоит в том, что размер окна в протоколе TCP может с течением времени изменяться. В каждом сообщении, подтверждающем получение данных, которое присыпает получатель, указывается количество принятых октетов и так называемый *анонс окна* (*window advertisement*). В анонсе указывается количество дополнительных октетов данных, которые сможет принять получатель. Анонс окна можно считать сообщением, присыпаемым получателем потока данных, о текущем размере своего буфера. В ответ на получение анонса окна с увеличенными размерами отправитель соответственно увеличивает размер своего движущегося окна и отправляет получателю дополнительные октеты, не дожидаясь получения сигналов подтверждения приема. Получив анонс окна с уменьшенными размерами, отправитель должен

уменьшить размеры своего движущегося окна и не отправлять октеты, расположенные за границами окна. Модуль протокола TCP не должен сразу реагировать на поступление анонса окна с уменьшенными размерами и сдвигать влево границу окна в потоке данных. Он просто должен дождаться подтверждения приема, не посыпая октеты получателю. Таким образом, через некоторое время размер окна уменьшится автоматически по мере сдвига вправо его левой границы.

Преимущество использования окна переменного размера заключается в том, что, помимо повышения надежности передачи данных, оно обеспечивает возможность управления потоком данных. Чтобы не допустить переполнения буфера, получатель должен по мере его заполнения посылать анонсы окон меньшего размера. В исключительных случаях для прекращения передачи данных получатель анонсирует окно нулевого размера. После того как освободится место в буфере, получатель присылает анонс ненулевого окна для возобновления передачи данных<sup>2</sup>.

Описанный механизм управления потоком данных протокола TCP очень важен для работы реальной объединенной сети, состоящей из большого количества сетей разной протяженности и пропускной способности, соединенных маршрутизаторами различной мощности и скорости. Однако следует знать о двух не связанных между собой проблемах управления потоком. Первая — заключается в том, что в семействе протоколов TCP/IP должен быть предусмотрен механизм сквозного управления потоком данных между отправителем и получателем. Предположим, например, что небольшой персональный компьютер подключается по сети с большой пропускной способностью к мощной суперЭВМ. При этом у персонального компьютера должны быть средства управления поступающим от суперЭВМ потоком данных, иначе очень быстро переполнится входной буфер модуля протокола. Поэтому для гарантирования надежной доставки данных в протоколе TCP должны быть реализованы средства сквозного управления потоком данных. Вторая проблема состоит в том, что в семействе протоколов TCP/IP должен быть реализован механизм управления потоком данных для промежуточных устройств (т.е. маршрутизаторов). Он задействуется, когда промежуточная машина не справляется с потоком данных, поступающим от отправителя.

Если промежуточное устройство не справляется с поступающим потоком данных, такая ситуация называется *перегрузкой (congestion)*. При этом в действие приводятся механизмы *устранения перегрузки (congestion control)*. Для сквозного управления потоком данных в протоколе TCP используется механизм движущихся окон. Однако в нем не предусмотрен отдельный механизм устранения перегрузки. Тем не менее, ниже будет показано, что при хорошо продуманной и тщательной реализации программ протокола TCP можно обнаружить перегрузку на промежуточном устройстве и устраниТЬ ее. В то же время некачественная реализация программ протокола может даже усугубить ситуацию. В частности, хорошо продуманная схема повторной передачи данных может устраниТЬ перегрузку, возникшую на одном из промежуточных устройств, тогда как плохо выбранная схема может привести к еще большей перегрузке устройства.

### 13.11. Формат TCP-сегмента

Единицей передачи данных между двумя машинами в протоколе TCP является *сегмент*. Сегменты используются для установки соединения, передачи данных, отправки сигналов подтверждения приема, анонсирования размера окон и

<sup>2</sup> Существует два исключительных случая при передаче данных, когда получен анонс окна нулевого размера. Во-первых, отправитель имеет право передать сегмент данных, в заголовке которого установлен бит срочности. Это позволяет проинформировать получателя о том, что должны быть переданы срочные данные. Во-вторых, чтобы избежать потенциальной тупиковой ситуации, когда после получения анонса окна нулевого размера будет утерян анонс окна ненулевого размера, отправитель периодически посыпает запросы на возобновление передачи данных.

закрытия соединения. Поскольку протокол TCP является дуплексным (т.е. потоки данных могут течь одновременно в двух направлениях) сигналы подтверждения, посланные от машины A к машине B, могут передаваться в тех же сегментах, что и потоки данных от машины A к машине B. И это несмотря на то, что сигналы подтверждения приема относятся к потокам данных, текущих от машины B к машине A<sup>3</sup>. Формат сегмента протокола TCP приведен на рис. 13.7.



Рис. 13.7. Формат TCP-сегмента, состоящего из заголовка и блока данных. Сегменты используются не только для передачи данных, а также для отправки сигналов подтверждения приема, установки и разрыва соединения

Каждый сегмент состоит из двух частей — заголовка и блока данных. Как и следовало ожидать, в заголовке, называемом также *TCP-заголовком*, находятся идентификационные данные и управляющая информация. В первых двух полях заголовка располагаются *номера TCP-портов отправителя и получателя*, которые идентифицируют прикладные программы по обе стороны соединения. Поле *порядкового номера* идентифицирует положение текущего сегмента в потоке данных отправителя. В поле *номера сигнала подтверждения* указывается номер октета, который отправитель ожидает в дальнейшем получить обратно. Обратите внимание: порядковый номер сегмента относится к потоку данных, текущему в том же направлении, что и сегмент, номер же сигнала подтверждения — потоку данных, текущему в противоположном направлении.

В поле *длины заголовка*<sup>4</sup> указывается целое число, обозначающее длину заголовка сегмента, выраженное в блоках длиной 32 бита. Данное поле включено в заголовок, поскольку поле *параметров протокола* имеет переменную длину, которая зависит от того, какие параметры включены в сегмент. Таким образом, размер TCP-заголовка зависит от того, какие параметры в него включены. Поле, обозначенное на рис. 13.7 как *Резерв* имеет размер 6 битов и зарезервировано для использования в будущих стандартах протокола.

Поскольку в протоколе TCP сегменты используются и для передачи данных, и для передачи сигналов подтверждения их приема, а также для передачи запросов на установку и закрытие соединения, в TCP-заголовок включено специальное 6-битовое поле *кода сегмента*, которое определяет его формат и содержимое. От значения битов этого поля зависит, как будут интерпретироваться другие поля заголовка (табл. 13.1).

<sup>3</sup> На практике дуплексный режим обычно редко используется, поскольку большинство приложений, как правило, не посыпают данные одновременно в двух направлениях.

<sup>4</sup> В спецификации протокола TCP отмечается, что в поле длины заголовка указывается смещение области данных относительно начала сегмента.

---

**Таблица 13.1. Значение битов кода сегмента TCP-заголовка**

---

<i>Название бита</i>	<i>Значение, если бит установлен в 1 (слева направо)</i>
URG	В заголовке присутствует указатель срочных данных
ACK	В заголовке указано поле подтверждения приема
PSH	В данном сегменте указан запрос на немедленную отправку данных (push)
RST	Сброс соединения
SYN	Синхронизация порядковых номеров
FIN	Отправитель достиг конца потока данных

---

В поле *размера окна* отправляемого сегмента программы протокола TCP помещают количество октетов, которые они могут принять (т.е. указывают размер своего приемного буфера). В это поле помещается 16-битовое беззнаковое целое число, имеющее стандартный сетевой порядок следования байтов. Анонсирование окна — еще один пример дуплексной передачи данных, поскольку оно выполняется для всех сегментов, включая сегменты, в которых передаются данные, и сегменты с сигналами подтверждения приема.

## 13.12. Передача данных вне очереди

Поскольку протокол TCP является потоково-ориентированным, иногда прикладной программе, запущенной на одном конце соединения, необходимо отправить данные другой программе *вне очереди* (*out of band*). Речь идет о том, чтобы программа на другом конце соединения получила их сразу, не ожидания приема октетов, которые были отправлены раньше. Например, если протокол TCP используется для создания сеанса связи с удаленным терминалом, пользователю иногда нужно послать удаленной программе специальный сигнал с клавиатуры, который *прервет* (*interrupt*) ее выполнение или вовсе *завершит* (*abort*) работу программы. Обычно подобные сигналы чаще всего посылаются, если программа, запущенная на удаленном компьютере, зависла или перестала корректно работать. Понятно, что управляющие сигналы должны быть переданы программе вне очереди, без ожидания, пока она считает из входного потока все посланные ранее октеты. В противном случае, если программа по какой-либо причине прекратит считывание потока данных, то управляющие сигналы никогда не смогут ее достичь.

Для реализации режима передачи внеочередных сигналов в протоколе TCP для отправителя предусмотрена возможность помечать находящиеся в сегменте данные как *срочные* (*urgent*). Это означает, что программа-получатель должна бытьзвещена о поступлении таких данных так быстро, насколько это возможно, вне зависимости от состояния входящего потока данных. В спецификации протокола указано, что после поступления срочных данных, модуль протокола TCP должен уведомить прикладную программу, открывшую соединение, о переходе в “срочный режим” работы. После того как поток срочных данных будет исчерпан, модуль протокола TCP вновь должен уведомить прикладную программу о переходе в обычный режим работы.

Естественно, что все детали механизма уведомления прикладной программы о прибытии срочных данных зависят от используемой на компьютере операционной системы. Однако при отправке срочных данных этот механизм унифицирован. В поле *кода сегмента* должен быть установлен бит срочных данных URG, и

в поле *указателя срочных данных* занесено соответствующее значение. Если бит URG установлен, то поле *указателя срочных данных* определяет позицию в сегменте, где заканчиваются срочные данные.

### 13.13. Параметр максимального размера сегмента

Следует отметить, что через открытые соединения посылаются сегменты различного размера. Естественно, обе стороны должны заранее договориться о том, какова максимально возможная длина передаваемого сегмента. Для обмена информацией с модулем протокола TCP, запущенным на противоположной стороне соединения, используется поле *параметров протокола*. Один из этих параметров позволяет программе протокола TCP указать *максимальный размер сегмента* (*maximum segment size*, или *MSS*), который она будет принимать. Например, если небольшой встроенный микрокомпьютер, оснащенный малым объемом оперативной памяти подключается по сети к мощному суперкомпьютеру, он сначала должен сообщить суперкомпьютеру максимальный размер сегмента, чтобы получаемые от суперкомпьютера сегменты могли поместиться в выделенных для этой цели буферах.

Описываемый параметр играет важную роль, когда компьютеры подключены к высокоскоростной локальной сети. В этом случае для повышения пропускной способности сети максимальный размер сегмента должен быть таким, чтобы сегмент целиком помещался в пакете. Поэтому, если отправитель и получатель находятся в пределах одной физической сети, модуль протокола TCP обычно выбирает максимальный размер сегмента так, чтобы получившаяся в результате IP-дейтаграмма соответствовала максимальной единице передачи данных (MTU) физической сети. Если же отправитель и получатель находятся в разных физических сетях, то можно попытаться определить минимальный размер MTU сети, находящейся по пути передачи пакетов, или же установить максимальный размер сегмента равным 536 октетов, т.е. стандартный размер IP-дейтаграммы (576 октетов) минус стандартный размер IP- и TCP-заголовков.

В реально работающей объединенной сети выбрать оптимальный размер сегмента не так-то просто, поскольку ее пропускная способность будет снижаться при передаче как очень больших, так и малых сегментов. С одной стороны, при небольшом размере сегмента эффективность использования сети остается крайне низкой. Чтобы понять, почему так происходит, необходимо вспомнить, что TCP-сегменты при передаче инкапсулируются в IP-дейтаграммы, а те, в свою очередь, — в физические сетевые фреймы. Поэтому, кроме данных, к каждому сегменту добавляется по меньшей мере еще 40 октетов — заголовки TCP-сегмента и IP-дейтаграммы. В результате при передаче дейтаграммы, содержащей только один октет данных, для передачи пользовательских данных используется всего 1/41 часть выделенной полосы пропускания сети. На практике, если учесть минимальный межпакетный промежуток и биты синхронизации фреймов, автоматически добавляемые сетевым оборудованием, то это соотношение будет еще меньше.

С другой стороны, очень большие сегменты также снижают производительность сети. Для больших сегментов создаются большие IP-дейтаграммы. Когда такая дейтаграмма передается по сети с малым значением MTU, на уровне протокола IP она фрагментируется. В отличие от TCP-сегментов, подтверждение получения каждого фрагмента на уровне протокола не выполняется, кроме того отдельные фрагменты дейтаграммы при необходимости не могут быть повторно переданы независимо от всей дейтаграммы. Таким образом, в случае утери или повреждения одного из фрагментов повторно должна быть передана вся дейтаграмма. Поскольку вероятность потери отдельного фрагмента отлична от нуля, увеличение размеров сегмента больше определенного значения (порога

фрагментации) уменьшает вероятность успешного приема дейтаграммы и, следовательно, ведет к уменьшению пропускной способности сети.

Теоретически оптимальный размер сегмента  $S$  определяется максимально возможным размером генерируемых IP-дейтаграмм, которые на протяжении всего маршрута от отправителя до конечного получателя не фрагментируются. На практике нахождение значения  $S$  затруднено по нескольким причинам. Во-первых, в большинстве реализаций протокола TCP не включен механизм для определения оптимального размера сегмента<sup>5</sup>. Во-вторых, поскольку маршрутизаторы в объединенной сети могут динамически изменять маршруты дейтаграмм, возможно изменение пути их следования между двумя взаимодействующими компьютерами. Следовательно, величина MTU в любой момент может без предупреждения измениться, после чего неминуемо последует фрагментация дейтаграмм. В-третьих, оптимальный размер сегмента зависит от длины заголовков протоколов более низкого уровня. Кроме того, при использовании параметров протокола IP размер сегмента должен быть еще меньше. Поэтому с грустью приходится констатировать, что проблема нахождения оптимального размера сегментов протокола TCP не решена до сих пор.

### 13.14. Вычисление контрольной суммы TCP-сегмента

В поле *контрольной суммы* TCP-заголовка помещается 16-битовое целое число, которое используется для проверки целостности полученных данных и заголовка TCP-сегмента. При вычислении контрольной суммы модуль протокола TCP, работающий на машине отправителя, использует алгоритмы, аналогичные описанным в главе 12, “Передача пользовательских дейтаграмм (UDP)”, для протокола UDP. Перед вычислением контрольной суммы к началу сегмента добавляется *псевдозаголовок*, а в конец сегмента — необходимое количество нулевых битов для выравнивания общей длины полученной конструкции на границу 16-битов. После выполнения описанных подготовительных действий вычисляется значение 16-битовой контрольной суммы полученной конструкции. Длина псевдозаголовка и битов, добавленных для выравнивания, не учитывается в общей длине TCP-сегмента, поскольку они не передаются получателю. Кроме того, при вычислении значение поля *контрольной суммы* TCP-заголовка полагается равным нулю. Как и при вычислении других контрольных сумм, при сложении используется двоичная арифметика с представлением отрицательных чисел в инверсном коде (так называемое сложение с учетом знака). Затем полученный результат инвертируется, чтобы получилось положительное значение контрольной суммы. После получения сегмента, модуль протокола TCP выполняет над ним аналогичные вычисления и сравнивает контрольные суммы. Если они совпадают, значит сегмент получен без ошибок.

Псевдозаголовок используется в тех же целях, что и в протоколе UDP, — он позволяет принимающей стороне удостовериться в том, что сегмент доставлен по назначению, путем проверки IP-адресов отправителя и конечного получателя, а также номера порта протокола. В протоколе TCP IP-адреса отправителя и конечного получателя играют очень важную роль, поскольку они используются для идентификации соединения, к которому относится полученный сегмент. Поэтому после прибытия дейтаграммы, содержащей TCP-сегмент, модуль протокола IP

<sup>5</sup> Для нахождения минимального значения MTU на всем пути следования дейтаграмм отправитель должен прозондировать маршрут их следования с помощью специальных IP-дейтаграмм, в заголовке которых установлен бит запрета фрагментации. При получении ICMP-сообщения об ошибке фрагментации отправитель должен уменьшить размер дейтаграммы и повторить зондирование. Описанный алгоритм выполняется до тех пор, пока дейтаграмма не пройдет весь маршрут без ошибок.

должен передать модулю протокола TCP, кроме самого сегмента, еще и IP-адреса отправителя и конечного получателя. Формат псевдозаголовка, использующегося для вычисления контрольной суммы TCP-сегмента, показан на рис. 13.8.

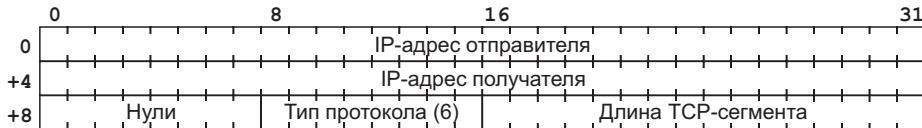


Рис. 13.8. Формат псевдозаголовка, использующегося для вычисления контрольной суммы TCP-сегмента. На принимающей стороне эта информация извлекается из IP-дейтаграммы, содержащей TCP-сегмент

При отправке сегмента модуль протокола TCP помещает в поле *типа протокола* значение, которое затем будет использоваться в соответствующем поле типа протокола низкоуровневой системы доставки. Для IP-дейтаграмм, содержащих TCP-сегменты, это значение, согласно спецификации, равно 6. В поле *длины TCP-сегмента* указывается общая длина сегмента вместе с заголовком. На принимающей стороне информация, помещаемая в псевдозаголовок, извлекается из IP-дейтаграммы, содержащей TCP-сегмент, и используется затем в процессе вычисления контрольной суммы. По значению контрольной суммы можно судить о том, доставлен ли сегмент указанному получателю без ошибок.

### 13.15. Подтверждение приема и повторная передача

Поскольку в протоколе TCP данные посылаются в виде сегментов переменного размера и в повторно передаваемые сегменты, кроме оригинала, могут быть включены дополнительные данные, получаемые сигналы подтверждения приема не так-то просто соотнести с отправленными дейтаграммами или сегментами. По этой причине сигналы подтверждения приема соотносятся с положением данных в передаваемом потоке, которым присваиваются порядковые номера. Получатель собирает поступившие в сегментах данные и воспроизводит точную копию передаваемого потока данных. Так как сегменты поступают получателю в виде IP-дейтаграмм, которые в процессе передачи могут быть утеряны или доставлены в другом порядке, для упорядочения сегментов получатель использует их порядковые номера. В какой-то момент времени у получателя может быть восстановлено произвольное количество октетов (включая ноль) от начала потока. Кроме того, получателю может быть доставлена еще какая-то часть потока в виде пропущенных вне очереди дейтаграмм. Несмотря на это, получатель всегда посылает сигналы подтверждения приема только на самую большую и непрерывную часть потока (считая от его начала), которая была корректно получена. В каждом подтверждении приема указывается порядковый номер, который соответствует положению самого старшего октета в непрерывной части доставленного потока (точнее, больше его на единицу). Таким образом, по мере передачи потока данных отправитель постоянно получает сигналы обратной связи, отражающие состояние принятого потока данных. Исходя из сказанного выше можно сделать такой вывод.

*В подтверждениях приема протокола TCP указывается порядковый номер в потоке следующего октета, который должен быть принят получателем.*

Принятую в протоколе TCP систему подтверждения приема называют *кумулятивной (cumulative)*, поскольку она отражает количество октетов потока данных, накопленных получателем. Следует отметить, что кумулятивная система

подтверждения приема имеет как преимущества, так и недостатки. Одно из преимуществ состоит в том, что сигналы подтверждения приема легко генерируются и являются однозначными. Еще одно преимущество заключается в том, что в случае потери сигналов подтверждения приема не нужно повторно передавать ни их, ни соответствующие им сегменты данных. Недостатком описываемого кумулятивного метода является отсутствие у отправителя информации об успешных передачах. Ему только известно, какая часть потока была успешно доставлена получателю.

Чтобы понять, почему отсутствие информации об успешно выполненных передачах делает кумулятивный метод подтверждения приема менее эффективным, давайте рассмотрим такой пример. Предположим существует окно, расположенное в потоке данных с позиции 101, в которое включено 5000 октетов. Предположим также, что отправитель передал все находящиеся в окне данные в виде пяти сегментов. А теперь представим, что первый сегмент в процессе передачи был утерян, тогда как все остальные успешно достигли получателя. По мере прибытия сегментов, получатель будет отправлять сигналы подтверждения приема. В каждом из них будет указан порядковый номер октета 101, т.е. номер следующего по порядку старшего октета, принадлежащего непрерывному потоку, который ожидает получить принимающая сторона. В данном случае у получателя нет никакой возможности сообщить отправителю, что большая часть данных, принадлежащих к текущему окну, уже получена.

Когда истекает время ожидания сигнала подтверждения приема, отправитель попытается выйти из сложившейся ситуации одним из двух потенциально неэффективных способов — повторно передать один сегмент или сразу все пять. Очевидно, что повторная передача пяти сегментов неэффективна. После того как первый сегмент будет успешно доставлен, у получателя будет полный комплект данных, составляющих окно. Поэтому он подтвердит прием октета с порядковым номером 5101. Таким образом, если отправитель будет действовать согласно принятым стандартам и выполнит повторную передачу только первого не принятого сегмента, то прежде, чем предпринимать дальнейшие шаги, он должен дождаться подтверждения приема этого сегмента. В результате отправитель невольно возвратится к простейшему методу подтверждения приема, описанному в начале этой главы. При этом все преимущества использования большого окна теряются.

### 13.16. Время ожидания и повторная передача сегментов

Одно из самых важных и сложных понятий протокола TCP — метод обработки истекшего времени ожидания и выполнения повторной передачи. Как и в других надежных протоколах, в протоколе TCP предполагается, что получатель пришлет сигнал подтверждения приема, успешно получив новые октеты из потока данных. Каждый раз при отправке сегмента в модуле протокола TCP устанавливается значение таймера ожидания получения сигнала подтверждения приема. Если время ожидания истечет, прежде чем будет подтвержден прием отправленного сегмента, модуль протокола TCP считает, что сегмент не достиг получателя или его данные были искажены в процессе передачи, и передает его повторно.

Чтобы понять, почему алгоритм повторной передачи протокола TCP отличается от аналогичных алгоритмов, используемых в других сетевых протоколах, необходимо вспомнить, что протокол TCP изначально предназначался для работы в межсетевом окружении. Передаваемый по объединенной сети между парой машин сегмент может проходить по одной сети с низким временем задержки (т.е. по высокоскоростной локальной сети) либо передаваться через несколько взаимодействующих между собой сетей, объединенных маршрутизаторами. Поэтому заранее невозможно предсказать, насколько быстро отправитель получит сигнал подтверждения приема. Ситуация усугубляется еще и тем, что задержка

прохождения каждого маршрутизатора зависит от величины трафика в конкретной сети. Поэтому суммарное время задержки на передачу сегмента и получение отправителем подтверждения его приема может колебаться в очень широких пределах. Это проиллюстрировано на рис. 13.9, где изображены данные измерений полного (т.е. туда и обратно) времени доставки 100 последовательных пакетов по глобальной сети Internet. Из графика видно, что в программах протокола TCP должны учитываться два фактора: существенные отличия во времени передачи пакетов разным получателям и колебания времени задержки при доставке пакетов одному получателю, вызванные изменением трафика.

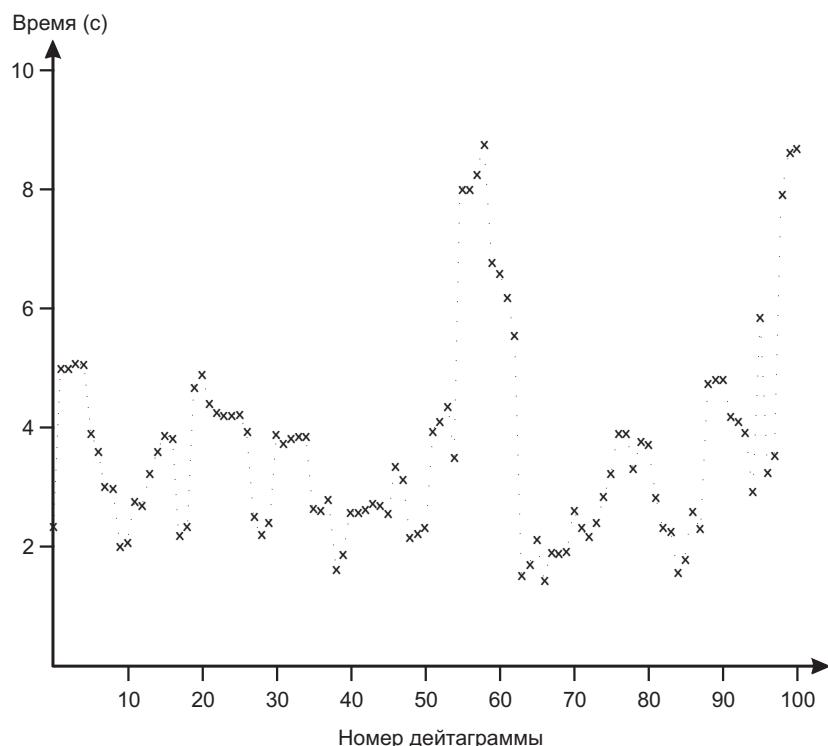


Рис. 13.9. Данные измерений полного времени доставки 100 последовательных IP-дейтограмм. Хотя в современной сети Internet задержки передачи пакетов намного ниже, разброс остается таким же большим

Для учета изменяющегося в широких пределах времени задержки в объединенной сети в протоколе TCP используется *адаптивный алгоритм повторной передачи* (*adaptive retransmission algorithm*). В общих чертах суть его состоит в том, что в протоколе TCP отслеживается производительность каждого соединения и на основе полученных значений выбирается приемлемое время задержки. По мере изменения производительности соединения соответственно изменяется и величина задержки (т.е. величина задержки адаптируется к изменениям в производительности соединения).

При сборе данных, необходимых для работы адаптивного алгоритма, модуль протокола TCP регистрирует время отправки каждого сегмента и время получения сигналов подтверждения приема данных, находящихся в этих сегментах. По разности определяется время, затраченное на доставку каждого пакета. Описанная выше процедура называются *замером полного времени доставки пакета*

(*sample round trip time*). После очередного замера модуль протокола TCP пересчитывает среднее время доставки для данного соединения. Обычно модуль протокола TCP вычисляет предполагаемую средневзвешенную величину полного времени доставки пакетов *RTT (round trip time)*. При таком подходе вновь полученные результаты замеров мало влияют на среднее время доставки. Например, при вычислении новой средневзвешенной величины по одной из ранних методик использовалось постоянное значение весового коэффициента  $\alpha$ , где  $0 \leq \alpha < 1$ . Это позволяло регулировать степень влияния новых результатов измерений на среднее значение полного времени доставки:

$$RTT = (\alpha \times \text{Старый\_RTT}) + ((1 - \alpha) \times \text{Новый\_RTT})$$

Выбирая величину  $\alpha$  близкой к 1, можно добиться того, чтобы новые результаты измерений, полученные за короткий промежуток времени, не влияли на средневзвешенную величину доставки пакетов. Это позволяет отсеять случайные отклонения значений от среднего, например, если один из сегментов доставлялся к получателю слишком долго. Если величину  $\alpha$  выбрать близкой к 0, то средневзвешенная величина доставки пакетов будет достаточно быстро корректироваться в соответствии с текущими изменениями времени доставки пакетов.

При отправке пакета, модуль протокола TCP вычисляет величину тайм-аута как функцию от текущего значения предполагаемой величины полного времени доставки. В ранних реализациях протокола TCP использовалась линейная функция с постоянным весовым коэффициентом  $\beta$  ( $\beta > 1$ ). Это позволяло сделать величину тайм-аута большей, чем текущее значение предполагаемой величины полного времени доставки:

$$\text{Тайм-аут} = \beta \times RTT$$

Выбрать величину  $\beta$  не так-то просто. С одной стороны, величина тайм-аута не должна сильно отличаться от текущего времени полной доставки пакетов (т.е. значение коэффициента  $\beta$  должно быть близким к 1). Это позволит быстро обнаруживать потери пакетов и благодаря этому увеличить пропускную способность сети, поскольку модулю протокола TCP не нужно будет понапрасну долго ожидать, пока истечет тайм-аут перед повторной передачей пакета. С другой стороны, при  $\beta=1$  модуль протокола TCP может генерировать ненужный сетевой трафик, поскольку любая небольшая задержка в доставке пакета будет сразу же вызывать его повторную передачу. В результате пропускная способность сети снизится. В первоначальной спецификации стандарта TCP рекомендуется выбирать значение  $\beta=2$ . Однако проведенные недавно исследования (о них пойдет речь ниже) позволили выработать более точную методику выбора времени тайм-аута. Таким образом можно подвести некоторые итоги.

*Чтобы учесть большой разброс задержек во времени доставки пакетов в реально действующей обединенной сети, в протоколе TCP используется адаптивный алгоритм повторной передачи. Суть его состоит в том, что величина тайм-аута выбирается в соответствии с текущим временем задержки доставки пакета, которое постоянно отслеживается во всех открытых соединениях.*

### 13.17. Точный замер полного времени доставки пакета

Теоретически измерить полное время доставки пакета несложно: нужно из времени получения сигнала подтверждения доставки сегмента вычесть время отправки сегмента. Следует заметить, что при этом могут возникнуть определенные за-

труднения, поскольку в протоколе TCP используется кумулятивная система подтверждения приема. Суть ее заключается в том, что сигнал подтверждения приема отражает факт успешного получения данных, но не конкретной дейтаграммы, в которой эти данные были переданы. Рассмотрим подробнее процесс повторной передачи. Сначала модуль протокола TCP формирует сегмент, помещает его в дейтаграмму и отправляет ее получателю. После исчерпания значения таймера выполняется повторная передача сегмента, но уже в другой дейтаграмме. Поскольку в обеих дейтаграммах содержатся одни и те же данные, отправитель не может узнать, к какой из дейтаграмм (оригинальной или повторной) относится полученный сигнал подтверждения приема. Налицо явление, которое было названо *неоднозначностью сигналов подтверждения приема (acknowledgment ambiguity)*. Таким образом, сигналы подтверждения приема данных в протоколе TCP являются *неоднозначными*.

Проблема заключается в том, к какому из двух моментов передачи модулю протокола TCP следует привязать полученный сигнал подтверждения приема — к более раннему (т.е. оригинальному) или к более позднему (т.е. к тому, который был выполнен совсем недавно). Самое удивительное заключается в том, что нельзя принять ни одно из этих предположений. Если сигнал подтверждения приема соотнести с моментом передачи оригинальной дейтаграммы, то если эта дейтаграмма будет утеряна, предполагаемое полное время доставки пакета будет большим<sup>6</sup>. Рано или поздно сигнал подтверждения приема будет получен, даже если для этого потребуется выполнить одну или несколько повторных передач сегмента. В результате модуль протокола TCP измерит полное время доставки сегмента относительно времени его первоначальной посылки и на основе этого чрезвычайно большого значения вычислит новое значение RTT. Таким образом, по сравнению со старым, новое значение RTT вырастет незначительно. В следующий раз, когда модуль протокола TCP будет отправлять сегмент получателю, увеличенное значение RTT приведет к заметному увеличению времени ожидания сигнала, подтверждения приема (тайм-аута). Поэтому, если сигнал подтверждения приема будет получен также после одной или нескольких повторных передач, полное время доставки пакета будет еще больше, и т.д.

Нельзя также соотносить сигнал подтверждения приема со временем самой последней повторной передачи сегмента. Давайте рассмотрим, что произойдет, если внезапно возрастет полное время доставки сегмента. При отправке сегмента модулем протокола TCP для вычисления времени тайм-аута будет использоваться старое (небольшое) значение предполагаемого полного времени доставки пакета. Предположим, что, после успешной доставки сегмента получателю, отправителю был послан сигнал подтверждения приема. Однако из-за перегрузок в сети сигнал подтверждения приема не будет получен до истечения значения таймера. В этом случае модуль протокола TCP выполнит повторную передачу сегмента. Вскоре после этого отправитель получит первый сигнал подтверждения приема и соотнесет его с моментом последней повторной передачи. Измеренное полное время доставки пакета будет маленьким, что приведет к небольшому снижению значения предполагаемого полного времени доставки пакета, или RTT. К сожалению, уменьшение значения RTT приведет к тому, что для передачи следующего сегмента модуль протокола TCP выберет малое значение тайм-аута. В конечном счете, значение предполагаемого полного времени доставки пакета стабилизируется на величине  $T$ , такой, что корректное значение полного времени доставки будет немного превышать значение  $T$ , помноженное на некоторый коэффициент. Проведенные исследования показали, что в тех реализациях

<sup>6</sup> В худшем случае, когда утеряны все посланные сегменты, значение предполагаемого времени доставки увеличивается до очень больших значений.

протокола TCP, где сигналы подтверждения приема соотносятся с моментом последней повторной передачи сегмента, стабильное значение RTT немного меньше половины корректного значения полного времени доставки. Следовательно, при отсутствии потерь сегментов в сети, модуль протокола TCP будет два раза посыпать один и тот же сегмент получателю.

### 13.18. Алгоритм Карна и коррекция тайм-аута

В предыдущем разделе шла речь о том, что независимо от того к какому из моментов времени будет соотнесен полученный сигнал подтверждения приема (к передаче оригинального сегмента или к его повторной передаче), измерение полного времени доставки пакета будет неточным. Как же быть? Традиционный ответ на этот вопрос очень простой: модуль протокола TCP не должен пересчитывать значение предполагаемого полного времени доставки пакета (RTT) на основе данных, полученных при повторной передаче сегмента. Эта идея, называемая также *алгоритмом Карна (Karn's Algorithm)*, позволяет полностью избежать проблем, связанных с неоднозначностью сигналов подтверждения приема. Значение предполагаемого полного времени доставки пакета должно вычисляться только на основании данных, полученных для однозначных сигналов подтверждения приема (т.е. тех сигналов, которые были получены в ответ на однократную передачу пакета).

Естественно, что упрощенный вариант реализации алгоритма Карна (тот, в котором попросту игнорируются замеры, сделанные для повторно переданных сегментов) также не может использоваться для определения точного времени доставки сегмента. Давайте рассмотрим, что произойдет, если в момент отправки сегмента модулем протокола TCP резко возрастет задержка в сети. Напомним, что значение тайм-аута вычисляется на основании текущего значения предполагаемого полного времени доставки пакета. Очевидно, что для больших задержек в сети вычисленное значение тайм-аута будет слишком малым. Это вызовет повторную передачу сегмента. Таким образом, если игнорировать сигналы подтверждения приема для повторно переданных сегментов, новое значение предполагаемого полного времени доставки пакета никогда не изменится, и описанный выше процесс будет продолжаться до тех пор, пока не уменьшится время задержки.

Чтобы устранить подобные недостатки, в алгоритме Карна используется метод *коррекции значения тайм-аута (timer backoff)*. Суть его состоит в том, что начальное значение тайм-аута вычисляется на основании текущих данных (по формуле, подобной приведенной выше). Однако, если в результате истечения тайм-аута произойдет повторная передача сегмента модуль протокола TCP увеличит значение тайм-аута. На практике, каждый раз перед повторной передачей сегмента, модуль протокола TCP увеличивает значение тайм-аута. Чтобы не допустить бесконтрольного увеличения тайм-аута, в большинстве реализаций оно ограничивается сверху заранее заданным значением, которое всегда больше максимально возможной задержки передачи пакета по любому из маршрутов объединенной сети.

Алгоритм вычисления нового значения тайм-аута зависит от конкретной реализации протокола TCP. В большинстве реализаций оно вычисляется путем умножения старого значения тайм-аута на специальный множитель  $\gamma$ :

$$\text{Новый\_тайм-аут} = \gamma \times \text{Тайм-аут}$$

Обычно  $\gamma$  выбирается равным 2. (Выше уже шла речь о том, что при  $\gamma < 2$  система будет работать нестабильно.) В некоторых реализациях протокола TCP значение

множителя выбирается из таблицы. Таким образом, значение нового тайм-аута будет непосредственно зависеть от номера шага<sup>7</sup>.

Для того чтобы решить проблему постоянства предполагаемого полного времени доставки пакета, в алгоритме Карна используется методика принудительного изменения тайм-аута.

*Суть алгоритма Карна состоит в следующем. При вычислении предполагаемого полного времени доставки пакета игнорируются замеры, сделанные для повторно посланных сегментов. Для определения точного времени доставки пакетов в алгоритме Карна используется методика увеличения значения тайм-аута при повторной передаче сегмента.*

Обобщая все сказанное выше можно отметить, что при возникновении больших задержек в объединенной сети, алгоритм Карна позволяет разделить вычисление текущего значения тайм-аута и определение предполагаемого полного времени доставки пакета. Предполагаемое значение полного времени доставки используется только для вычисления начального значения тайм-аута. При каждой повторной передаче сегмента значение тайм-аута увеличивается на некоторую величину до тех пор, пока сегмент не будет успешно доставлен получателю. Таким образом, при отправке последовательности сегментов используется значение тайм-аута, полученное в результате принудительной коррекции таймера. Так происходит до тех пор, пока не будет получен сигнал подтверждения приема, соответствующий однократно посланному сегменту. После этого модуль протокола TCP на основании сделанного замера пересчитывает предполагаемое значение полного времени доставки и соответствующим образом изменяет значение тайм-аута. Эксперименты показывают, что алгоритм Карна хорошо себя зарекомендовал даже в тех сетях, где потери пакетов очень велики<sup>8</sup>.

### 13.19. Обработка большого разброса значений задержки

Исследования связанные с вычислением предполагаемого значения полного времени доставки пакетов показали, что описанные выше методы неприменимы в том случае, если значение времени задержки в сети колеблется в очень широких пределах. В теории массового обслуживания доказывается, что отклонение полного времени доставки пакетов  $\sigma$ , изменяется пропорционально  $1/(1-L)$ , где  $L$  — коэффициент текущей загрузки сети ( $0 \leq L \leq 1$ ). Таким образом, при 50-процентной загрузке объединенной сети предполагаемое полное время доставки пакетов будет изменяться в пределах  $\pm 2\sigma$ , или пропорционально коэффициенту 4. При 80-процентной загрузке сети коэффициент изменения будет равняться 10. Выше была описана методика определения предполагаемого значения полного времени доставки на основе информации, приведенной в первоначальном варианте стандарта протокола TCP. Таким образом, полагая значение  $\beta=2$  (оно рекомендовано в первоначальном стандарте), получим, что описанная выше методика применима при загрузках сети, колеблющихся в районе 30%.

В спецификации протокола TCP 1989 года требуется, чтобы в реализациях протокола оценивалось как среднее время полной доставки пакетов, так и величины его отклонения. При этом предполагаемая величина отклонения должна

<sup>7</sup> Самой известной операционной системой, в которой используется таблица множителей, является Berkeley UNIX. Однако в текущих версиях системы в таблице хранятся одинаковые значения, эквивалентные  $\gamma=2$ .

<sup>8</sup> Филипп Карн — радиолюбитель и энтузиаст своего дела. Он придумал этот алгоритм для того, чтобы можно было использовать протокол TCP для передачи данных по радиоканалу в условиях помех, когда происходят большие потери пакетов.

использоваться в формуле вместо коэффициента  $\beta$ . В результате новые реализации протокола TCP могут работать в очень широком диапазоне задержек и позволяют существенно повысить пропускную способность сети. Большим преимуществом описанного выше метода является то, что он не требует проведения сложных вычислений. Высокоэффективная программа может быть создана на основе нескольких простых формул, приведенных ниже.

$$\begin{aligned} \text{РАЗНОСТЬ} &= \text{ЗАМЕР} - \text{Старый\_RTT} \\ \text{Сглаженный\_RTT} &= \text{Старый\_RTT} + \delta \times \text{РАЗНОСТЬ} \\ \text{ОТКЛ} &= \text{Старое\_ОТКЛ} + \rho (|\text{РАЗНОСТЬ}| - \text{Старое\_ОТКЛ}) \\ \text{ТАЙМ-АУТ} &= \text{Сглаженный\_RTT} + \eta \times \text{ОТКЛ} \end{aligned}$$

Здесь:

- $\text{ОТКЛ}$  — оценка среднего значения отклонения;
- $\delta$  — дробный коэффициент, значение которого лежит между 0 и 1, определяющий насколько сильно значения новых замеров будут влиять на средневзвешенную величину  $RTT$ ;
- $\rho$  — дробный коэффициент, значение которого лежит между 0 и 1, определяющий насколько сильно значения новых замеров будут влиять на оценку среднего значения отклонения;
- $\eta$  — коэффициент, определяющий насколько сильно оценка среднего значения отклонения будет влиять на величину тайм-аута.

Для повышения скорости вычислений в протоколе TCP значения коэффициентов  $\delta$  и  $\rho$  выбираются так, чтобы они были кратны  $1/2^n$ . Тогда для вычислений по приведенным выше формулам можно использовать целочисленную арифметику<sup>9</sup>. Опытным путем установлено, что приемлемый результат достигается при  $\delta=1/2^3$  и  $\rho=1/2^2$ . Первоначальное значение коэффициента  $\eta$  в 4.3BSD UNIX равнялось 2. Однако в версии 4.4 BSD UNIX оно было изменено на 4.

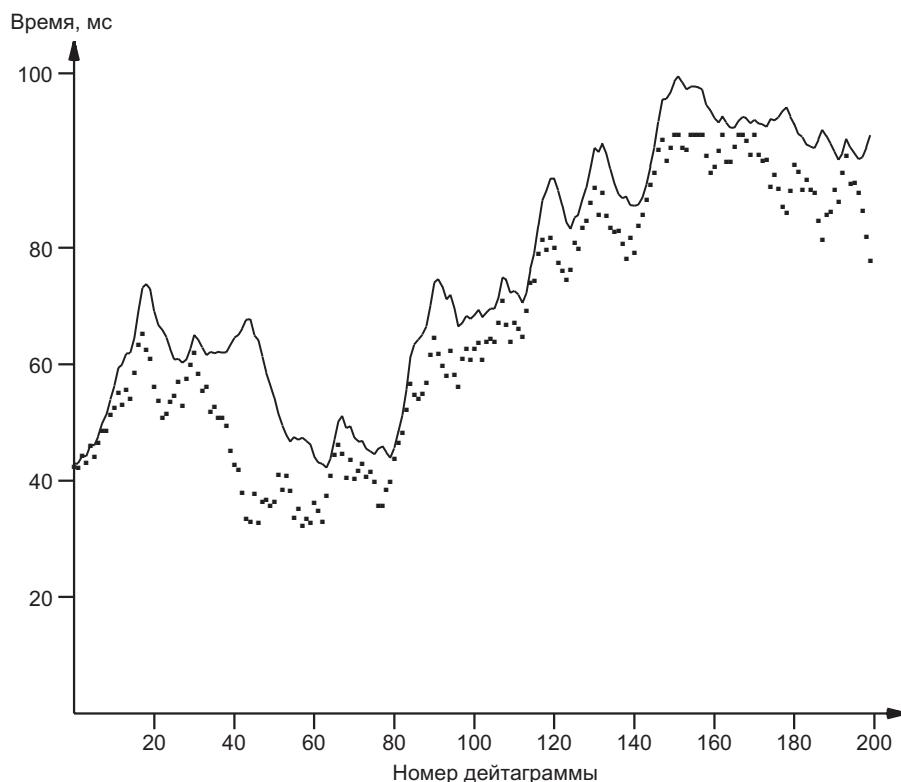
На рис. 13.10 с помощью набора случайно выбранных величин продемонстрировано, как изменяется вычисленное значение тайм-аута в зависимости от разброса полного времени доставки пакета. Хотя данные для графика получены искусственным путем, они соответствуют реальной ситуации: при передаче последовательности пакетов наблюдается малый разброс задержек, а общее среднее время то увеличивается, то уменьшается.

Частое изменение полного времени доставки (включая циклы увеличения и уменьшения) может вызвать увеличение среднего значения тайм-аута. Более того, хотя при увеличении задержек значение тайм-аута растет очень быстро, при их уменьшении это значение падает не так быстро.

Чтобы показать, как описанный алгоритм вычисления тайм-аута будет реагировать на чрезвычайно большие изменения задержки, мы воспользовались данными, приведенными на рис. 13.9 и построили график изменения значения тайм-аута (рис. 13.11). Напомним, что наша цель — вычислить текущее значение тайм-аута так, чтобы оно как можно ближе соответствовало реальному полному времени доставки пакета, и чтобы при этом не было недооценки. Из рис. 13.11 видно, что, хотя значение тайм-аута довольно близко соответствует реальным данным, случаи недооценки полностью исключить нельзя. Например, во время передачи двух последовательных дейтаграмм, обозначенных стрелками, величина задержки возросла более чем в два раза (была менее 4-х секунд, а стала больше 8). Важно отметить, что внезапное большое изменение

<sup>9</sup> Напомним, что для деления целого числа на  $2^n$  нужно сдвинуть его двоичное представление на  $n$  разрядов вправо, а для умножения — на  $n$  разрядов влево.

задержки произошло после периода относительной стабильности, когда разброс задержек был невелик. Поэтому нельзя применить какой-либо алгоритм, позволяющий предсказать это изменение. В рассматриваемом нами случае применение алгоритма протокола TCP для вычисления тайм-аута (его значение равно 5 с) привело к существенной недооценке времени задержки. В результате будет выполнена ненужная повторная передача одного из сегментов. Тем не менее оценочное значение тайм-аута очень быстро “догоняет” возросшее значение задержки, а это означает, что все последующие пакеты будут доставлены получателю без повторной передачи.



*Рис. 13.10. Изменение вычисленного значения тайм-аута в зависимости от разброса полного времени доставки пакета. Точками показан набор из 200 случайно выбранных значений полного времени доставки, а вычисленное значение тайм-аута изображено сплошной линией. Обратите внимание: увеличение времени задержки вызывает соответствующее увеличение значения тайм-аута*

### 13.20. Реакция на перегрузку сети

На первый взгляд может показаться, что при разработке модуля протокола TCP нужно учитывать только механизм взаимодействия между двумя конечными точками соединения и задержки, возникающие при передаче пакетов между этими точками. Однако на практике модуль протокола TCP должен также учитывать *перегрузку*, возникающую в объединенной сети. О возникновении перегрузки свидетельствует резкое увеличение задержки при доставке пакетов, вызванное скоплением большого количества дейтаграмм на одном или нескольких

устройствах коммутации (или маршрутизаторах). При перегрузке устройства маршрутизации задержка увеличивается, поскольку поступающие дейтаграммы ставятся в очередь и находятся там до тех пор, пока маршрутизатор не сможет их обработать. Не стоит забывать, что емкость запоминающего устройства каждого маршрутизатора имеет ограниченный объем и что каждая дейтаграмма занимает часть этой памяти. Другими словами, при передаче дейтаграмм по объединенной сети никакие ресурсы памяти заранее не выделяются для отдельных TCP-соединений. В самом худшем случае, когда на перегруженный маршрутизатор поступает такое количество дейтаграмм, что они не могут разместиться в его памяти, все "лишние" дейтаграммы попросту теряются.

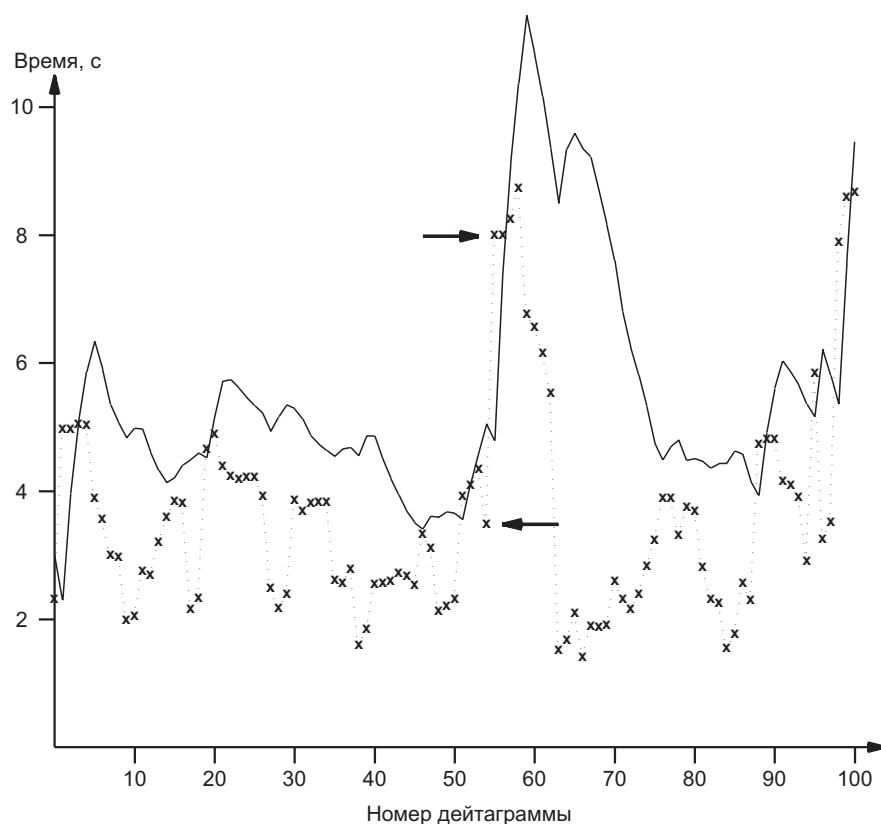


Рис. 13.11. Значение тайм-аута, вычисленное с помощью алгоритма протокола TCP для данных, приведенных на рис. 13.9. Стрелками обозначены две последовательные дейтаграммы, при передаче которых величина задержки возросла в два раза

Как правило, конечным точкам соединения не известно, в каком месте объединенной сети и по какой причине возникла перегрузка. Для них перегрузка просто означает увеличение задержки. К сожалению, в большинстве транспортных протоколов используется стратегия повторной передачи пакетов в случае истечения времени ожидания сигнала подтверждения приема. Поэтому увеличение задержки вызывает повторную передачу дейтаграмм, что, в свою очередь, еще больше усугубляет ситуацию. Если ничего не предпринять, то увеличение трафика вызовет увеличение задержки, а увеличение задержки снова приведет к

увеличению трафика в сети, и т.д. до тех пор, пока работа сети не будет полностью парализована. Подобная ситуация называется *полным затором в сети* (*congestion collapse*).

Чтобы избежать полного затора в сети, при возникновении перегрузки модуль протокола TCP должен уменьшить интенсивность передачи пакетов. Устройства маршрутизации постоянно следят за длиной внутренней очереди дейтаграмм и в случае ее переполнения сообщают о возникшей перегрузке всем своим отправителям с помощью механизма, напоминающего рассылку ICMP-сообщений о давлении источника данных<sup>10</sup>. Однако протоколы транспортного уровня, наподобие TCP, также могут регулировать степень перегрузки в сети, автоматически снижая интенсивность передачи данных при возникновении задержек. Естественно, что применяемые при этом алгоритмы должны быть тщательно продуманы, поскольку даже при нормальных условиях полное время доставки пакета в объединенной сети может колебаться в очень широких пределах.

Для предотвращения перегрузки в сети современная версия стандарта TCP рекомендует использовать две методики: *медленный запуск* (*slow-start*) и *мультипликативное уменьшение* (*multiplicative decrease*). Обе методики взаимосвязаны и могут быть легко реализованы. Выше уже говорилось о том, что для каждого открытого соединения модуль протокола TCP хранит размер окна получателя (т.е. размер буфера, анонсированного в сигнале подтверждения приема). Для недопущения перегрузки в протоколе TCP установлено еще одно ограничение, которое называется *ограничением размера окна перегрузки* или просто *окном перегрузки*. При возникновении перегрузки оно ограничивает поток данных настолько, чтобы он был меньше размера приемного буфера. Таким образом, в любой момент времени модуль протокола TCP функционирует так, как если бы размер его окна был равен:

Размер\_окна = Мин\_размер (Анонс\_получателя, Окно\_перегрузки)

В стационарном режиме при неперегруженном соединении размер окна перегрузки совпадает с размером окна получателя. При уменьшении размера окна перегрузки уменьшается поток данных, передаваемый модулем протокола TCP через конкретное соединение. Чтобы определить ориентировочный размер окна перегрузки, модуль протокола TCP считает, что большая часть дейтаграмм теряется из-за перегрузки. При этом используется описанная ниже стратегия.

*Стратегия мультипликативного уменьшения заключается в том, что каждый раз после возникновения перегрузки размер окна перегрузки уменьшается в два раза (вплоть до минимума, составляющего одну дейтаграмму). Для тех сегментов, которые попали в окно нового размера, применяется стратегия экспоненциального увеличения значения таймера повторной передачи (тайм-аута).*

Поскольку в протоколе TCP при *каждой* потере пакета размер окна перегрузки уменьшается в два раза, размер основного окна передачи уменьшается по экспоненциальному закону в случае продолжения потери пакетов. Другими словами, при возникновении перегрузки модуль протокола TCP уменьшает величину потока данных в сети по экспоненциальному закону. При этом интенсивность повторной передачи сегментов также снижается экспоненциально. При продолжительной перегрузке модуль протокола TCP в конечном счете ограничивает интенсивность передачи данных до одной дейтаграммы и удваивает значение тайм-аута перед повторной передачей сегмента. Идея состоит в том, чтобы в критических ситуациях система могла резко уменьшить величину потока данных в сети.

---

<sup>10</sup> В перегруженной сети длина очереди дейтаграмм с течением времени возрастает по экспонциальному закону.

При этом маршрутизаторам будет предоставлено достаточно времени на проверку и аннулирование “лишних” дейтаграмм, находящихся во внутренней очереди.

Теперь рассмотрим, как происходит восстановление работоспособности системы на основе протокола TCP после устранения перегрузки. Логично предположить, что все должно выполняться в обратном, по сравнению с методом мультипликативного уменьшения порядке. Т.е., как только в сети нормализуется скорость передачи потока данных, размер окна перегрузки должен удваиваться. Однако следствием подобных действий может стать нестабильность работы системы: ее состояние будет постоянно изменяться в широких пределах — от перегруженности до полного отсутствия трафика. Поэтому в протоколе TCP для постепенного увеличения интенсивности передачи данных используется так называемый метод *медленного запуска*<sup>11</sup> (*slow-start*), суть которого состоит в следующем.

*Методика медленного запуска предназначена для восстановления работоспособности сети после перегрузки, а также для начала передачи данных по новому соединению. При этом первоначальный размер окна перегрузки выбирается равным одному сегменту и каждый раз после получения сигнала подтверждения приема он увеличивается на один сегмент.*

Таким образом, метод медленного запуска позволяет избежать затора в обединенной сети сразу после прекращения перегрузки или при начале передачи данных по новому соединению.

Описанный выше термин “*медленный запуск*” может ввести в заблуждение, поскольку в идеальных условиях запуск механизма передачи данных происходит не так уж медленно. Первоначально размер окна перегрузки устанавливается равным 1 сегменту, после чего выполняется передача этого сегмента, и система переходит в состояние ожидания. После получения сигнала подтверждения приема этого сегмента, размер окна перегрузки устанавливается равным 2 сегментам, после чего посыпается уже 2 сегмента и система снова переходит в состояние ожидания. После получения каждого из двух сигналов подтверждения приема этих сегментов, размер окна перегрузки снова увеличивается на 1 сегмент. Поэтому модуль протокола TCP может отправить уже 4 сегмента. После получения 4 сигналов подтверждения приема размер окна перегрузки будет равен уже 8 сегментам. Таким образом, после завершения четырех циклов передачи-приема, модуль протокола TCP может отправить уже 16 сегментов, что часто превышает размеры приемного окна получателя пакетов. Таким образом, перед тем, как модуль протокола TCP сможет отправить  $N$  сегментов, он должен выполнить  $\log_2 N$  циклов передачи-приема. Очевидно, что даже в случае очень больших размеров окон выход системы на проектную мощность будет довольно быстрым.

Чтобы не допустить слишком быстрого увеличения размеров окна и возникновения дополнительной перегрузки, в протокол TCP введено дополнительное ограничение. Как только размер окна перегрузки достигнет половины своего первоначального значения (того, что было до возникновения перегрузки), модуль протокола TCP переходит к фазе *аннулирования перегрузки* и снижает скорость нарастания размеров этого окна. Во время этой фазы размер окна перегрузки увеличивается на 1 только в том случае, если для всех сегментов, находящихся в окне, будут получены сигналы подтверждения приема.

Подводя итоги всему сказанному, необходимо отметить, что методы медленного запуска, мультипликативного уменьшения, фаза аннулирования перегрузки,

---

<sup>11</sup> Этот термин приписывают Джону Нэйгу (John Nagle). Первоначально этот метод восстановления работоспособности системы назывался плавным запуском (*soft-start*).

измерение отклонения полного времени доставки и экспоненциальное изменение значения тайм-аута позволяют существенно поднять производительность системы на основе протокола TCP без выполнения сложных математических расчетов внутри программы, реализующей функции этого протокола. В тех версиях протокола TCP, где применены все описанные методы, удалось поднять производительность системы от 2 до 10 раз по сравнению с реализациями, где одна или несколько методик не использовались.

### 13.21. Перегрузка сети и усечение хвоста очереди

Выше уже отмечалось, что коммуникационные протоколы разделены на логические уровни. Это облегчает их разработку, поскольку инженеры могут сосредоточиться на решении какой-то одной проблемы. Кроме того, разделение выполняемых функций по уровням является очень полезным и нужным делом. Это означает, что в программу, реализующую функции одного уровня, можно легко вносить изменения. При этом программы других уровней остаются без изменений. Однако в методе разделения есть и негативная сторона — работа программ всех уровней осуществляется изолированно друг от друга. Например, протокол TCP ориентирован на обмен данными между двумя конечными точками соединения. Поэтому его работоспособность сохраняется при изменении маршрута следования дейтаграмм по объединенной сети между этими точками (т.е. при изменении или добавлении дополнительных сетевых маршрутов). Несмотря на это изоляция уровней непосредственно оказывается на возможности взаимодействия между ними. В частности, хотя модуль протокола TCP машины отправителя может взаимодействовать с модулем протокола машины получателя, он не может взаимодействовать с модулями протоколов более низкого уровня, находящимися на пути следования пакетов. Таким образом, модули протокола TCP отправителя и получателя никогда не смогут получить отчет о текущем состоянии сети, а также проинформировать модули протоколов низкого уровня, находящиеся на пути следования пакетов о начале передачи данных.

Разработчики заметили одну интересную особенность: отсутствие средств взаимодействия между уровнями часто приводит к тому, что изменение алгоритма работы или программы реализации на одном из уровней кардинальным образом влияет на производительность более высоких уровней. В случае протокола TCP, от алгоритмов, которые используют маршрутизаторы для обработки дейтаграмм, в значительной степени зависит как производительность одного TCP-соединения, так и суммарная пропускная способность всех соединений. Например, если при обработке одних дейтаграмм в маршрутизаторе будет возникать большая задержка, чем при обработке других<sup>12</sup>, то это приведет к увеличению тайм-аута повторной передачи протокола TCP. Если эта задержка превысит величину тайм-аута, модуль протокола TCP “решит”, что в сети возникла перегрузка. Поэтому, несмотря на то, что стандарты протоколов каждого уровня определены независимо от стандартов остальных уровней, разработчики попытались продумать и реализовать механизм взаимодействия между протоколами разных уровней.

Одно из основных взаимодействий между модулями протоколов IP и TCP происходит в случае перегрузки маршрутизатора, следствием которой является потеря дейтаграмм. Поскольку маршрутизатор помещает каждую вновь прибывшую дейтаграмму в очередь на обработку, находящуюся в оперативной памяти, основное внимание инженеров было сосредоточено на алгоритмах манипуляции элементами этой очереди. Если скорость поступления дейтаграмм превышает скорость их об-

<sup>12</sup> Формально разброс значений задержек называют *дрожжанием (jitter)*.

работки в маршрутизаторе, размер очереди будет постоянно увеличиваться. Если же маршрутизатор перенаправляет дейтаграммы быстрее, чем они поступают, размер очереди сокращается. Поскольку объем оперативной памяти в маршрутизаторе ограничен, размер очереди не может увеличиваться до бесконечности. Поэтому в ранних версиях программного обеспечения для маршрутизаторов при переполнении очереди использовалась стратегия *усечения ее хвоста* (*tail-drop*).

*Стратегия усечения хвоста очереди заключается в том, что в случае переполнения входной очереди маршрутизатора, все вновь прибывающие дейтаграммы аннулируются.*

Название *усечение хвоста* происходит от эффекта, возникающего при обработке последовательности прибывающих дейтаграмм в маршрутизаторе. При переполнении очереди маршрутизатор начинает аннулировать все вновь поступающие дейтаграммы. Другими словами, маршрутизатор усекает “хвост” последовательности прибывающих дейтаграмм.

Усечение хвоста очереди оказывает необычный эффект на работу протокола TCP. В простейшем случае, когда в проходящих через маршрутизатор дейтаграммах содержатся сегменты, относящиеся к одному TCP-соединению, потеря дейтаграмм заставляет модуль протокола TCP применять методику медленного запуска, о которой шла речь выше. В результате производительность TCP-соединения падает до тех пор, пока не начнут приходить сигналы подтверждения приема и увеличиваться размер окна перегрузки. Если через маршрутизатор проходят дейтаграммы, относящиеся к разным TCP-соединениям, применение методики усечения хвоста очереди приводит к более серьезным проблемам. Так происходит потому, что аннулирование последних прибывающих дейтаграмм может привести к эффекту глобальной синхронизации. Чтобы понять суть проблемы, следует отметить, что обычно дейтаграммы поступают на маршрутизатор вперемешку (т.е. друг за другом могут быть получены дейтаграммы от разных отправителей). Поэтому усечение хвоста очереди в этом случае вызывает потерю одного сегмента в каждом из  $N$  соединений, а не потерю  $N$  сегментов, относящихся к одному соединению. Потеря сегментов во всех  $N$  соединениях заставляет модули протоколов TCP этих соединений одновременно переходить к медленному запуску.

## 13.22. Произвольное раннее обнаружение (RED)

Как же избежать эффекта глобальной синхронизации в маршрутизаторе? Для этого необходим алгоритм, который позволит избежать усечения хвоста очереди там, где это возможно. И такой алгоритм был найден. Его называют по разному: *произвольное раннее аннулирование* (*Random Early Discard*), *произвольное раннее удаление* (*Random Early Drop*), или *произвольное раннее обнаружение* (*Random Early Detection*). Но чаще всего используют сокращение *RED*. В маршрутизаторе, где используется методика RED, задаются два пороговых значения, относящихся к положению дейтаграммы в очереди:  $T_{min}$  и  $T_{max}$ . В общих чертах работу алгоритма RED можно описать тремя правилами, с помощью которых определяется положение в очереди каждой из вновь прибывающих дейтаграмм.

- Если в настоящий момент в очереди находится меньше, чем  $T_{min}$  дейтаграмм, новая дейтаграмм просто добавляется в конец очереди.
- Если в очереди находится больше, чем  $T_{max}$  дейтаграмм, то все новые дейтаграммы аннулируются.
- Если число дейтаграмм в очереди колеблется между  $T_{min}$  и  $T_{max}$ , то аннулируется одна из поступивших дейтаграмм, выбранная случайным образом в соответствии с заданной долей вероятности  $p$ .

Стратегия случайного выбора, заложенная в RED, позволяет маршрутизатору по мере увеличения перегрузки переходить к постепенному и случайному удалению дейтаграмм. В этом ее коренное отличие от алгоритма усечения хвоста очереди, при использовании которого в случае переполнения очереди большое количество TCP-соединений переводится в состояние медленного запуска. Таким образом, можно повести некоторый итог.

*Суть метода RED, используемого в маршрутизаторах состоит в следующем. Если входная очередь дейтаграмм переполнена, то все вновь поступающие дейтаграммы аннулируются. Если же входная очередь не заполнена до конца, но ее размер превышает заранее установленный минимальный порог, то во избежание эффекта глобальной синхронизации, аннулируется одна из поступивших дейтаграмм, выбранная случаем образом в соответствии с заданной долей вероятности  $p$ .*

Ключом к эффективной работе метода RED является правильный выбор пороговых значений  $T_{min}$  и  $T_{max}$ , а также величины вероятности потери  $p$ . Значение  $T_{min}$  должно быть достаточно большим, чтобы обеспечить высокую пропускную способность выходного канала связи. Кроме того, поскольку в случае, когда размер очереди превосходит  $T_{max}$ , алгоритм RED работает также как и алгоритм усечения хвоста очереди, значение  $T_{max}$  должно быть больше  $T_{min}$  по крайней мере на среднюю величину увеличения размера очереди за время одной полной доставки TCP-сегмента (т.е. значение  $T_{max}$  должно быть по меньшей мере в два раза больше, чем  $T_{min}$ ). В противном случае использование методики RED может привести к тем же глобальным колебаниям трафика в сети, что и при использовании метода усечения хвоста очереди.

Самой сложной задачей в алгоритме RED является нахождение величины вероятности потери дейтаграммы  $p$ . Очевидно, что величина  $p$  не может быть постоянной и должна вычисляться заново для каждой из вновь прибывших дейтаграмм. Ее значение зависит от текущего размера очереди и установленных пороговых значений  $T_{min}$  и  $T_{max}$ . Идея будет понятнее, если описать алгоритм, используемый в RED, с вероятностной точки зрения. Когда размер очереди меньше  $T_{min}$ , потери дейтаграмм не происходит. Поэтому можно считать, что в этом случае значение  $p$  равно нулю. Когда же размер очереди превышает  $T_{max}$ , все дейтаграммы теряются. Поэтому в последнем случае значение  $p$  равно единице. Если размер очереди колеблется между  $T_{min}$  и  $T_{max}$ , значение вероятности потери дейтаграммы  $p$  можно изменять по линейному закону от 0 до 1.

Описанный выше линейный алгоритм вполне можно положить в основу используемого в RED метода вычисления вероятности потери дейтаграммы  $p$ . Однако в него следует внести изменения для того, чтобы система не так остро реагировала на изменение загрузки сети. Это вызвано тем, что в реально действующей сети значение сетевого трафика все время колеблется в очень широких пределах. В результате размер внутренней очереди маршрутизатора также колеблется с очень высокой скоростью. Если в RED используется упрощенный линейный метод, то вероятность потери последних поступивших в одном блоке, вызвавшем перегрузку, дейтаграмм будет очень высокой. Так происходит потому, что дейтаграммы поступают на маршрутизатор в тот момент, когда размер его внутренней очереди становится очень большим. Как бы то ни было, маршрутизатор не должен без крайней необходимости аннулировать полученные дейтаграммы, поскольку это отрицательно сказывается на производительности TCP-соединения. Поэтому при возникновении кратковременной перегрузки в сети неизвестно удалять дейтаграммы, поскольку внутренняя очередь маршрутизатора никогда не переполнится. С другой стороны, нельзя бесконечно долго оттягивать процесс удаления дейтаграмм, поскольку в случае возникновения затяжной пе-

регрузки очередь переполнится очень быстро. Тогда пропадет весь эффект от использования алгоритма RED, и наверняка возникнут проблемы с глобальной синхронизацией, описанные выше.

Как же назначить в алгоритме RED более высокую вероятность потери  $p$  по мере увеличении размера внутренней очереди и при этом сделать так, чтобы без крайней необходимости не аннулировались дейтаграммы в блоках, вызвавших перегрузку? Для ответа на этот вопрос необходимо вспомнить метод, использовавшийся в протоколе TCP. Суть его состоит в том, что при вычислениях в каждый момент времени не стоит использовать реальный размер очереди. Вместо этого, в алгоритме RED определяется средневзвешенное значение размера внутренней очереди  $avg$ , которое используется для вычисления вероятности потери дейтаграмм. Величина средневзвешенного значения  $avg$  изменяется по экспоненциальному закону и корректируется каждый раз при получении очередной дейтаграммы. При этом используется следующая формула:

$$avg = (1 - \gamma) \times \text{Старое\_avg} + \gamma \times \text{Текущий\_размер\_очереди}$$

В этой формуле значение коэффициента  $\gamma$  находится в интервале между 0 и 1. Если значение  $\gamma$  достаточно мало, средневзвешенное значение  $avg$  будет отслеживать тенденции, происходящие на большом интервале времени. При этом оно будет маловосприимчиво к кратковременным перегрузкам сети<sup>13</sup>.

Кроме формулы, по которой определяется значение  $\gamma$ , в алгоритме RED есть и другие тонкости, в которые мы не будем вдаваться. Например, процесс вычислений можно сделать чрезвычайно эффективным, если выбрать значения чисел кратными  $2^n$  и использовать целочисленную арифметику. Еще одна тонкость состоит в способе измерения размера очереди, который влияет как на процесс вычисления по алгоритму RED, так и на общую производительность протокола TCP. В частности, поскольку время, необходимое для перенаправления дейтаграммы, прямо пропорционально ее длине, то имеет смысл измерять размер очереди не в количестве находящихся в ней дейтаграмм, а в октетах. Тогда в формулы, по которым определяется значение коэффициентов  $p$  и  $\gamma$ , нужно внести небольшие изменения. Измерение размера очереди в октетах влияет также на тип трафика, который будет аннулироваться при перегрузке сети. Все дело в том, что в этом случае вероятность потери будет пропорциональна количеству данных, а не количеству сегментов, помещенных отправителем в поток. Поэтому, чем меньше размер дейтаграммы, тем ниже будет вероятность ее потери. Следовательно, небольшие дейтаграммы (например те, в которых находятся данные сеанса регистрации на удаленном компьютере или запросы к серверу) будут проходить к получателю практически без потерь, тогда как дейтаграммы большого размера (например те, в которых находятся данные передаваемого файла) с высокой долей вероятности будут аннулироваться. Из этого следует положительный вывод: при прохождении сигналов подтверждения приема по перегруженному маршруту вероятность их потери будет низкой. Если будет получен сегмент данных (большого размера), то отправитель практически всегда получит сигнал подтверждения его приема, что позволит избежать ненужной повторной передачи этого сегмента.

В результате моделирования работы алгоритма RED и исследования его поведения в реальных условиях было выявлено много преимуществ этого метода. При перегрузке сети он позволяет избежать эффекта глобальной синхронизации, который возникает при применении метода усечения хвоста очереди. Кроме того, при кратковременной перегрузке метод RED без крайней необходимости не аннулирует дейтаграммы. В настоящее время группа IETF рекомендует использовать в программном обеспечении маршрутизаторов метод RED.

<sup>13</sup> Для примера можно положить  $\gamma = 0,002$ .

### 13.23. Установка TCP-соединения

Для установки соединения в протоколе TCP используется *трехэтапный метод квитирования* (*three-way handshake*). В простейшем случае процесс квитирования происходит так, как показано на рис. 13.12.

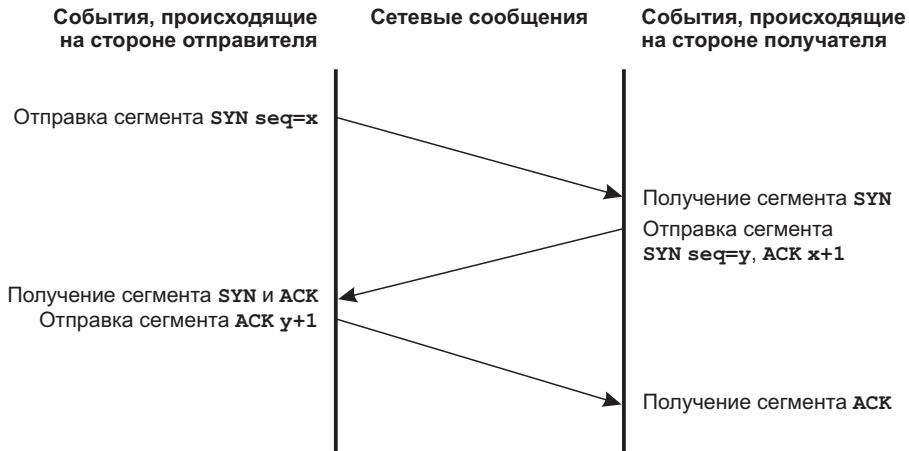


Рис. 13.12. Последовательность сообщений, посылаемых во время трехэтапного метода квитирования. Ось времени отложена по вертикали и направлена вниз. Диагональные линии отображают процесс пересылки сегментов между узлами сети. В сегменте типа SYN пересыпается начальный порядковый номер

Первый сегмент, посылаемый в процессе квитирования, легко распознать, поскольку в поле кода сегмента его заголовка установлен бит SYN<sup>14</sup>. В заголовке второго сегмента устанавливаются сразу два бита: SYN и ACK. Это означает, что в нем находится сигнал подтверждения приема первого сегмента SYN и данные, предназначенные для продолжения процесса квитирования. В заголовке последнего сегмента устанавливается только бит ACK. Этот сегмент используется только для информирования получателя о том, что обе стороны уведомлены об установке соединения.

Обычно инициатором установки соединения выступает модуль протокола, запущенный на одной из машин. При этом модуль, запущенный на другой машине, просто ожидает момента начала процесса квитирования. Тем не менее при квитировании предусмотрена ситуация, когда обе машины одновременно выступят инициаторами начала соединения. Таким образом, TCP-соединение может быть установлено как по инициативе одной из сторон, так и двух сторон сразу. После установки соединения данные могут течь в обоих направлениях совершенно одинаково. Другими словами, обе стороны TCP-соединения являются равноправными, при этом нет ни основного, ни подчиненного устройства.

Трехэтапный метод квитирования является необходимым и достаточным условием выполнения успешной синхронизации между двумя сторонами соединения. Чтобы понять это, необходимо вспомнить, что в протоколе TCP используется ненадежная служба доставки пакетов. А это означает, что сообщения могут быть потеряны, задержаны, продублированы или доставлены с нарушением порядка их следования. Для решения перечисленных проблем в протоколе TCP

<sup>14</sup> Название бита SYN происходит от английского слова “synchronization”, т.е. синхронизация и произносится “син”.

используется механизм повторной передачи утерянных пакетов, который задействуется после истечения времени ожидания сигналов подтверждения приема. Серьезная проблема возникает, если в процессе установки соединения получателю приходят два запроса (первоначальный и повторный) либо когда повторный запрос приходит с большой задержкой (т.е. после того, как соединение было установлено, использовано и разорвано). Перечисленные выше проблемы и позволяет решить метод трехэтапного квитирования. Кроме того, в протоколе TCP принято правило, согласно которому после установки соединения все дополнительные запросы на его установку попросту игнорируются.

### 13.24. Начальный порядковый номер

Трехэтапный метод квитирования выполняет две важные функции. Во-первых, он гарантирует, что обе стороны соединения будут готовы к приему данных и что о готовности одной стороны будет знать другая сторона. Во-вторых, с его помощью выполняется процесс согласования начального порядкового номера. Во время квитирования обе стороны соединения обмениваются начальными порядковыми номерами и ожидают подтверждения их приема. Порядковые номера используются для идентификации потоков данных, посылаемых каждой из сторон открытого соединения. Они выбираются сторонами самостоятельно (обычно случайным образом) во время открытия соединения. Порядковые номера не могут всегда начинаться с одного и того же значения. В частности, модуль протокола TCP не может каждый раз при создании нового соединения назначать порядковый номер, равный 1 (в конце главы в одном из упражнений читателю предлагается самостоятельно ответить на вопрос, почему именно не может). Естественно, что при открытии соединения обе стороны должны согласовать свои порядковые номера. Это делается для того, чтобы номера октетов, указываемые в сигналах подтверждения приема, соответствовали номерам, используемым в заголовках при передаче сегментов данных.

Чтобы понять, как стороны, обменявшиеся всего тремя сообщениями, могут согласовать порядковые номера для двух потоков, необходимо вспомнить, что в заголовке любого сегмента содержится как поле порядкового номера, так и поле номера сигнала подтверждения приема. Предположим, что машина *A* является инициатором соединения. Тогда во время трехэтапного процесса квитирования в первом SYN-сегменте она передает второй стороне (т.е. машине *B*) свой начальный порядковый номер *x*. Получив SYN-сегмент, машина *B* сохраняет в памяти начальный порядковый номер машины *A*, после чего посыпает ей ответный сегмент. В одноименном поле его заголовка она указывает уже свой начальный порядковый номер *y*, а в поле номера сигнала подтверждения приема — значение *x+1*. Тем самым машина *B* уведомляет машину *A*, что ее начальный порядковый номер успешно получен, и что машина *B* ожидает получить от машины *A* поток октетов, начинающийся с номера *x+1*. В третьем (последнем) сегменте процесса квитирования машина *A* подтверждает получение порядкового номера машины *B* и сообщает ей, что она ожидает получить от машины *B* поток октетов, начинающийся с номера *y+1*. В обоих случаях номера в сигналах подтверждения приема соответствуют принятому в протоколе TCP соглашению о *следующем* ожидаемом сегменте.

Выше было описано, как в протоколе TCP выполняется процесс трехэтапного квитирования с помощью обмена сегментами, содержащими минимальное количество информации. Протокол TCP спроектирован так, что в нем возможна передача данных вместе с начальным порядковым номером непосредственно в сегментах квитирования. В подобных случаях модуль протокола TCP должен заблокировать данные до завершения процесса квитирования. После того как

соединение установлено, модуль протокола TCP может разблокировать данные и доставить их очень быстро ожидающему приложению. Подробнее об этом можно прочитать в спецификации протокола TCP.

### 13.25. Закрытие TCP-соединения

Две взаимодействующие по протоколу TCP программы могут корректно завершить сеанс соединения с помощью так называемой операции *закрытия* (*close*). В протоколе для этой цели используется модифицированный метод трехэтапного квитирования. Напомним, что TCP-соединение является дуплексным. Поэтому его можно использовать для передачи двух независимых потоков данных в противоположных направлениях. Получив от прикладной программы сообщение о том, что все данные переданы, модуль протокола TCP закрывает соединение *в одном направлении*. Перед тем как закрыть половину соединения, модуль протокола должен отправить получателю все находящиеся в буфере данные и подождать подтверждения их приема. После этого получателю отправляется сегмент, в заголовке которого устанавливается бит FIN. После того, как FIN-сегмент достигнет машины получателя, ее модуль протокола посыпает отправителю сигнал подтверждения приема и уведомляет свою прикладную программу о том, что отправителем переданы все данные. Для этой цели могут использоваться стандартные средства операционной системы, например устанавливаться признак конца файла при очередной попытке считать данные из открытого соединения.

Закрыв соединение в заданном направлении, модуль протокола TCP отвергает все попытки прикладной программы передать данные в этом направлении. Тем не менее в противоположном направлении данные могут передаваться до тех пор, пока отправитель не закроет вторую половину соединения. Естественно, что даже после закрытия первой половины соединения, по ней все равно будут передаваться отправителю сигналы подтверждения приема. После закрытия соединения в обоих направлениях модули протокола TCP машин отправителя и получателя удаляют из своих системных таблиц записи об этом соединении.

Следует заметить, что на самом деле закрытие соединения происходит несколько сложнее, чем было описано выше. Все дело в том, что для этой цели в протоколе TCP используется модифицированный метод трехэтапного квитирования, работа которого показана на рис. 13.13.

Отличие модифицированного трехэтапного квитирования, используемого для разрыва соединения, от аналогичного метода, применяемого для установки соединения, состоит в следующем. После получения первого FIN-сегмента второй FIN-сегмент посыпается отправителю не сразу (в отличие от SYN-сегмента). Отправителю посыпается сигнал подтверждения приема первого FIN-сегмента, после чего запущенная на машине получателя прикладная программа уведомляется о получении запроса на закрытие соединения. От момента уведомления прикладной программы до получения от нее отклика может пройти довольно много времени (например, если для закрытия соединения требуется вмешательство пользователя программы). Отправка сигнала подтверждения приема первого FIN-сегмента позволяет исключить повторную передачу этого сегмента отправителем по истечении тайм-аута. Получив от прикладной программы, запущенной на машине получателя, команду на закрытие соединения, модуль протокола TCP отошлет отправителю второй FIN-сегмент. При этом отправитель должен прислать уведомление о получении этого сегмента (третий ACK-сегмент).

**Печатай файл этой страницы отдельно**

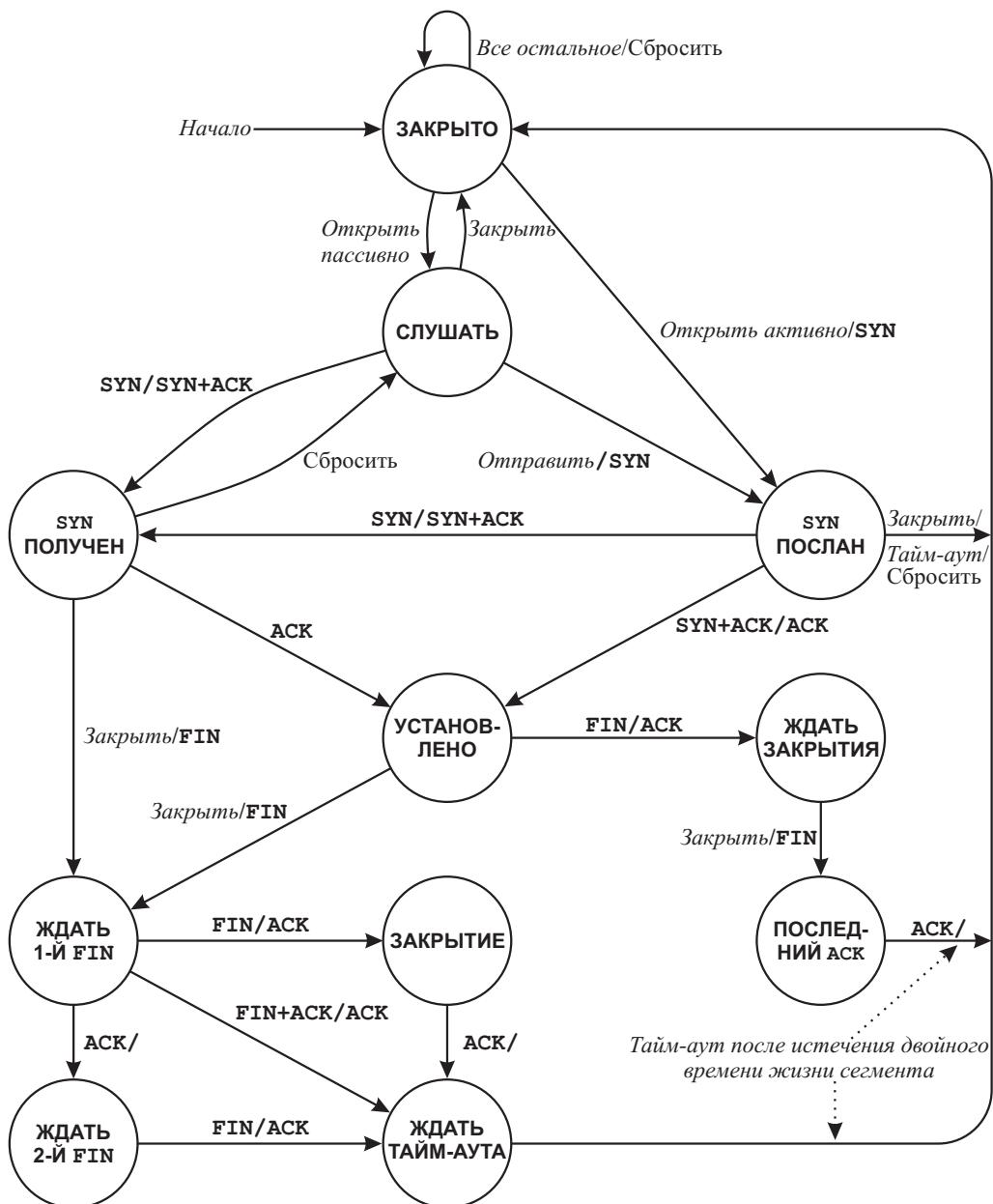


Рис. 13.14. Модель конечного автомата протокола TCP. В исходном состоянии каждая из сторон соединения находится в закрытом состоянии. Надписи на стрелках перехода соответствуют входным данным, которые вызывают изменение состояние системы, а также выходным данным, которые генерирует система (при наличии)

перехода системы в другое состояние и что модуль протокола должен отослать в ответ. Например, рассмотрение функционирования программ протокола TCP, находящихся на каждой из сторон соединения, начинается с состояния, которое называется **ЗАКРЫТИЕМ**. Для вывода системы из этого состояния одна из при-

кладных программ должна выполнить команду *пассивного открытия* соединения (если она собирается ожидать, пока другая машина установит с ней соединение) или команду *активного открытия* соединения (если нужно инициировать соединение). Команда активного открытия соединения вызывает переход системы из **ЗАКРЫТОГО** состояния в состояние сигнал *SYN ПОСЛАН*. Перейдя в это состояние, модуль протокола TCP отправляет получателю SYN-сегмент. Как только от противоположной стороны будет получен сегмент, в заголовке которого установлены биты SYN и ACK, модуль протокола TCP переходит в состояние **УСТАНОВЛЕНО** и начинает передачу данных.

Состояние **ОЖИДАНИЯ ТАЙМ-АУТА** показывает, как в протоколе TCP обрабатываются ситуации, связанные с ненадежной доставкой. Для этой цели в нем предусмотрена специальная переменная, называемая *максимальным временем жизни сегмента* (*maximum segment lifetime*, или *MSL*). Это время определяет, как долго сегмент может находиться в объединенной сети. Протокол TCP переходит в состояние **ОЖИДАНИЯ ТАЙМ-АУТА** сразу после закрытия соединения. Это сделано для того, чтобы сегменты, относящиеся к предыдущему соединению, не влияли на текущее соединение. В таком состоянии протокол TCP должен находиться определенное время, которое равно удвоенному значению максимального времени жизни сегмента, после чего из системных таблиц удаляется запись, относящаяся к текущему соединению. Если во время ожидания тайм-аута к получателю поступит один из сегментов-дубликатов, относящихся к текущему соединению, он будет проигнорирован. Однако, чтобы обработать ситуацию возможной потери последнего сигнала подтверждения, модуль протокола TCP отправляет сигналы подтверждения на все успешно полученные сегменты и перезапускает таймер. Поскольку таймер предназначен для различия сегментов, относящихся к старым и новым соединениям, он не позволяет модулю протокола TCP отвечать RST-сегментами на посланные повторно FIN-запросы.

### 13.28. Принудительная доставка данных

Выше уже шла речь о том, что в протоколе TCP деление потока данных на сегменты, предназначенные для передачи по сети, происходит без учета количества данных, переданных модулю протокола прикладной программой. Разработчики поступили так из соображений эффективности. Это позволяет модулю протокола накапливать во внутреннем буфере необходимое для эффективной передачи количество данных. Благодаря этому существенно снижаются накладные расходы при передаче по сети сегментов небольшого размера.

Несмотря на то что буферизация повышает пропускную способность сети, она может повлиять на работу некоторых прикладных программ. Давайте рассмотрим процесс передачи символов с клавиатуры терминала на удаленный компьютер по установленному TCP-соединению. Очевидно, что пользователь хотел бы увидеть "мгновенную" реакцию удаленного компьютера на нажатие любой клавиши терминала. Поэтому, если модуль протокола TCP перед отправкой символов помещает их в буфер, удаленный компьютер будет реагировать на каждое нажатие клавиши с большой задержкой (вполне возможно, что удаленный компьютер отреагирует только после нажатия нескольких сотен клавиш, причем на все сразу). Точно так же модуль протокола TCP, запущенный на машине получателя, выполняет буферизацию полученных по сети данных, прежде чем доставить их прикладной программе. Очевидно, что для своевременной доставки данных прикладной программе получателя, недостаточно заставить отправителя отослать данные по сети.

Чтобы обеспечить работу интерактивных программ, в протоколе TCP предусмотрена специальная команда *принудительной отправки данных* (push). Ею могут воспользоваться прикладные программы для немедленной (без задержки,

вызванной заполнением выходного буфера) передачи октетов данных, помещенных в выходной поток. Однако команда принудительной отправки вызывает не только немедленную передачу сегмента данных модулем протокола TCP. Она предписывает модулю протокола TCP также установить в поле кода сегмента бит PSH. Эта операция гарантирует, что данные будут доставлены без задержек прикладной программе, запущенной на машине получателя. Таким образом, работая с интерактивным терминалом, прикладная программа должна при получении символа нажатой клавиши вызывать команду принудительной отправки данных. По аналогии, чтобы пользователь без задержки смог увидеть на экране терминала результат нажатия на клавишу, прикладная программа, запущенная на машине получателя, также должна вызвать команду принудительной отправки данных после помещения в выходной поток отдельного символа или целой строки текста.

### 13.29. Зарезервированные номера портов протокола TCP

Как и в протоколе UDP, в TCP используется статическое и динамическое назначение номеров портов. При этом для часто используемых программ, таких как, например, программы работы с электронной почтой, назначаются фиксированные номера портов, называемые *широко известными (well-known)*, или *стандартными*. Оставшаяся большая часть номеров портов поступает в распоряжение операционной системы, которая распределяет их по мере надобности приложениям. Хотя в первоначальном варианте стандарта TCP для стандартных портов резервировались номера меньше 256, в настоящее время можно встретить номера широкопопулярных портов, большие чем 1024. Список наиболее распространенных номеров стандартных портов протокола TCP приведен в табл. 13.2. Следует отметить, что хотя порты протоколов UDP и TCP являются независимыми, разработчики назначили одинаковые номера портов для тех служб, к которым можно обратиться как через протокол UDP, так и TCP. Например, к серверу доменных имен можно обращаться с помощью любого из этих двух протоколов. Поэтому в протоколах UDP и TCP для него назначен один и тот же порт с номером 53.

**Таблица 13.2. Часто используемые стандартные номера портов протокола TCP**

<i>Номер порта</i>	<i>Идентификатор в Internet</i>	<i>Идентификатор в UNIX</i>	<i>Описание</i>
0			Зарезервирован
1	TCPMUX	—	Мультиплексор протокола TCP
7	ECHO	echo	Эхо
9	DISCARD	discard	Аннулирование
11	USERS	systat	Список активных пользователей
13	DAYTIME	daytime	Текущая дата
15	—	netstat	Программа выдачи состояния сети
17	QUOTE	qotd	Цитата дня
19	CHARGEN	chargen	Генератор символов
20	FTP-DATA	ftp-data	Данные протокола передачи файлов (FTP)
21	FTP	ftp	Протокол передачи файлов (FTP)
22	SSH	ssh	Защищенная оболочка

Окончание табл. 13.2

<b>Номер порта</b>	<b>Идентификатор в Internet</b>	<b>Идентификатор в UNIX</b>	<b>Описание</b>
23	TELNET	telnet	Подключение к терминалу
25	SMTP	smtp	Простой протокол передачи электронной почты (SMTP)
37	TIME	time	Текущее время
43	NICNAME	whois	Информация о пользователе
53	DOMAIN	nameserver	Сервер доменных имен (DNS)
67	BOOTPS	bootps	Сервер BOOTP или DHCP
77	—	rje	Любая приватная служба RJE
79	FINGER	finger	Программа finger
80	WWW	www	Сервер World Wide Web
88	KERBEROS	kerberos	Служба аутентификации Kerberos
95	SUPDUP	supdup	Протокол SUPDUP
101	HOSTNAME	hostnames	Сервер NIC имен хостов
102	ISO-TSAP	iso-tsap	Служба ISO-TSAP
103	X400	x400	Почтовая служба X.400
104	X400-SND	x400-snd	Отправитель почты X.400
110	POP3	pop3	Почтовый протокол версии 3 (POP3)
111	SUNRPC	sunrpc	Дистанционный вызов процедур в системе Sun
113	AUTH	auth	Служба аутентификации
117	UUCP-PATH	uucp-path	Служба путей протокола UUCP
119	NNTP	nntp	Протокол передачи сетевых новостей Usenet (NNTP)
123	NTP	ntp	Протокол синхронизации сетевого времени (NTP)
139	NETBIOS-SSN	—	Сеансовая служба протокола NETBIOS
161	SNMP	snmp	Простой протокол сетевого управления (SNMP)

*Примечание.* В других протоколах, например UDP, для доступа к одноименным службам по возможности назначаются такие же номера портов.

### 13.30. Производительность протокола TCP

Как вы уже наверное заметили, TCP является довольно сложным протоколом. Независимость от сетевого оборудования позволяет использовать его для обмена данными в сетях практически любого типа. Поэтому многие считают, что, поскольку TCP выполняет куда более сложные задачи, чем любой другой транспортный протокол, его программный код должен быть весьма запутан и потому неэффективен. Однако универсализм протокола TCP, о котором мы говорили, никак не сказался на его производительности. Эксперименты, проведенные в Беркли, показали, что протокол TCP может эффективно работать не только в глобальной сети Internet. При связи двух компьютеров по локальной сети

Ethernet с пропускной способностью 10 Мбит/с средняя скорость передачи непрерывного потока данных за достаточно большой отрезок времени достигала 8 Мбит/с<sup>15</sup>. Инженеры компании Cray Research, Inc. показали, что при наличии соответствующего оборудования, пропускная способность протокола TCP может достигать 1 Гбит/с.

### 13.31. Синдром полного окна и короткие пакеты

При разработке протокола TCP было замечено значительное снижение производительности соединения в том случае, если отправляющая и принимающая прикладные программы работают с разными скоростями. Чтобы понять суть проблемы, давайте вспомним, что полученные модулем протокола TCP данные помещаются в буфер. А теперь представим, что произойдет, если прикладная программа будет считывать данные из этого буфера по одному октету. Сразу после установки соединения модуль протокола TCP, запущенный на машине получателя, выделяет буфер размером  $K$  байт. При этом отправителю посыпается анонс, в котором указывается исходный размер приемного буфера. (Напомним, что анонс размера приемного буфера посыпается отправителю в поле размера окна заголовка сегмента, содержащего сигнал подтверждения приема данных.) Если приложение-отправитель генерирует поток данных с очень высокой скоростью, то модуль протокола TCP начнет передачу сегментов с учетом полного размера окна приемного буфера. В конечном итоге отправитель получит анонс, сообщающий, что приемное окно полностью заполнено, означающий, что во входном буфере не осталось свободного места.

Как только приложение-получатель считает из заполненного буфера октет данных, освободится место для получения еще одного октета данных. Выше уже было сказано, что модуль протокола TCP машины получателя должен проинформировать об этом отправителю. Он посыпает отправителю сигнал подтверждения приема, в заголовке которого в поле размера окна указано количество свободного места в приемном буфере. В нашем примере получатель пришлет отправителю анонс на один октет. В ответ на это модуль протокола TCP машины отправителя отошлет получателю сегмент, содержащий один октет данных.

И хотя анонсы окна размером в один октет будут отрабатываться вполне корректно, для полного заполнения приемного буфера отправителю понадобится переслать несколько коротких сегментов данных. При этом модуль протокола TCP машины отправителя должен скомпоновать сегмент, содержащий один октет данных, поместить его в IP-дейтаграмму и передать ее по сети. Когда приложение-получатель считает из входного потока еще один октет данных, будет сгенерирован соответствующий сигнал подтверждения приема, который вызовет передачу отправителем следующего сегмента с одним октетом данных. В результате такого взаимодействия отправителя с получателем система может перейти в стабильное состояние, т.е. модуль протокола TCP будет посыпать каждый октет данных в отдельном сегменте.

Передача коротких сегментов по сети резко снижает ее пропускную способность и приводит к излишней нагрузке на центральные процессоры машин отправителя и получателя. Снижение пропускной способности сети объясняется тем, что при передаче коротких сегментов увеличивается соотношение (длина заголовка)/(длина данных). Т.е. по сути по сети вместо данных передаются одни заголовки. Нагрузка на процессоры повышается из-за того, что как приемной, так и передающей стороне приходится обрабатывать большое количество сегментов.

<sup>15</sup> Оставшиеся 2 Мбит/с пропускной способности локальной сети использовались для передачи заголовков фрейма Ethernet, IP- и TCP-пакетов. Кроме того, при передаче последовательности пакетов часть полосы попросту теряется на межпакетные промежутки.

Модуль протокола TCP машины отправителя должен выделить в памяти место под буфер, сформировать заголовок и вычислить для сегмента контрольную сумму. Программы протокола IP машины отправителя должны инкапсулировать сегмент в дейтаграмму, вычислить контрольную сумму ее заголовка, определить маршрут следования дейтаграммы и передать ее соответствующему модулю сетевого интерфейса. Модуль протокола IP, запущенный на машине получателя, должен проверить контрольную сумму IP-заголовка и передать сегмент на обработку модулю протокола TCP. В свою очередь, модуль протокола TCP должен проверить контрольную сумму сегмента и его порядковый номер, извлечь из сегмента данные и поместить их во входной буфер.

Мы рассмотрели процесс образования коротких сегментов в сети в результате получения анонса на окно небольшого размера. Однако отправитель также может генерировать сегменты, содержащие небольшое количество данных. Например, представьте, что в некоторой реализации протокола TCP данные, как только попадают в выходной буфер, сразу же отправляются получателю. Давайте рассмотрим, что произойдет, если прикладная программа при обращении к протоколу TCP будет передавать за раз один октет данных. После того как приложение генерирует октет данных, модуль протокола TCP поместит его в сегмент и передаст по сети получателю. Модуль протокола TCP может также посыпать в сеть короткие сегменты, если прикладная программа генерирует данные блоками фиксированного размера, равными  $B$  октетов. Если предположить, что размер выходного буфера модуля протокола TCP равен  $M$  октетов, причем  $M \neq B$ , то размер последнего передаваемого сегмента вероятнее всего будет небольшим.

Описанная в этом разделе проблема называется *синдромом полного окна* (*silly window syndrome*, или *SWS*). Она проявлялась в ранних реализациях протокола TCP. Подводя итоги, можно сказать следующее.

*Ранним реализациям протокола TCP была присуща проблема, которая называлась синдромом полного окна. Суть ее состоит в том, что при практически полностью заполненном приемном буфере отправителю посылаются анонсы на окна небольшого размера. В результате отправитель посыпает в сеть сегменты небольшого размера.*

### 13.32. Как избежать синдрома полного окна

В современные реализации протокола TCP включен специальный эвристический алгоритм, позволяющий избежать синдрома полного окна. Он не позволяет отправителю посыпать в сеть несколько сегментов подряд, содержащих небольшое количество данных. Еще один эвристический алгоритм запрещает получателю анонсировать окна небольшого размера в случае, если происходит небольшой прирост их размера. Напомним, что именно анонсирование окон небольшого размера является причиной появления коротких сегментов в сети. Следует отметить, что два упомянутых выше эвристических алгоритма очень хорошо работают не только вместе, но и по отдельности. Это позволяет добиться приемлемой производительности сети в случае, если по какой-либо причине одна из сторон не сможет самостоятельно подавить синдром полного окна.

При создании реально действующего модуля протокола TCP разработчики должны включить в него специальный код, подавляющий синдром полного окна как на стороне отправителя, так и на стороне получателя. Чтобы понять, зачем это нужно, достаточно вспомнить, что протокол TCP является дуплексным, т.е. данные в нем могут течь в обоих направлениях. Поэтому при любой реализации модуля протокола TCP в него должен быть включен как код для отправки данных, так и код для их получения.

### **13.32.1. Борьба с синдромом полного окна на стороне получателя**

Эвристический алгоритм, используемый для подавления синдрома полного окна на стороне получателя, довольно простой, поэтому его легко понять. В двух словах, суть его заключается в том, что получатель отслеживает с помощью специальной переменной текущий размер свободного буфера. При его увеличении анонс окна не посыпается отправителю сразу, а задерживается до того момента, когда освободиться значительная часть приемного буфера. Осталось только выяснить, что вкладывается в понятие “значительная часть”. Все зависит от текущего размера приемного буфера и максимального размера сегмента. В стандарте протокола TCP сказано, что анонс окна посыпается в случае освобождения как минимум половины приемного буфера либо пространства в буфере, достаточного для приема сегмента максимальной длины.

Таким образом, эвристический алгоритм, используемый для подавления синдрома полного окна на стороне получателя, препятствует анонсированию окон небольшого размера в том случае, если прикладная программа считывает медленно принятые октеты данных. Например, при заполнении приемного буфера отправителю будет послан сигнал подтверждения, содержащий анонс окна нулевого размера. По мере того как приложение-получатель считывает октеты из приемного буфера, модуль протокола TCP машины-получателя пересчитывает количество свободного места в этом буфере. Однако при этом анонс окна не посыпается отправителю сразу, а задерживается до момента, пока не освободится половина приемного буфера или его часть, равная максимальной длине сегмента. Следовательно, отправитель будет всегда получать анонсы на окна большого размера, что позволит ему передавать только длинные сегменты. Из всего сказанного выше можно сделать такой вывод.

*Суть эвристического алгоритма, используемого для подавления синдрома полного окна на стороне получателя, состоит в следующем. После отправки анонса на окно нулевого размера, следующий анонс посыпается отправителю только после освобождения не менее половины приемного буфера либо при освобождении в буфере места, достаточного для приема сегмента максимального размера.*

### **13.32.2. Задержка сигналов подтверждения приема**

Для подавления синдрома полного окна на стороне получателя используются два подхода. Первый подход был рассмотрен в предыдущем разделе. Суть его заключается в том, что после получения каждого сегмента, модуль протокола TCP сразу же посыпает отправителю сигнал подтверждения его приема. Однако при этом отправка анонсов на увеличение приемного окна задерживается до момента, пока в приемном буфере не освободится достаточно места, как было сказано выше при описании эвристического алгоритма. При использовании второго подхода модуль протокола TCP не посыпает отправителю сигнал подтверждения приема сегмента, если в приемном буфере недостаточно свободного места. В стандарте протокола TCP рекомендуется использовать второй подход.

Метод задержки сигналов подтверждения приема имеет как преимущества, так и недостатки. Основное преимущество заключается в том, что при задержке подтверждения приема уменьшается трафик в сети, т.е. увеличивается ее общая пропускная способность. Например, если за время задержки будут получены дополнительные сегменты данных, то подтвердить их успешное получение можно с помощью всего одного сегмента. Если же прикладная программа, получив данные, сразу должна генерировать на них ответ (как, например, при выполнении сеанса регистрации на удаленном компьютере, когда на терминал должны посы-

ляться символы, соответствующие нажатым клавишам), введение небольшой задержки позволит поместить эти данные в тот же сегмент, в котором отправителю будет передан сигнал подтверждения приема. Более того, модуль протокола TCP не сможет переместить свое окно в потоке данных до тех пор, пока приложение-получатель не считает эти данные из буфера. Если приложение-получатель считывает данные из буфера сразу после их получения, введение небольшой задержки позволяет модулю протокола TCP в одном сегменте подтвердить прием данных и анонсировать изменение размеров приемного окна. Без введения задержки модуль протокола TCP пошлет отправителю подтверждение приема сразу после получения сегмента данных. По прошествии некоторого времени отправителю посыпается второй сигнал подтверждения приема, в котором указывается анонс на изменение размеров приемного окна.

Недостатки метода задержки сигналов подтверждения приема, должно быть, уже очевидны. Самый существенный из них состоит в том, что если получатель вовремя не отослал сигнал подтверждения приема, отправитель пришлет сегмент повторно. Очевидно, что подобные ненужные повторы существенно снижают пропускную способность сети. Кроме того, повторная отправка сегментов требует дополнительных затрат процессорного времени на машине как отправителя, так и получателя. Следует также напомнить, что в протоколе TCP по времени прибытия сигналов подтверждения приема оценивается полное время доставки сегмента. Поэтому любые задержки этих сигналов приводят к неточности результатов измерений, что влечет за собой увеличение времени тайм-аута повторной передачи.

Чтобы избежать потенциальных проблем, в стандарте протокола TCP установлены предельные значения задержек отправки сигналов подтверждения приема. При реализации модуля протокола TCP максимальная величина задержки не должна превышать 500 миллисекунд. Кроме того, для гарантии достоверности измерений полного времени доставки пакета в стандарте TCP рекомендуется, чтобы получатель сразу же посыпал сигналы подтверждения приема как минимум на каждый второй сегмент данных.

### 13.32.3. Борьба с синдромом полного окна на стороне отправителя

Для подавления синдрома полного окна на стороне отправителя в протоколе TCP используется довольно элегантный эвристический алгоритм. Напомним, что конечная цель — избежать отправки в сеть коротких сегментов. Кроме того, следует иметь в виду, что приложение-отправитель может генерировать данные блоками сколь угодно малого размера (вплоть до одного октета). Поэтому, чтобы достичь конечной цели, модуль протокола TCP машины отправителя должен предоставить прикладной программе возможность многократного вызова функции *записи* в выходной буфер. При этом все переданные программой данные должны собираться в один достаточно большой сегмент и только затем отправляться получателю. Таким образом, модуль протокола TCP должен задерживать отправку сегмента до тех пор, пока его размер не достигнет приемлемых значений. Эта методика называется *накоплением* (*clumping*).

Возникает вопрос: на сколько модуль протокола TCP должен задержать отправку пакета? С одной стороны, при слишком больших задержках возрастает время доставки сегментов прикладной программе-получателю. Хуже всего то, что модуль протокола TCP не может точно определить время задержки, поскольку заранее никогда не известно, будут ли в ближайшем будущем получены какие-либо данные от прикладной программы. С другой стороны, при слишком короткой задержке в сеть посыпаются сегменты небольшого размера, в результате чего падает общая производительность сети.

С подобной проблемой разработчики протоколов столкнулись уже давно (еще до создания протокола TCP). Решалась она путем объединения данных в большие пакеты. Например, для повышения эффективности передачи данных по сети в ранних протоколах работы с терминалом передача каждого введенного с клавиатуры символа задерживалась на несколько сотен миллисекунд. За это время программа могла определить, будет ли пользователь продолжать ввод данных с клавиатуры. Поскольку протокол TCP создавался для универсального применения, его можно приспособить для решения специфических задач. Данные, передаваемые по TCP-соединению, могут быть введены с клавиатуры, прочитаны из файла либо получены в результате работы программы. Очевидно, что фиксированное значение задержки не может быть оптимальным для программ всех типов. Поэтому в протоколе TCP для ее определения используется адаптивный алгоритм, наподобие того, что использовался для определения времени тайм-аута повторной передачи и алгоритма медленного запуска, применяющегося для избежания кратковременных перегрузок сети. Величина задержки зависит от текущей скорости работы объединенной сети. Как и алгоритм медленного запуска, эвристический алгоритм подавления синдрома полного окна на стороне отправителя называется *самосинхронизирующимся* (*self clocking*), поскольку в нем не вычисляются величины задержек. Сигналом к началу передачи дополнительных пакетов служит момент получения сигнала подтверждения приема. Работу эвристического алгоритма можно вкратце описать следующим образом.

*Суть эвристического алгоритма подавления синдрома полного окна на стороне отправителя состоит в следующем. Если по установленному TCP-соединению были посланы данные, но прием их не подтвержден, то новые данные, сгенерированные прикладной программой, помещаются в выходной буфер, но не отправляются получателю. Эти данные будут отправлены только после того, как от получателя поступит анонс на окно, размер которого позволяет принять сегмент максимальной длины. Если после получения сигнала подтверждения приема в выходном буфере все еще находятся данные, ожидающие отправки, они немедленно посыпаются получателю. Это правило применяется даже в том случае, если от прикладной программы получен запрос на принудительную отправку данных.*

Если при каждом обращении к модулю протокола TCP прикладная программа генерирует по одному октету данных, то первый октет посылается получателю немедленно. Остальные же октеты накапливаются в выходном буфере до тех пор, пока отправитель не получит сигнал подтверждения приема. Поэтому, если скорость потока данных, генерируемых прикладной программой, сравнима с пропускной способностью сети (как, например, при передаче файлов), во всех последующих сегментах будет передаваться много октетов данных. Если же скорость потока данных, генерируемых прикладной программой, крайне мала по сравнению с пропускной способностью сети (как в случае программы ввода данных с клавиатуры пользователя), короткие сегменты будут посыпаться без существенной задержки.

Описанная в этом разделе методика борьбы с синдромом полного окна на стороне отправителя называется *алгоритмом Нейгла* (*Nagle algorithm*) по имени его создателя. Ценность этой методики заключается в том, что для выполнения алгоритма не требуются большие вычисления. Узлу сети не нужно поддерживать работу отдельных таймеров для каждого соединения. При получении данных от прикладной программы модуль протокола не должен опрашивать значение таймера. Однако важнее всего то, что эту методику можно использовать при любых значениях задержек в сети, максимальных размерах сегмента и скорости работы приложения. При нормальных условиях она не снижает пропускную способность сети.

Следует объяснить, почему при нормальных условиях алгоритм Нейгла не снижает пропускную способность сети. Если прикладная программа рассчитана на обработку большого потока данных, она не будет генерировать по одному октету данных за раз, поскольку это приведет к излишней нагрузке на операционную систему. При каждом обращении к модулю протокола подобные программы, передают большой блок данных. Поэтому в выходном буфере всегда будет содержаться достаточное количество данных (по крайней мере не меньше размера одного максимального сегмента). Более того, поскольку прикладные программы обычно генерируют данные гораздо быстрее, чем их может передать модуль протокола TCP, выходной буфер практически всегда полон, и модулю протокола TCP незачем откладывать передачу данных. Если прикладные программы постоянно заполняют выходной буфер данными, модуль протокола TCP отправляет сегменты с максимально возможной для объединенной сети скоростью. Подведем итоги.

*В современные реализации протокола TCP включен эвристический алгоритм подавления синдрома полного окна как на стороне отправителя, так и на стороне получателя. Суть его состоит в том, что получатель старается избежать анонсирования окон небольшого размера, а отправитель использует адаптивный алгоритм для задержки передачи сегментов. Это дает возможность отправителю собрать данные в сегмент большого размера.*

### 13.33. Резюме

В протоколе TCP определены стандарты на одну из основных служб объединенной сети — надежную потоковую транспортную службу. Стандартом протокола TCP предусмотрена установка дуплексного соединения между двумя машинами, что позволяет им эффективно обмениваться большими массивами данных.

Для повышения эффективности использования сети в протоколе TCP используется механизм движущихся окон. Протокол TCP не зависит от низкоуровневой системы доставки пакетов, поэтому его можно использовать в сети практически любого типа. Используемый в протоколе TCP механизм управления передачей позволяет двум компьютерам, работающим с разной скоростью, без проблем обмениваться данными.

Основной единицей передачи данных, используемой в протоколе TCP, является сегмент. Сегменты используются для передачи как данных, так и служебной информации, необходимой для установки и разрыва соединения между двумя машинами. Выбранный формат сегмента позволяет поместить сигнал подтверждения приема для данных, текущих в обратном направлении, в заголовок сегмента данных, передаваемых в одном с сигналом направления.

Управление потоком данных в протоколе TCP происходит на основе сообщений (анонсов) получателя, в которых указывается количество данных, которое он может принять. Кроме того, имеется возможность отправить сегмент “вне очереди”, задействовав механизм передачи срочных данных. Для принудительной передачи данных в протоколе TCP предусмотрен специальный механизм “проталкивания” сегментов.

В современной версии стандарта TCP используется метод экспоненциальной коррекции тайм-аута, позволяющий более точно определить время повторной передачи сегментов. Для предотвращения кратковременных перегрузок в сети в протоколе TCP используются специальные алгоритмы — медленного запуска и мультиплексивного уменьшения. Кроме того, в протоколе TCP предусмотрены специальные меры для подавления передачи коротких пакетов с помощью эвристического алгоритма. Группа IETF рекомендует использовать в современном

программном обеспечении маршрутизаторов алгоритм RED, а не метод усечения хвоста очереди. Благодаря этому удается существенно повысить пропускную способность сети и избежать эффекта глобальной синхронизации в маршрутизаторе.

## Материал для дальнейшего изучения

Стандарт протокола TCP был описан Постелом в [RFC 793]. В [RFC 1122] Браден уточняет этот стандарт и проясняет некоторые моменты. Система управления окнами протокола TCP описана Кларком в [RFC 813]. В [RFC 816] Кларк описывает методику изоляции неисправностей и восстановления работоспособности системы после сбоя. О максимальном размере TCP-сегмента речь идет в [RFC 879], написанном Постелом. В [RFC 896] Нейгл рассматривает процесс возникновения перегрузок в сетях TCP/IP и объясняет эффект самосинхронизации при применении эвристического алгоритма для подавления синдрома полного окна на стороне отправителя. В статье Карна (Karn) и Партриджа (Partridge) [76] обсуждаются методы оценки полного времени доставки пакетов и описан алгоритм Карна. В статье Якобсона (Jacobson) [64] описан алгоритм устранения перегрузки, который является неотъемлемой частью современного стандарта протокола TCP. Алгоритм RED описан в статье Флойда (Floyd) и Якобсона [53], а способы его применения — в статье Кларка и Фанга (Fang) [24]. Более подробно метод трехэтапного квитирования описан в статье Томлинсона (Tomlinson) [129]. Отчет об измерениях полного времени доставки пакетов в Internet составлен Миллсом (Mills) и опубликован в [RFC 889]. Метод устранения перегрузки с помощью движущихся окон, основанный на значениях таймера, описан в статье Джейна (Jain) [66]. Результат эксперимента по применению протокола TCP в высокоскоростных сетях на основе компьютеров Gray описан в статье Бормана (Borman) [13].

## Упражнения

- 13.1. Как известно, поле, содержащее порядковый номер сегмента в потоке данных, имеет конечный размер. Создайте спецификацию протокола, которая позволяла бы передавать от одной машины к другой поток данных произвольного (не ограниченного сверху) размера.
- 13.2. В спецификации протокола TCP указано, что с помощью одного из его параметров получатель может сообщить отправителю максимальный размер сегмента, который он может принять. Почему, несмотря на наличие механизма анонсирования размера приемного окна, в протокол TCP введен этот параметр?
- 13.3. При каких значениях задержки, полосы пропускания и загрузки сети, а также процента потерь пакетов модулю протокола TCP не придется повторно передавать значительное количество данных?
- 13.4. Объясните, почему в протоколе TCP при потере сигнала подтверждения приема не требуется его повторная передача.
- 13.5. Поэкспериментируйте с компьютерами вашей локальной сети и определите, как в протоколе TCP обрабатывается процесс перезагрузки машины. Для этого установите соединение с одним из компьютеров (например, с помощью средств удаленной регистрации) и оставьте его в бездействующем состоянии. Затем инициируйте процесс зависания удаленного компьютера и его последующую перезагрузку. После этого попытайтесь отослать с локальной машины TCP-сегмент (например, попытайтесь что-то ввести с клавиатуры в сеансе удаленной регистрации).

- 13.6.** Предположим, что существует вариант реализации модуля протокола TCP, в котором аннулируются все сегменты, доставленные получателю с нарушением порядка следования (даже если они попадают в текущее окно). Другими словами, вымышленная нами версия реализации протокола принимает только такие сегменты, которые пополняют принятый ранее поток октетов. Будет ли такая реализация работать? Чем она будет отличаться от стандартной реализации протокола TCP?
- 13.7.** Рассмотрим процесс вычисления контрольной суммы TCP-сегмента. Предположим, что перед вычислением поле контрольной суммы заголовка сегмента *не было обнулено*. В результате вычислений контрольная сумма оказалась равной нулю? Какой вывод вы можете сделать?
- 13.8.** Какие вы можете привести аргументы за и против автоматического закрытия бездействующих TCP-соединений?
- 13.9.** Предположим, что двум прикладным программам необходимо обменяться данными по TCP-соединению. Однако специфика их работы такова, что в каждом сегменте они должны пересыпать по одному символу (т.е. после передачи каждого символа модулю протокола они принудительно вызывают его отправку получателю с помощью соответствующей команды “проталкивания”). Определите максимальный процент использования полосы пропускания сети при передаче данных этими программами.
- 13.10.** Предположим, что в некой реализации протокола TCP при установке соединения ему назначается начальный порядковый номер, равный 1. Поясните, почему перезагрузка одного из компьютеров, вызванная сбоем в его работе, может привести к неполадкам в работе модуля протокола TCP удаленного компьютера (т.е. удаленный компьютер будет “ считать ” старое соединение открытым ).
- 13.11.** Ознакомьтесь с алгоритмом измерения полного времени доставки пакетов, предложенным в спецификации протокола ISO TP-4, и сравните его с тем, который используется в протоколе TCP (он был описан в этой главе). Каким из них вы бы предпочли воспользоваться?
- 13.12.** Выясните, как в реализациях протокола TCP должна решаться проблема перекрытия сегментов данных. Эта проблема возникает из-за того, что получатель должен принять только одну копию всех октетов потока данных, хотя отправитель может передать два частично перекрывающихся сегмента (т.е. в первом сегменте передаются байты 100–200 текущего потока данных, а во втором — 150–250).
- 13.13.** Проследите по схеме модели конечного автомата протокола TCP (см. рис. 13.14) все переходы, которые выполняют обе стороны соединения при выполнении операции его пассивного и активного открытия. Обратите внимание на то, как происходит трехэтапное квитирование.
- 13.14.** Ознакомьтесь со спецификацией протокола TCP и опишите условия, при которых конечный автомат может перейти из состояния ожидания 1-го сигнала FIN в состояние ожидания тайм-аута.
- 13.15.** Проследите по схеме модели конечного автомата протокола TCP (см. рис. 13.14) все переходы, которые выполняют обе стороны соединения при поэтапном его закрытии.

- 13.16.** Предположим, что модуль протокола TCP пересыпает сегменты с помощью окна максимального размера (64 Кбайт) по каналу с неограниченной пропускной способностью и средним временем доставки пакетов 20 мс. Какова максимальная пропускная способность такого соединения? Как изменится пропускная способность соединения, если время доставки пакетов возрастет до 40 мс (при условии, что пропускная способность сети не ограничена)?
- 13.17.** На примере предыдущего упражнения было показано, что для повышения пропускной способности TCP-соединения нужно увеличивать размер окна. Одним из недостатков формата TCP-сегмента является то, что под анонс окна выделено поле ограниченного размера. Как следует изменить спецификацию протокола TCP, чтобы можно было использовать окна большего, чем 64 Кбайт, размера без изменения формата сегмента?
- 13.18.** Выведите формулу, по которой можно вычислить максимально возможную пропускную способность TCP-соединения в зависимости от пропускной способности сети, времени передачи пакета, времени обработки сегмента и генерирования сигнала подтверждения приема. (*Подсказка.* Воспользуйтесь предыдущим упражнением.)
- 13.19.** Опишите нестандартные условия в сети, при которых одна из сторон соединения может оставаться неопределенным долгое время в состоянии ожидания второго сигнала FIN. (*Подсказка.* Рассмотрите случай потери дейтаграмм в результате зависания системы.)
- 13.20.** Покажите, что при использовании алгоритма RED в маршрутизаторе вероятность потери пакета, относящегося к заданному TCP-соединению, прямо пропорциональна количеству трафика (выраженному в процентах), генерируемого данным соединением.

# 14

## Основы маршрутизации

### 14.1. Введение

В предыдущих главах внимание уделялось службам сетевого уровня семейства протоколов TCP/IP, а также особенностям протоколов, которые используются в узлах сети и маршрутизаторах, предоставляющих такие услуги. При обсуждении этих вопросов мы предположили, что маршрутизаторы всегда содержат только правильную информацию о маршрутах, а также отметили, что маршрутизаторы могут заставить узлы, непосредственно подключенные к одной с ними сети, изменить маршруты посредством механизма перенаправления протокола ICMP.

В этой главе рассматриваются два обширных вопроса: какие значения должна содержать таблица маршрутизации и каким образом можно получить эти значения? Чтобы ответить на первый вопрос, рассмотрим взаимосвязь между структурой объединенной сети и механизмом маршрутизации. В частности, мы рассмотрим принципы объединения сети на основе одной широкополосной магистрали и нескольких равноправных сетевых магистралей. Затем мы обсудим необходимые условия для осуществления маршрутизации. Поскольку большинство примеров из этой главы взяты из глобальной сети Internet, рассматриваемые здесь вопросы касаются также небольших корпоративных объединенных сетей. Чтобы получить ответ на второй вопрос, мы рассмотрим два основных вида алгоритмов распространения маршрутной информации, а также увидим, каким образом каждый из них автоматически передает маршрутную информацию.

В начале мы рассмотрим, что представляет собой процесс маршрутизации в общем смысле этого слова. В последующих разделах мы уделим внимание структуре объединенной сети и алгоритмам, которые используются маршрутизаторами для обмена маршрутной информацией. В главах 15, “Маршрутизация между автономными системами (BGP)”, и 16, “Маршрутизация внутри автономной системы (RIP, OSPF, HELLO)”, продолжается более углубленное обсуждение методов маршрутизации. В них описываются протоколы, используемые маршрутизаторами, которые принадлежат к двум независимым административным группам, для обмена маршрутной информацией, а также протоколы, которые используются на всех внутренних маршрутизаторах группы.

### 14.2. Понятие о таблицах маршрутизации

В главе 3, “Основы и структура межсетевого взаимодействия”, мы уже говорили, что IP-маршрутизаторы обеспечивают активные взаимосвязи между сетями. Каждый маршрутизатор подключается к двум или более физическим сетям и выполняет пересылку IP-дейтаграмм между ними. Он получает дейтаграммы,

поступающие через один сетевой интерфейс, и переправляет их через другой. За исключением получателей, которые непосредственно подключены к тем же сетям, что и маршрутизатор, узлы передают весь свой информационный поток маршрутизаторам, используя для этого протокол IP. Последние, в свою очередь, переправляют дейтаграммы далее по маршруту к конечным пунктам назначения. Дейтаграмма переходит от одного маршрутизатора к другому до тех пор, пока не достигнет маршрутизатора, непосредственно подключенного к той же сети, что и ее конечный получатель. Таким образом, система маршрутизаторов создает структурную основу объединенной сети и управляет всем трафиком, за исключением непосредственной доставки пакетов от одного узла до другого.

В главе 8, “Протокол IP: маршрутизация дейтаграмм”, описан алгоритм маршрутизации протокола IP, которым руководствуются узлы сети и маршрутизаторы, перенаправляющие дейтаграммы. В ней также демонстрируется, как в этом алгоритме используются таблицы маршрутизации для выбора маршрута следования дейтаграмм. В каждом элементе таблицы маршрутизации указывается сетевая часть IP-адреса получателя и адрес следующей машины, расположенной по маршруту следования пакетов до сети получателя. Подобно узлам сети, маршрутизаторы доставляют дейтаграммы получателям, которые непосредственно подключены к тем же сетям, что и маршрутизатор.

Рассматривая основы передачи дейтаграмм, мы не выяснили, каким образом узлы сети и маршрутизаторы получают информацию для своих таблиц маршрутизации. Этот вопрос имеет два аспекта: *какие именно* значения должны быть занесены в таблицу и *как* маршрутизаторы получают эти значения? Оба аспекта зависят от структуры и размера объединенной сети, а также принципов управления ею.

Создание системы маршрутизации происходит в два этапа: 1) инициализация и 2) обновление. При включении питания каждый маршрутизатор должен установить исходный набор маршрутов. По мере изменения маршрутов маршрутизатор должен обновлять свою таблицу (например, если происходит отказ одного из сетевых интерфейсов). Этап инициализации зависит от типа используемой операционной системы. В некоторых системах маршрутизатор считывает при запуске исходную таблицу маршрутизации из внешнего запоминающего устройства и сохраняет ее в своей оперативной памяти. В других системах работа маршрутизатора начинается с пустой таблицы, которую необходимо заполнить, выполняя определенные команды (например, команды, находящиеся в сценарии начальной загрузки). И, наконец, некоторые операционные системы начинают свою работу с того, что логически определяют исходный набор маршрутов из набора адресов локальных сетей, к которым подключена машина, и затем связываются с соседней машиной с целью запроса дополнительных маршрутов.

Когда исходная таблица маршрутизации создана, маршрутизатор долженнести изменения в маршруты. В небольших, редко изменяющихся сетях администраторы могут устанавливать и модифицировать маршруты вручную. В крупных, быстро изменяющихся средах ручное обновление происходит слишком медленно и чревато ошибками. В этом случае необходимо применять автоматические методы обновления.

Для того чтобы понять сущность протоколов для автоматического обновления таблиц маршрутизации в IP-маршрутизаторах, необходимо рассмотреть несколько основополагающих понятий. В последующих разделах изложены концептуальные основы процесса маршрутизации. В конце главы обсуждается структура объединенной сети и протоколы, которые используются маршрутизаторами для обмена маршрутной информацией.

### 14.3. Маршрутизация при неполной информации

Существенное отличие между маршрутизатором и типичным узлом сети заключается в том, что узлы, как правило, владеют недостаточной информацией о структуре объединенной сети, к которой они подключены. Узлы не располагают полной информацией о *всех возможных* адресах получателей или *всех возможных* сетях-адресатах. На самом деле, в таблицах маршрутизации многих узлов есть информация только для двух маршрутов: один из них — для локальной сети, а другой — маршрут по умолчанию, или стандартный маршрут, определяющий путь прохождения пакетов к соседнему маршрутизатору. Все дейтаграммы, адресованные во внешний мир, узел сети отсылает локальному маршрутизатору для дальнейшей доставки. Обратите внимание, что

*узел сети может успешно выполнять маршрутизацию дейтаграмм, даже если он располагает частичной информацией о маршрутизации. Причина в том, что узел сети может свободно положиться на маршрутизатор.*

Могут ли маршрутизаторы также успешно выполнять маршрутизацию дейтаграмм, располагая лишь частичной информацией? Да, могут, но только при определенных условиях. Чтобы понять необходимые для этого условия, представим, что объединенная сеть — это чужая страна, которую вдоль и поперек пересекают грунтовые дороги, а на их перекрестках установлены указатели. Представьте, что у вас нет карты, вы не можете спросить, в каком направлении вам необходимо передвигаться, так как не говорите на языке местных жителей, не имеете представления о местности, в которой находитесь. Но вам просто необходимо добраться до селения под названием *Суссекс*. Вы отправляетесь в путешествие, передвигаясь по единственной дороге, ведущей из вашего города, и начинаете искать глазами указатели. На первом указателе написано<sup>1</sup>:

Нортфок — налево; Хаммонд — направо; другие города — прямо.

Поскольку необходимый вам пункт назначения явно не указан, вы продолжаете двигаться прямо. На языке маршрутизации мы назовем маршрут, по которому вы следите, *стандартным маршрутом*, или *маршрутом, принятым по умолчанию*. Через несколько знаков вы, наконец, встречаете указатель, на котором написано:

Эссекс — налево; Суссекс — направо; другие города — прямо.

Вы поворачиваете направо, передвигаетесь по нескольким последующим указателям и, наконец, попадаете на дорогу, ведущую в Суссекс.

Наше воображенное путешествие является полным аналогом прохождения дейтаграммы по объединенной сети, а дорожные знаки — это таблицы маршрутизации, которые находятся в устройствах маршрутизации, расположенных по пути следования дейтаграммы. Без карты или других навигационных приборов путешествие полностью зависит от дорожных знаков, точно так же, как маршрутизация дейтаграммы в объединенной сети полностью зависит от содержащего таблиц маршрутизации. Понятно, что передвижение возможно несмотря на то, что каждый дорожный знак содержит только частичную информацию.

Главный вопрос заключается в достоверности информации. Как путешественник вы можете спросить: “Как я могу быть уверен что, следя по знакам, достигну конечного пункта назначения?” или “Могу ли я быть уверен, что, следя по знакам, достигну пункта назначения кратчайшим путем?” Ответить на эти вопросы особенно трудно, если на пути вам встречается много знаков, в которых явно не указан ваш пункт назначения. Конечно, ответы на поставленные вопросы зависят от топологической схемы дорог, а также содержащейся на знаках информации. Но

---

<sup>1</sup> Предполагается, что указатели написаны на понятном вам языке.

основной смысл заключается в том, что в целом информация, содержащаяся на знаках, должна быть полной и непротиворечивой. С другой стороны, очевидно, что на каждом перекрестке неизбежно должен присутствовать знак, указывающий на любой пункт назначения. На указателях могут перечисляться стандартные маршруты при условии, что на других знаках будет явно указан кратчайший путь к нужному пункту назначения. Кроме того, должны быть обозначены повороты, ведущие ко всем пунктам назначения. Ниже мы на нескольких примерах продемонстрируем, как достичь согласованности информации.

Как крайний случай, рассмотрим простую схему дорог, которая имеет форму звезды. В этой схеме к каждому селению ведет только одна дорога, а все пути к другим населенным пунктам сходятся в одной центральной точке. Чтобы обеспечить непротиворечивость информации, указатель на главном перекрестке должен содержать информацию обо всех возможных пунктах назначения. Есть и другая крайность. Представьте себе произвольный набор дорог, на всех перекрестках которых размещены указатели, содержащие информацию о всех возможных пунктах назначения. Тогда, чтобы обеспечить непротиворечивость информации, на любом перекрестке должны соблюдаться следующие правила: если для пункта назначения  $\Gamma$  указатель показывает на дорогу  $\Delta$ , то только дорога  $\Delta$  ведет к пункту  $\Gamma$  по кратчайшему пути.

Ни одна из описанных схем не работает хорошо в системе маршрутизации объединенной сети. С одной стороны, первый, приведенный выше подход к решению проблемы, неэффективен, потому что ни одна машина не может выполнять роль центрального коммутатора, через который проходит весь сетевой трафик. С другой стороны, держать на всех маршрутизаторах информацию о маршрутах ко всем возможным пунктам назначения непрактично, поскольку это требует распространения большого объема информации в случае каких-либо изменений в сети, или каждый раз, когда администраторам необходимо проверить непротиворечивость информации. Таким образом, необходимо найти решение, которое позволило бы группе лиц управлять локальными маршрутизаторами в автономном режиме так, чтобы добавление новых сетевых взаимосвязей и маршрутов не влияло на работу удаленных маршрутизаторов.

Для того чтобы в дальнейшем было легче объяснить некоторые из особенностей структуры сетевой маршрутизации, рассмотрим третью топологическую схему, в которой половина населенных пунктов расположена в восточной части страны, а другая половина — в западной. Предположим, что через реку, которая отделяет Восток от Запада, перекинут только один мост. Допустим, что люди, живущие в восточной части, не любят уроженцев Запада. По этой причине они используют только те дорожные знаки, на которых указаны пункты восточного региона. Предположим также, что люди, проживающие на Западе, делают обратное. Маршрутизация будет непротиворечивой только в том случае, если на любом дорожном знаке в восточной части страны будут явно перечислены все пункты этого региона, а также указан стандартный маршрут, ведущий к мосту. Что касается западной части, то на каждом знаке, соответственно, должны быть явно перечислены все пункты западного региона, а также указан стандартный маршрут, ведущий к мосту.

## 14.4. Первоначальная структура сети Internet и ее ядро

Большая часть наших знаний о маршрутизации и протоколах распространения маршрутов была почерпнута из опыта, приобретенного в результате работы в глобальной сети Internet. Когда впервые был создан протокол TCP/IP, сетевые центры, принимающие участие в исследованиях, были подключены к сети APRANET, которая выполняла функции магистрального канала Internet. Когда проводились первые эксперименты, каждый из сетевых центров сам управлял

таблицами маршрутизации и вручную устанавливали маршруты к другим получателям. Когда сеть Internet начала расти и развиваться, стало ясно, что вручную устанавливать и контролировать маршруты непрактично. Возникла потребность в выработке автоматических механизмов.

Создатели сети Internet выбрали такую структуру системы маршрутизации, которая состояла из небольшого набора базовых маршрутизаторов, содержащих полную информацию обо всех возможных получателях, и более обширного набора внешних маршрутизаторов, содержащих неполную информацию о маршрутах. Если провести аналогию, этот подход напоминает создание небольшого количества перекрестков в центре города, на которых есть указатели. На них перечислены все возможные пункты назначения. Причем на внешних перекрестках перечислены только местные пункты. При условии, что стандартный маршрут на каждом внешнем перекрестке указывает на один из центральных перекрестков, путешественники в конечном счете достигнут нужных пунктов назначения. Преимущество использования неполной информации во внешних маршрутизаторах заключается в том, что это позволяет местным администраторам управлять изменениями в структуре локальной сети, не затрагивая другие части объединенной сети. Недостаток этого подхода заключается в том, что создается вероятность возникновения противоречий. В самом худшем случае, ошибка во внешнем маршрутизаторе может помешать доступу к удаленным маршрутизаторам. Подведем итог всему сказанному выше.

*Таблица маршрутизации произвольного маршрутизатора содержит неполную информацию о возможных получателях. Маршрутизация, в которой используется неполная информация, позволяет сетевым центрам самостоятельно вносить изменения в локальные маршруты, но вместе с тем создает вероятность возникновения противоречий. Все это может помешать доступу к определенным получателям со стороны некоторых отправителей.*

Несоответствие информации в таблицах маршрутизации, как правило, возникает в результате ошибок в алгоритмах, по которым вычисляются значения в таблице маршрутизации, а также из-за неправильных данных, предоставленных этим алгоритмам, или в результате ошибок, возникающих при передаче результатов другим маршрутизаторам. Разработчики протоколов ищут способы, с помощью которых можно было бы ограничить последствия ошибок. Их главная задача заключается в том, чтобы обеспечить отсутствие противоречий во всех маршрутизаторах в любой момент их работы. Если по какой-либо причине в маршрутизаторах возникают несоответствия, протоколы маршрутизации должны быстро обнаружить и исправить ошибки. И самое важное: протоколы должны быть разработаны таким образом, чтобы ограничить последствия этих ошибок.

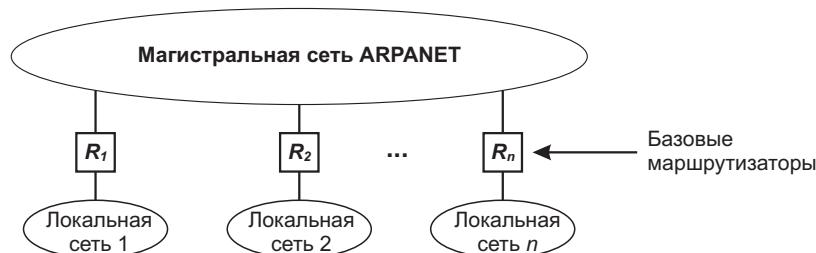
## 14.5. Базовая система маршрутизации

Первые появившиеся в сети Internet маршрутизаторы можно условно разделить на две группы: небольшой набор базовых маршрутизаторов (*core routers*), обслуживаемых Информационным центром сети Internet (INOC, или Internet Network Operations Center), и более обширный набор *второстепенных маршрутизаторов*<sup>2</sup> (*noncore routers*), обслуживаемых отдельными административными группами. Система базовых маршрутизаторов была разработана для того, чтобы обеспечить создание надежной и согласованной системы маршрутизации для

<sup>2</sup> Кроме термина *второстепенный* (*noncore*), для обозначения этих маршрутизаторов использовались также термины *тупиковый* (*stub*), или *немаршрутизуемый* (*nonrouting*).

всех возможных получателей. Она была связующим звеном в сети Internet и создавала возможность глобальной связи. Согласно нормативным документам, каждый сетевой центр должен был сообщить выделенный ему IP-адрес сети системе базовых маршрутизаторов. Поскольку базовые маршрутизаторы обменивались информацией, существовала гарантия, что их общая информация не содержит противоречий. Благодаря тому что работу базовых маршрутизаторов контролировал центральный административный орган Internet, надежность их работы не вызывала сомнений.

Для того чтобы понять особенность системы базовых маршрутизаторов, необходимо вспомнить, что Internet возникла на основе уже существующей глобальной сети APRANET. Когда начали проводить первые эксперименты с Internet, разработчики выбрали APRANET в качестве основной сетевой магистрали, на которой в будущем должна была строиться сеть Internet. Таким образом, толчком к созданию системы базовых маршрутизаторов стало прежде всего желание подключить локальные сети к магистрали APRANET. На рис. 14.1 показана структурная схема этой системы.



*Рис. 14.1. Первая система базовых маршрутизаторов в Internet состояла из набора маршрутизаторов, с помощью которых локальные сети подключались к магистрали APRANET. Узлы в локальных сетях направляли весь внешний трафик ближайшему из базовых маршрутизаторов*

Чтобы понять, почему такую структуру нельзя применить для маршрутизации при неполной информации, предположим, что большая объединенная сеть полностью состоит из локальных сетей, которые подключены к магистральной сети через маршрутизаторы. Представим также, что некоторые из маршрутизаторов выполняют маршрутизацию на основе стандартных маршрутов. Теперь рассмотрим маршрут, по которому следует дейтаграмма. В сетевом центре отправителя локальный маршрутизатор проверяет, есть ли у него явно заданный маршрут к получателю. При отсутствии такого маршрута маршрутизатор отсылает дейтаграмму по стандартному маршруту. Все дейтаграммы, для которых у маршрутизатора отсутствует явно заданный маршрут, следуют по одному и тому же стандартному маршруту, независимо от получателя, которому они предназначены. Следующий маршрутизатор, расположенный по маршруту следования, отводит в сторону дейтаграммы, для которых он имеет явно заданный маршрут, а остальные отсылает по своему стандартному маршруту. Чтобы обеспечить согласованную работу глобальной сети, цепочка стандартных маршрутов должна объединять каждый из маршрутизаторов в одно гигантское кольцо, как показано на рис. 14.2. Таким образом, структура сети требует, чтобы все локальные сетевые центры согласовывали свои стандартные маршруты. Кроме того, зависимость от маршрутов по умолчанию может быть неэффективной, даже если в самой системе нет противоречий. Как показано на рис. 14.2, в самом худшем случае при передаче дейтаграммы от источника к получателю, она пройдет через все  $n$  маршрутизаторов, вместо того, чтобы прямо пойти через магистраль.

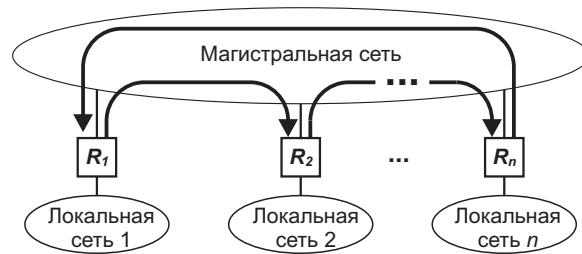


Рис. 14.2. Набор маршрутизаторов, подключенных к магистрали. В данном случае процесс маршрутизации неэффективен, даже при условии, что в системе нет противоречий, поскольку каждое из устройств выполняет маршрутизацию на основе стандартного маршрута

Чтобы избежать недоразумений при использовании стандартных маршрутов, создатели Internet организовали обмен информацией между всеми базовыми маршрутизаторами. Таким образом, каждый из них располагал полной информацией об оптимальных маршрутах ко всем возможным получателям. Поскольку любому из базовых маршрутизаторов были известны маршруты ко всем возможным получателям, им не нужно задавать стандартные маршруты. Если адрес конечного получателя, указанный в дейтаграмме, отсутствовал в таблице базового маршрутизатора, он посыпал отправителю ICMP-сообщение о недоступности получателя и аннулировал дейтаграмму. По существу, введение системы базовых маршрутизаторов позволило решить проблему неэффективной маршрутизации и исключить применение стандартных маршрутов.

На рис. 14.3 схематично показана структура системы базовой маршрутизации. Она состоит из ядра, в которое входит один или несколько базовых маршрутизаторов, и набора внешних маршрутизаторов, расположенных в локальных сетевых центрах. Внешние маршрутизаторы содержат информацию о получателях,

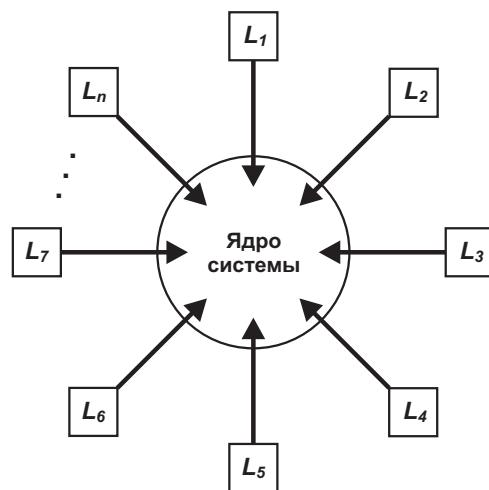


Рис. 14.3. Упрощенная схема базовой системы маршрутизации, на которой показаны стандартные маршруты. В базовых маршрутизаторах стандартные маршруты не используются. Каждый из внешних маршрутизаторов, обозначенный  $L_i$ , имеет свой стандартный маршрут, ведущий в ядро системы

расположенных в их локальных сетях, и стандартный маршрут. Благодаря ему дейтаграммы, предназначенные для других сетевых центров, пересыпаются в ядро системы.

Хотя упрощенная структурная схема базовой системы маршрутизации (см. рис. 14.3) достаточно прозрачна, но в то же время непрактична по трем причинам. Во-первых, Internet выросла из одной протяженной сетевой магистрали, управляемой из единого центра. Топологическая схема усложнилась, нетривиальными стали и протоколы, необходимые для сохранения непротиворечивости информации в базовых маршрутизаторах. Во-вторых, не каждый сетевой центр мог иметь в своем составе базовый маршрутизатор, подключенный к магистрали. Таким образом, возникла необходимость создания дополнительной структуры маршрутизации и протоколов. В третьих, поскольку для поддержания непротиворечивости информации все базовые маршрутизаторы должны были взаимодействовать между собой, структуру сетевого ядра нельзя было увеличивать до произвольных размеров. Мы вернемся к последнему вопросу в главе 15, “Маршрутизация между автономными системами (BGP)”, после того как рассмотрим протоколы, используемые системой базовых маршрутизаторов для обмена маршрутной информацией.

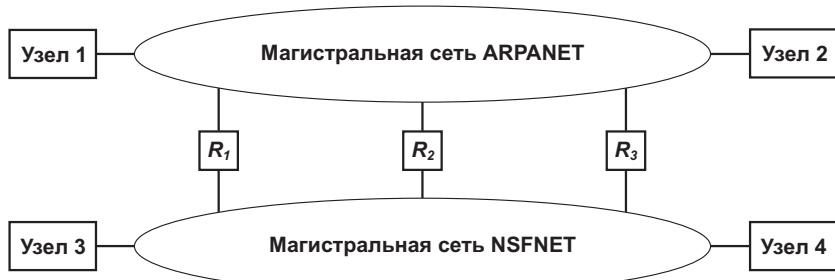
## 14.6. От сетевого ядра до системы равноправных магистралей

Подключение магистрали NSFNET к сети Internet создало дополнительную сложность в структуре системы маршрутизации. С точки зрения базовых маршрутизаторов подключение сети NSFNET вначале ничем не отличалось от подключения любого другого сетевого центра к магистральной сети Internet. Магистраль NSFNET была подключена к магистрали ARPANET через единственный маршрутизатор, расположенный в Питсбурге. В ядре системы явно указывались маршруты ко всем получателям, подключенными к магистрали NSFNET. Маршрутизаторы внутри магистрали NSFNET располагали информацией о локальных получателях и использовали стандартные маршруты для отправки сетевому ядру всего трафика, не принадлежащего магистрали, посредством маршрутизатора в Питсбурге.

Когда магистраль NSFNET разрослась и стала основной составляющей сети Internet, стало ясно, что структура базовой маршрутизации не сможет удовлетворить всем требованиям. Важное изменение в структуре системы маршрутизации произошло тогда, когда между магистралями ARPANET и NSFNET возникло несколько дополнительных соединений. Таким образом, ARPANET и NSFNET превратились в *равноправные сетевые магистрали* (*peer backbone networks*). На рис. 14.4 показана получившаяся в результате топологическая схема равноправных сетевых магистралей.

Чтобы понять трудности IP-маршрутизации между равноправными магистралями, рассмотрим маршруты, ведущие от узла 3 до узла 2, которые изображены на рис. 14.4. Представим себе, что на рисунке изображена географическая карта. Поэтому узел 3 находится на Западном побережье, которое принадлежит магистрали NSFNET, в то время как узел 2 — на Восточном побережье, которое принадлежит магистрали ARPANET. При установке маршрутов между узлами 3 и 2 администраторам необходимо решить следующие вопросы:

- а) стоит ли направлять трафик от узла 3 через маршрутизатор  $R_1$ , расположенный на Западном побережье, а затем по магистрали ARPANET, или
- б) направлять трафик от узла 3 по магистрали NSFNET, через центральный маршрутизатор  $R_2$ , а затем по магистрали ARPANET — к узлу 2, или все-таки
- в) направлять трафик по магистрали NSFNET, через маршрутизатор на Восточном побережье,  $R_3$ , а затем по магистрали ARPANET — к узлу 2.



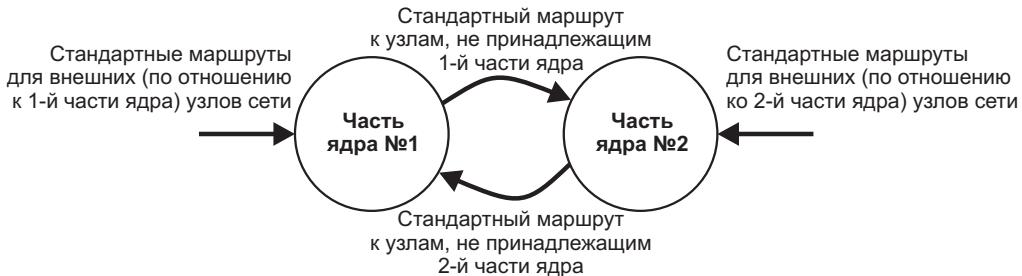
*Рис. 14.4. Система равноправных сетевых магистралей, соединенных между собой посредством нескольких маршрутизаторов. На схеме показана структура сети Internet по состоянию на 1989 год. При дальнейших расширениях Internet каждая из параллельных сетевых магистралей принадлежала одному из провайдеров*

Можно также проложить маршрут и обходными путями. Например, трафик может проходить от узла 3 через маршрутизатор на Западном побережье, по магистрали ARPANET к центральному маршрутизатору, снова по магистрали NSFNET к маршрутизатору на Восточном побережье и, наконец, по магистрали ARPANET к узлу 2. Такой маршрут может быть целесообразным или — нет, в зависимости от целей использования сети и пропускной способности различных маршрутизаторов и магистралей.

Для большинства конфигураций равноправных сетевых магистралей должно соблюдаться следующее правило: трафик между двумя географически близко расположеными узлами сети должен проходить по кратчайшему пути, независимо от настроек маршрутизаторов, перенаправляющих межгосударственный трафик. Например, трафик от узла 3 до узла 1 должен проходить через маршрутизатор на Западном побережье, потому что он сокращает до минимума путь прохождения пакетов по обеим магистралям.

Все эти положения звучат довольно просто, но на самом деле их тяжело реализовать по двум причинам. Во-первых, хотя стандартный алгоритм маршрутизации протокола IP использует сетевую часть IP-адреса для выбора маршрута, при реализации методов оптимальной маршрутизации в структуре равноправных сетевых магистралей для отдельных узлов требуется проложить отдельные маршруты. Что касается примера, приведенного выше, в таблице маршрутизации узла 3 должны присутствовать разные маршруты для достижения узлов 1 и 2, несмотря на то, что оба этих узла подключены к магистрали APRANET. Во-вторых, администраторы обоих магистралей должны договориться о том, чтобы в таблицах маршрутизации всех устройств содержалась непротиворечивая информация. В противном случае могут возникать так называемые *маршрутные петли (routing loops)*. (Маршрутная петля возникает тогда, когда несколько маршрутизаторов “перебрасывают” пакеты по кругу друг другу.)

Важно уметь отличать топологическую схему сети от структуры системы маршрутизации. Например, может существовать единое ядро сети, состоящее из нескольких магистральных сетей. Машины сетевого ядра можно запрограммировать таким образом, чтобы они скрывали низкоуровневые особенности структуры системы и вычисляли кратчайшие маршруты друг к другу. Однако невозможно разбить систему сетевого ядра на подмножества, каждое из которых содержало бы частичную информацию, не теряя при этом своих функциональных возможностей. Эта проблема продемонстрирована на рис. 14.5.



*Рис. 14.5. Попытка разбить систему базовой маршрутизации на два набора маршрутизаторов, содержащих неполную информацию и поэтому использующих стандартные маршруты. Такая структура приводит к образованию маршрутной петли для дейтаграмм, ошибочно посланных несуществующему получателю*

Как показано на рис. 14.5, стандартные маршруты внешних маршрутизаторов направлены в одну из частей разделенного сетевого ядра. Маршрутизаторы каждой из частей ядра обладают информацией о получателях, находящихся в своем регионе, а также используют стандартный маршрут для передачи пакетов, предназначенных для другого региона. В такой структуре любая дейтаграмма, отосланная по недопустимому адресу, будет циклически передаваться по кругу между двумя частями ядра до тех пор, пока не истечет время ее жизни. Таким образом, можно сделать следующий вывод.

*Структура базовой системы маршрутизации предполагает, что централизованный набор маршрутизаторов выполняет функции хранилища информации обо всех возможных получателях в объединенной сети. Базовые системы маршрутизации лучше всего функционируют в тех объединенных сетях, которые имеют одну, управляемую из центра магистраль. Расширение топологических схем и подключение нескольких равноправных магистралей усложняет систему маршрутизации, а попытки разделить ядро сети на несколько частей таким образом, чтобы все маршрутизаторы могли использовать стандартные маршруты, создает предпосылку для образования маршрутных петель.*

## 14.7. Автоматическое распространение маршрутной информации

Выше уже отмечалось, что в первоначальной системе базовой маршрутизации сети Internet не использовались стандартные маршруты. Причина в том, что маршрутизаторы, находящиеся в ядре сети, располагали полной информацией обо всех возможных получателях и обменивались ею между собой. В настоящее время похожая система используется во многих корпоративных объединенных сетях. Корпоративные маршрутизаторы работают под управлением специальной программы, которая передает маршрутную информацию. В последующих разделах обсуждаются два основных вида алгоритмов, которые используются для вычисления и распространения маршрутной информации. Для иллюстрации одного из алгоритмов в них используется оригинальный протокол базовой маршрутизации. Затем, в следующем разделе описывается протокол, в котором применяется другой тип алгоритма.

На первый взгляд может показаться, что в механизмах для автоматического распространения маршрутной информации нет необходимости, особенно в случае

небольших объединенных сетей. Однако объединенные сети не являются статичными. Межсетевые соединения периодически выходят из строя и со временем заменяются на другие. В определенный момент в сети может возникнуть перегрузка либо сеть может использоваться не на полную мощность. Целью механизма автоматического распространения маршрутов является не только поиск набора маршрутной информации, но и постоянное ее обновление. Человек просто не может достаточно быстро реагировать на изменения, происходящие в объединенной сети, поэтому возникает необходимость в использовании компьютерных программ. Таким образом, говоря о распространении маршрутов, важно учитывать работу протоколов и алгоритмов в динамике.

## 14.8. Дистанционно-векторная маршрутизация (метод Беллмана-Форда)

Термин *дистанционный вектор*<sup>3</sup> (*distance-vector*) относится к классу алгоритмов, используемых маршрутизаторами для распространения маршрутной информации. Идея, положенная в основу дистанционно-векторного алгоритма, довольно проста. Маршрутизатор хранит в таблице список всех известных маршрутов. При загрузке он инициализирует таблицу маршрутизации и помещает в нее данные для всех непосредственно подключенных сетей. В каждом элементе таблицы определяется сеть получателя и указывается расстояние до этой сети, которое, как правило, измеряется количеством переходов (более точное определение этому понятию будет дано чуть позже). Например, в табл. 14.1 показано начальное содержимое таблицы маршрутизации устройства, подключенного к двум сетям. В ней хранится IP-адрес сети, а также целое число, определяющее расстояние до этой сети.

**Таблица 14.1. Начальное содержимое таблицы дистанционно-векторной маршрутизации, содержащей данные о каждой непосредственно подключенной сети**

Получатель	Расстояние	Маршрут
Сеть 1	0	прямой
Сеть 2	0	прямой

Периодически каждый маршрутизатор отсылает копию своей таблицы маршрутизации другим маршрутизаторам, к которым он может получить непосредственный доступ. Когда маршрутизатору  $K$  приходит сообщение от маршрутизатора  $J$ , он анализирует полученный в нем набор адресатов и расстояние к каждому из них. Маршрутизатор  $K$  заменяет данные в своей таблице в следующих случаях:

- а) если маршрутизатору  $J$  известен более короткий маршрут к получателю или в его списке есть получатель, сведения о котором отсутствуют в таблице маршрутизатора  $K$ ;
- б) если маршрутизатор  $K$  выполняет маршрутизацию своих пакетов к конечному получателю через маршрутизатор  $J$ , но расстояние от маршрутизатора  $J$  до конечного получателя изменилось. Например, в табл. 14.2(а) показана существующая таблица маршрутизатора  $K$  и сообщение об обновлении, полученное от

<sup>3</sup> У данного метода есть несколько синонимов: *векторно-дистанционная маршрутизация*, а также методы *Форда-Фалкерсона* (*Ford-Fulkerson*), *Беллмана-Форда* и *Беллмана*. Последние два синонима названы по имени исследователей, которые первыми опубликовали идею алгоритма.

маршрутизатора,  $J$  (б). Отмеченные данные будут использованы для обновления существующей информации или для введения новых данных в таблицу маршрутизатора  $K$ .

**Таблица 14.2. Существующая таблица маршрутизации устройства  $K$  (а) и входящее сообщение об обновлении маршрутной информации от маршрутизатора  $J$  (б)**

Получатель	Расстояние	Маршрут	Получатель	Расстояние
Сеть 1	0	прямой	Сеть 1	2
Сеть 2	0	прямой	→ Сеть 4	3
Сеть 4	8	к устройству L	Сеть 17	6
Сеть 17	5	к устройству M	→ Сеть 21	4
Сеть 24	6	к устройству J	Сеть 24	5
Сеть 30	2	к устройству Q	Сеть 30	10
Сеть 42	2	к устройству J	→ Сеть 42	3

(а)

(б)

Обратите внимание, если маршрутизатор  $J$  сообщает о расстоянии  $N$  до получателя, то в обновленных данных маршрутизатора  $K$  будет указано расстояние до этого же получателя, равное  $N+1$  (т.е. расстояние, которое необходимо преодолеть для достижения получателя от маршрутизатора  $J$  плюс расстояние к самому маршрутизатору  $J$  от маршрутизатора  $K$ .) Разумеется, что в элементах таблицы маршрутизации есть и третий столбец, в котором указывается адрес ближайшей точки перехода. Для всех непосредственно подключенных сетей в поле адреса ближайшей точки перехода отмечается, что они доступны *напрямую*. Если маршрутизатор  $K$  добавляет или обновляет элемент таблицы маршрутизации в ответ на сообщение, полученное от маршрутизатора  $J$ , он назначает маршрутизатор  $J$  в качестве адреса ближайшей точки перехода для этого элемента.

Термин *дистанционный вектор* происходит от типа информации, которая отсылается маршрутизаторами в периодических сообщениях. В сообщении содержится список пар  $(V, D)$ , в которых  $V$  определяет получателя (называемого *вектором*), а  $D$  — расстояние к этому получателю. Обратите внимание, что дистанционно-векторные алгоритмы сообщают маршруты от первого лица (т.е. подразумевается, что маршрутизатор сообщает: “Я могу достичь получателя  $V$  за  $D$  переходов.”) В описанной выше схеме все маршрутизаторы должны принимать участие в дистанционно-векторном обмене для того, чтобы обеспечить эффективную и согласованную работу системы маршрутизации.

Хотя дистанционно-векторные алгоритмы легко реализовать, они имеют ряд недостатков. В полностью статичной среде дистанционно-векторные алгоритмы распространяют информацию о маршрутах, ведущих ко всем получателям. Однако, если маршруты быстро изменяются, результат работы дистанционно-векторного алгоритма может быть нестабильным. При изменении маршрутов (т.е. если появляется новое соединение или выходит из строя старое) информация будет медленно распространяться от одного маршрутизатора до другого. При этом некоторые маршрутизаторы могут содержать неправильную маршрутную информацию.

А сейчас рассмотрим простой протокол, который используется в дистанционно-векторном алгоритме, не обращая внимание на все его недостатки. В главе 16,

“Маршрутизация внутри автономной системы (RIP, OSPF, HELLO)”, завершается рассмотрение протоколов маршрутизации. В ней описан еще один дистанционно-векторный протокол, рассмотрены проблемы, которые могут возникнуть при его использовании, а также эвристический алгоритм, применяемый для решения самых трудных из этих проблем.

## 14.9. Межшлюзовый протокол (GGP)

В первоначальной системе базовой маршрутизации для обмена маршрутной информацией между устройствами использовали дистанционно-векторный протокол, который назывался *межшлюзовый протокол*<sup>4</sup> (GGP, или *Gateway-To-Gateway Protocol*). Хотя протокол GGP только управлял информацией о классовых маршрутах и сейчас не входит в стандарты протокола TCP/IP<sup>5</sup>, он все же представляет собой конкретный пример дистанционно-векторной маршрутизации. Протокол GGP был разработан для передачи маршрутной информации в IP-дейтаграммах, по аналогии с тем, как передаются обычные данные в UDP-дейтаграммах или TCP-сегментах. Каждое сообщение протокола GGP имеет заголовок установленного формата, в котором указывается тип сообщения, а также формат остальных полей. Поскольку протокол GGP использовался только в базовых маршрутизаторах, а они, в свою очередь, управлялись центральным административным органом Internet, другие маршрутизаторы не могли повлиять на процесс обмена информацией.

Первоначальная система ядра сети была организована так, чтобы к ней можно было подключить новые базовые маршрутизаторы, при этом не изменяя информацию в существующих маршрутизаторах. Когда к ядру системы подключался новый маршрутизатор, для него назначались один или несколько базовых *соседних маршрутизаторов*, с которыми он должен был обмениваться информацией. Соседние маршрутизаторы, входящие в ядро сети, также распространяли маршрутную информацию между другими базовыми маршрутизаторами. Поэтому новому маршрутизатору нужно было лишь сообщить своим соседям о сетях, к которым он имел непосредственный доступ. Соседние маршрутизаторы, в свою очередь, обновляли свои таблицы маршрутизации и распространяли уже обновленную информацию другим маршрутизаторам.

GGP — это дистанционно-векторный протокол. Информация, которой маршрутизаторы обмениваются по протоколу GGP, состоит из парного набора ( $N$ ,  $D$ ), в котором  $N$  — это IP-адрес сети, а  $D$  — расстояние, до этой сети, измеряемое *количество переходов* (*hops*). Таким образом, маршрутизатор с помощью протокола GGP объявляет о сетях, доступ к которым он может получить, а также о затратах, необходимых для получения этого доступа.

В протоколе GGP расстояние измеряется *количество переходов через маршрутизаторы*. Причем маршрутизатор считается размещенным на нулевом расстоянии от непосредственно соединенных сетей, на расстоянии одного перехода от сетей, доступ к которым можно получить через другой маршрутизатор, и т.д. Таким образом, *количество*, или *счетчик*, *переходов* по маршруту от отправителя до конечного получателя позволяет судить о количестве маршрутизаторов, через которые проходит дейтаграмма. Очевидно, что использование метода подсчета количества переходов для определения кратчайшего пути не всегда приводит к желаемым результатам. Например, передать дейтаграмму по маршруту с количеством

<sup>4</sup> Напомним, что хотя у производителей сетевого оборудования прижился термин *IP-маршрутизатор*, исследователи первоначально использовали термин IP-шлюз.

<sup>5</sup> Группа IETF признала протокол GGP устаревшим, а это означает, что его больше не рекомендуется использовать совместно с семейством протоколов TCP/IP.

переходов, равным 3, проходящему по локальным сетям, может оказаться намного быстрее, чем по маршруту с количеством переходов, равным 2, но проходящему через два низкоскоростных канала последовательной передачи данных. Поэтому во многих маршрутизаторах для маршрутов, проходящих по низкоскоростным сетям, используется искусственно завышенное количество переходов.

## 14.10. Упорядочение расстояния

Подобно большинству протоколов маршрутизации, в протоколе GGP используется несколько *типов* сообщений, каждое из которых имеет определенный формат и назначение. Поле в заголовке сообщения содержит код, который определяет тип сообщения. Благодаря этому коду получатель знает, как обработать сообщение. Например, прежде чем два маршрутизатора обмениваются маршрутной информацией, они должны наладить между собой связь, для чего используются сообщения определенного типа. Основным типом сообщения в протоколе GGP является сообщение об обновлении маршрутной информации, с помощью которого маршрутизаторы обмениваются информацией. Этот тип сообщения также является основным для любого другого дистанционно-векторного протокола.

Теоретически в сообщении об обновлении маршрутной информации должен содержаться список пар, каждая из которых состоит из IP-адреса сети и расстояния до этой сети. Однако на практике во многих протоколах маршрутизации порядок подачи информации изменен, чтобы уменьшить объем передаваемых сообщений. В частности, существует незначительное количество структур, в которых маршрутизаторы и сами объединяемые ими сети размещены в линейном порядке. Чаще всего можно встретить иерархическую структуру, в которой к каждой сети подключено несколько маршрутизаторов. Следовательно, значения, указывающие на расстояние в сообщениях об обновлении, представляют собой небольшие числа. Кроме того, одни и те же значения могут часто повторяться. Чтобы уменьшить размер сообщения, в протоколах маршрутизации часто применяется метод, который был впервые использован в протоколе GGP. При использовании этого метода, который называется *упорядочением расстояния* (*distance factoring*), копии сообщений с одним и тем же показателем расстояния не отсылаются. Вместо этого список пар сортируется по расстоянию. Причем каждое значение расстояния указывается только один раз, а после него перечисляются сети, доступные на этом расстоянии. В следующей главе показано, каким образом другие протоколы маршрутизации упорядочивают информацию.

## 14.11. Надежность и протоколы маршрутизации

Большинство протоколов маршрутизации используют транспортный протокол, не требующий предварительной установки соединения. Например, протокол GGP инкапсулирует сообщения непосредственно в IP-дейтаграммы. Современные протоколы маршрутизации, как правило, инкапсулируют сообщения в UDP-дейтаграммы<sup>6</sup>. Оба протокола, IP и UDP, не гарантируют доставку дейтаграмм. Это означает, что в процессе доставки дейтаграммы могут затеряться, задержаться в пути, а также появиться их дубли. Кроме того, информация, содержащаяся в дейтаграммах, может быть искажена или доставлена в неправильном порядке. Поэтому протокол маршрутизации, который использует ненадежные транспортные службы доставки пакетов, должен самостоятельно компенсировать сбои в сети.

<sup>6</sup> Однако существуют и исключения. В следующей главе будет описан протокол маршрутизации, использующий в качестве транспортного протокол TCP.

В протоколах маршрутизации используется несколько методов для решения проблем, связанных с ненадежной доставкой сообщений. Чтобы определить факт искажения информации используется механизм подсчета контрольных сумм. Проблема потери данных решается либо с помощью технологии систем с *неустойчивым состоянием*<sup>7</sup> (*soft state*) на основе таймеров, либо через механизм подтверждений и повторной передачи сообщений. Например, в протоколе GGP применяется расширенная схема подтверждения приема, в которой получатель может после получения сообщения прислать положительный или отрицательный сигнал подтверждения приема.

Проблема нарушения порядка доставки сообщений или поступления соответствующего ответа после получения старого сообщения в протоколах маршрутизации часто решается с помощью *порядковых номеров*. Например, в протоколе GGP каждая из сторон при установке соединения выбирает свой начальный номер, который используется для управления обменом информацией. Другая сторона должна сначала подтвердить этот номер. После начального обмена информацией в каждом посыпаемом сообщении указывается следующий по порядку номер, который позволяет получателю определить, в правильном ли порядке получены сообщения. В последующей главе будет рассмотрен пример протокола маршрутизации, в котором используется технология систем с неустойчивым состоянием.

## 14.12. Маршрутизация, отслеживающая состояние соединения (SPF)

Самый большой недостаток дистанционно-векторного алгоритма заключается в том, что он недостаточно хорошо приспособлен к изменениям, происходящим в сети. Кроме замедленной реакции на изменения, о которой уже упоминалось, для работы алгоритма требуется, чтобы маршрутизаторы обменивались большими объемами информации. Поскольку в каждом сообщении об обновлении маршрутной информации содержатся данные о каждой из существующих сетей, размер сообщения пропорционален общему количеству сетей, подключенных к объединенной сети. Кроме того, поскольку дистанционно-векторный протокол требует, чтобы в процессе обмена информацией были задействованы все маршрутизаторы, объем передаваемой при этом информации может достигать огромных размеров.

Основной альтернативой дистанционно-векторных алгоритмов является класс алгоритмов, отслеживающих *состояние соединения* (*link state, link status*), или выполняющих поиск *первого кратчайшего пути*<sup>8</sup> (*SPF*, или *Shortest Path First*). Для работы алгоритма поиска кратчайшего пути (*SPF*) требуется, чтобы каждый задействованный маршрутизатор имел полную информацию о топологической схеме сети. Проще всего понять, что такая топологическая информация, представив себе, что у каждого маршрутизатора есть карта, на которой изображены все остальные маршрутизаторы, а также сети, к которым они подключены. В абстрактном понимании, маршрутизаторы соответствуют вершинам графа, а сети, связывающие маршрутизаторы, соответствуют ребрам графа. Между двумя вершинами графа существует ребро (соединение), если соответствующие им маршрутизаторы могут напрямую обмениваться информацией.

<sup>7</sup> Напомним, что в системах с неустойчивым состоянием информация считается устаревшей после истечения тайм-аута, а не после получения специального сообщения от отправителя.

<sup>8</sup> Название “первый кратчайший путь” немного сбивает с толку, поскольку алгоритмы во всех маршрутизаторах просто выполняют поиск кратчайшего пути.

Маршрутизатор, который задействован в алгоритме SPF, не отсылает сообщения, содержащие списки получателей, а выполняет две задачи. Во-первых, он активно проверяет состояние соседних маршрутизаторов. В терминах теории графов, два маршрутизатора являются соседними, если они пользуются одним и тем же соединением. С точки зрения сетей два соседних маршрутизатора должны быть подключены к общей сети. Во-вторых, алгоритм SPF периодически распространяет информацию о состоянии соединения всем остальным маршрутизаторам.

Чтобы проверить состояние своего “соседа”, маршрутизатор периодически обменивается с ним короткими сообщениями и таким образом выясняет, в рабочем ли он состоянии и возможен ли к нему доступ. Если соседний маршрутизатор отвечает на запросы, считается, что соединение между ними *установлено* (*up*). В противном случае считается, что соединение *разорвано*<sup>9</sup> (*down*). Чтобы проинформировать все остальные маршрутизаторы, каждый маршрутизатор периодически рассыпает в широковещательном режиме сообщение, в котором указывает состояние каждого своего соединения. Сообщение о состоянии соединения не определяет маршруты, а просто сообщает, возможен ли обмен информацией между парами маршрутизаторов. Программы протокола маршрутизатора обеспечивают доставку экземпляра каждого сообщения о состоянии соединения всем задействованным маршрутизаторам. Если используемая низкоуровневая сетевая технология не позволяет осуществить режим широковещательной передачи, доставка сообщений осуществляется путем рассылки его отдельных экземпляров каждому из получателей.

Когда приходит сообщение о состоянии соединения, маршрутизатор использует эту информацию, чтобы обновить свою карту объединенной сети, обозначая на ней установленные и разорванные соединения. Каждый раз, когда изменяется состояние соединения, маршрутизатор повторно вычисляет маршруты, применяя хорошо известный *алгоритм поиска кратчайшего пути Дейкстры* (*Dijkstra*) к результирующему графу. Алгоритм Дейкстры вычисляет кратчайшие пути от одного отправителя ко всем получателям.

Одно из основных преимуществ алгоритмов SPF заключается в том, что каждый маршрутизатор вычисляет кратчайшие пути независимо, используя одни и те же данные о состоянии соединения, которые не зависят от вычислений промежуточных машин. Поскольку сообщения о состоянии соединения распространяются без изменений, то возникающие проблемы легко локализовать. Поскольку маршрутизаторы вычисляют маршруты локально, их правильность гарантирована. И наконец, поскольку в сообщениях о состоянии соединения указывается информация только о тех соединениях, которые непосредственно подключены к данному маршрутизатору, их размер не зависит от количества сетей в объединенной сети. Таким образом, алгоритмы SPF лучше приспособлены к изменениям, происходящим в сети, чем дистанционно-векторные алгоритмы.

## 14.13. Резюме

Чтобы обеспечить надежный доступ ко всем сетям, подключенными к объединенной сети, в ней должна быть создана глобальная система согласованной маршрутизации. В узлах сети и в большинстве маршрутизаторов может находиться только неполная маршрутная информация. Поэтому для отправки дейтаграмм

<sup>9</sup> На практике для предотвращения колебаний системы из одного состояния в другое в большинстве протоколов используется метод статистической оценки. Соединение считается установленным до тех пор, пока соседний маршрутизатор отвечает на большую часть посланных ему запросов. И наоборот, соединение считается разорванным, если большая часть запросов остается без ответа.

удаленным получателям узлы сети и маршрутизаторы полагаются на стандартные маршруты. Первоначально в глобальной сети Internet проблема маршрутизации решалась путем использования системы базовых маршрутизаторов, каждый из которых содержал полную информацию обо всех сетях. Маршрутизаторы, входящие в первоначальную систему сетевого ядра Internet, периодически обменивались маршрутной информацией. Это означало, что как только о маршруте узнавал один базовый маршрутизатор, о нем становилось известно всем другим базовым маршрутизаторам. Чтобы предотвратить образование маршрутных петель, базовым маршрутизаторам запрещалось использовать стандартные маршруты.

Единая, управляемая из центра система базовой маршрутизации хорошо функционирует в случае, если структура объединенной сети построена на основе одной магистрали. Однако эта система недостаточно хорошо работает в объединенной сети, которая состоит из набора отдельно управляемых равноправных магистралей, соединенных друг с другом в нескольких местах.

Когда маршрутизаторы обмениваются маршрутной информацией, они используют для этого один из двух алгоритмов: дистанционно-векторный алгоритм или алгоритм поиска кратчайшего пути, SPF. Для распространения информации об обновлении маршрутов по ядру сети Internet сначала использовался дистанционно-векторный протокол GGP.

Главный недостаток дистанционно-векторных алгоритмов заключается в том, что в них вычисление кратчайшего пути выполняется распределенно. А это означает, что, если состояние сетевых соединений будет непрерывно меняться, то результат вычислений трудно предсказать. Другой недостаток заключается в том, что размер сообщений об обновлении маршрутной информации пропорционален количеству сетей.

Алгоритмы поиска кратчайшего пути (SPF) использовались еще до возникновения сети Internet. Один из первых протоколов SPF использовался в сетевой магистрали APRANET для создания и управления маршрутами в сетевых коммутаторах. Алгоритм APRANET применялся на протяжении целого десятилетия.

## Материал для дальнейшего изучения

Определение базовой системы маршрутизаторов и протокола GGP, о котором упоминалось в этой главе, сделано Хайнденом (Hinden) и Шельцером (Shelzer) в [RFC 823]. Браден (Braden) и Постел (Postel) в [RFC 1812] приводят дополнительные спецификации протоколов маршрутизации в сети Internet. Элмквист (Almquist) в [RFC 1716] подводит итоги недавних дискуссий. Браун (Braun) в [RFC 1093] и Рехтер (Rekhter) в [RFC 1092] рассматривают маршрутизацию в магистрали NSFNET. Кларк (Clark) в [RFC 1102] и Браун в [RFC 1104] обсуждают маршрутизацию, основанную на принятых сетевых правилах. В последующих двух главах будут описаны протоколы, которые используются для распространения маршрутной информации между отдельными сетевыми центрами, а также в пределах одного сетевого центра.

## Упражнения

- 14.1. Предположим, маршрутизатор обнаружил, что ему необходимо переслать IP-дейтаграмму через тот же сетевой интерфейс, через который она была получена. Что должен сделать маршрутизатор и почему?
- 14.2. Ознакомьтесь с RFC 823 и RFC 1812 и объясните, что должен сделать базовый маршрутизатор сети Internet (т.е. маршрутизатор, содержащий полную маршрутную информацию) в ситуации, описанной в предыдущем упражнении.

- 14.3.** Как маршрутизаторы сетевого ядра должны использовать стандартные маршруты, чтобы дейтаграммы, адресованные несуществующему получателю, отсылались определенной машине?
- 14.4.** Предположим, что студенты проводят эксперимент с маршрутизатором, через который их локальная сеть подключена к объединенной сети, состоящей из системы базовых маршрутизаторов. Студенты хотят сообщить данные о своей сети базовому маршрутизатору. Однако если они случайно анонсируют маршруты нулевой длины, ведущие к каким-либо сетям, то трафик, поступающий из объединенной сети, будет неправильно перенаправлен к их маршрутизатору. Таким образом сетевое ядро может защитить себя от получения некорректных данных, продолжая при этом принимать сообщения об обновлении маршрутной информации от таких “ненадежных” маршрутизаторов?
- 14.5.** Какие ICMP-сообщения генерирует маршрутизатор?
- 14.6.** Предположим, что маршрутизатор использует ненадежный транспортный протокол для отправки сообщений. Каким образом он может определить, работает ли соседний маршрутизатор? (*Подсказка*. Обратитесь к RFC 823, чтобы узнать, как эта проблема решалась в первоначальной системе сетевого ядра.)
- 14.7.** Предположим, каждый из двух маршрутизаторов сообщает, что на достижение некоторой сети  $N$  у него уходит  $k$  затрат. Опишите условия, при которых маршрутизация через одно устройство может иметь меньше переходов, чем через другое.
- 14.8.** Каким образом маршрутизатор может определить, что во входящей дейтаграмме находится сообщение протокола GGP? А как насчет сообщения протокола OSPF?
- 14.9.** Внимательно изучите сообщение об обновлении дистанционного вектора (см. табл. 14.2). Объясните причину, по которой маршрутизатор обновляет данные для каждого элемента таблицы.
- 14.10.** Рассмотрите, как используются порядковые номера для обеспечения согласованной работы двух маршрутизаторов в тех случаях, когда получен дубль дейтаграммы, произошла задержка дейтаграммы в пути или нарушен ее порядок доставки. Каким образом должны назначаться начальные порядковые номера и почему?

# 15

## *Маршрутизация между автономными системами (BGP)*

### **15.1. Введение**

В предыдущей главе был описан процесс распространения маршрутной информации и рассмотрен один из протоколов, который используется маршрутизаторами для этой цели. В этой главе будет более подробно описана структура маршрутизации в объединенной сети. Будет рассмотрено понятие автономной системы, а также протокол, который используется группой маршрутизаторов, функционирующих под контролем одного административного органа, для предоставления маршрутной информации о своих сетях другим группам.

### **15.2. Усложнение структурной модели**

Как было описано в предыдущей главе, первоначальная система базовой маршрутизации возникла, когда в сети Internet существовала одна глобальная магистраль. Поэтому основная причина создания подобной системы маршрутизации заключалась в том, чтобы обеспечить подключение к этой магистрали локальных сетевых центров. Если объединенная сеть состоит только из одной магистрали и набора подключенных к ней локальных сетей, создание системы базовой маршрутизации обеспечивает правильное распространение необходимой маршрутной информации. Поскольку все маршрутизаторы подключены к глобальной магистральной сети, они могут обмениваться маршрутной информацией непосредственно между собой. Каждый маршрутизатор располагает информацией об одной локальной сети, к которой он подключен, поэтому он может передать эти данные другим маршрутизаторам. Таким образом, любой из маршрутизаторов может получить необходимую ему информацию от других маршрутизаторов.

На первый взгляд может показаться, что структуру ядра сети можно расширить до объединенной сети произвольного размера только путем подключения новых сетевых центров к магистрали через маршрутизатор. К сожалению, такая структура плохо поддается расширению, поскольку вовлечение всех маршрутизаторов в единый процесс маршрутизации можно осуществить только в небольших по размеру объединенных сетях. На то существует три причины. Во-первых, даже если каждый сетевой центр состоит из одной сети, такая структура маршрутизации не может охватить произвольное количество сетевых центров, поскольку каждый дополнительный маршрутизатор создает свой трафик в сети. Поэтому если большое количество маршрутизаторов попытаются вступить во взаимодействие, общий трафик в сети достигнет огромных размеров. Во-вторых,

в подобной структуре в одном сетевом центре не может существовать несколько маршрутизаторов и, соответственно, несколько сетей. Причина в том, что только те маршрутизаторы, которые непосредственно подключены к магистральной сети, могут напрямую обмениваться маршрутной информацией. В-третьих, в большой объединенной сети не все сети и маршрутизаторы находятся под управлением одного административного органа. Кроме того, в ней не всегда используются кратчайшие маршруты. Напротив, поскольку сети принадлежат независимым группам и управляются ими, в каждой из них могут устанавливаться собственные правила административной политики. Поэтому структура маршрутизации в объединенной сети должна обеспечить каждой группе средство, с помощью которого она смогла бы независимо управлять маршрутизацией и доступом к сетям.

Таким образом, ограничив взаимодействие между маршрутизаторами, можно добиться ощутимых результатов. Эта идея является одной из главных причин создания структуры маршрутизации, использующейся в глобальной сети Internet. Кроме того, она позволяет объяснить несколько важных моментов, с которыми вы столкнетесь при изучении материала книги. Подводя итоги, можно так сформулировать один из основополагающих принципов маршрутизации.

*Хотя для выполнения маршрутизации необходимо, чтобы маршрутизаторы обменивались маршрутной информацией, в большой объединенной сети нереально вовлечь все маршрутизаторы в один глобальный процесс обмена информацией.*

### 15.3. Определение практических ограничений на размер группы

После прочтения предыдущего раздела у вас наверняка осталось много вопросов. Например, какого именно размера объединенная сеть считается “большой”? Если в обмене маршрутной информацией может участвовать только ограниченное количество маршрутизаторов, то что происходит с маршрутизаторами, исключенными из этого процесса? Правильно ли будут работать такие маршрутизаторы? Может ли маршрутизатор, не принимающий участие в процессе обмена информацией, направить дейтаграмму маршрутизатору, который в нем участвует? Может ли маршрутизатор, принимающий участие в процессе взаимодействия, направить дейтаграмму маршрутизатору, не участвующему в этом процессе?

Чтобы ответить на вопрос о размере сети, вначале следует рассмотреть алгоритм маршрутизации, который используется в этой сети, учесть пропускную способность сети, соединяющей маршрутизаторы, а также детали используемого протокола маршрутизации. При маршрутизации могут возникнуть две неприятные вещи: задержка распространения информации и перегрузка сети. Что такое задержка, должно быть, понятно всем. Например, рассмотрим, что такое максимальное время задержки оповещения всех маршрутизаторов, использующих дистанционно-векторный протокол. Каждый маршрутизатор должен получить информацию, обновить в соответствии с ней свою таблицу маршрутизации и затем направить эту информацию соседнему маршрутизатору. В объединенной сети с линейной структурой, содержащей  $N$  маршрутизаторов, для обновления информации требуется  $N$  этапов. Таким образом, чтобы обеспечить быстрое распространение информации, число  $N$  должно быть не очень большим.

Понятие перегрузки также очевидно. Поскольку каждый маршрутизатор, принимающий участие в обмене маршрутной информацией, должен рассыпать сообщения, наличие большого количества таких маршрутизаторов в сети увеличивает

объем трафика маршрутизации. Кроме того, если в маршрутных сообщениях содержится список возможных получателей, то размер каждого сообщения возрастает по мере увеличения количества маршрутизаторов и подключенных к ним сетей. Поэтому, чтобы при расширении сети поток данных маршрутизации оставался небольшим по сравнению с общим объемом трафика, необходимо ограничить размер маршрутных сообщений.

На самом деле большинство сетевых администраторов не располагают достаточно полной информацией, необходимой для детального анализа времени задержки оповещения и перегрузки в сети. Они просто придерживаются следующего простого эвристического правила.

*В один процесс маршрутизации в глобальной сети не стоит вовлекать больше 10–12 маршрутизаторов. Что касается локальной сети, то в ней эту цифру можно смело увеличить в 5 раз.*

Конечно, это правило дает лишь общие указания, и из него существует много исключений. Например, если используемое сетевое оборудование позволяет передавать пакеты с низкой задержкой и высокой пропускной способностью, количество задействованных маршрутизаторов может быть и больше. И наоборот, на слабом сетевом оборудовании или при большом трафике в сети, количество задействованных маршрутизаторов должно быть меньше, чтобы предотвратить перегрузку сетей дополнительным трафиком маршрутизации.

Поскольку объединенная сеть не является статичной, у администраторов могут возникать трудности при подсчете объема трафика, создаваемого в процессе маршрутизации, а также при определении процента пропускной способности сети, занимаемой трафиком маршрутизации. Например, со временем обычно возрастает количество машин, подключенных к сети, что увеличивает общий трафик в ней. Кроме того, к увеличению трафика может привести также использование новых прикладных программ. Таким образом, сетевые администраторы при выборе структуры системы маршрутизации не могут полностью полагаться только на приведенные выше рекомендации. Поэтому они, как правило, устанавливают в сети ту или иную *систему наблюдения за трафиком*. В сущности, программа-монитор трафика выполняет пассивное прослушивание сети и регистрирует статистические данные о проходящих в ней потоках данных. В частности, монитор может определить как степень использования сети (т.е. процент использования пропускной полосы сети), так и процентное количество пакетов, в которых передаются сообщения, относящиеся к протоколу маршрутизации. В результате администратор может вывести тенденции изменения трафика в сети, анализируя информацию, полученную за длительный период времени (например, неделю или месяц). С помощью полученных данных можно определить, не слишком ли много маршрутизаторов вовлечено в процесс маршрутизации.

## 15.4. Проблема дополнительных переходов

Хотя количество маршрутизаторов, задействованных в одном процессе обмена маршрутной информацией, необходимо ограничить, это может привести к ощутимым последствиям, поскольку в результате некоторые маршрутизаторы окажутся за пределами группы. На первый взгляд может показаться, что маршрутизатор, не включенный в группу, может проложить стандартный маршрут к одному из членов группы. В самом деле, на заре развития Internet, ядро сети выполняло функции центрального устройства маршрутизации, к которому вто-ростепенные маршрутизаторы отсылали дейтаграммы для дальнейшей доставки. Однако разработчики извлекли для себя важный урок: если не входящий в группу маршрутизатор прокладывает к одному из членов этой группы стандартный

маршрут, процесс маршрутизации будет частично оптимальным. Указанная проблема не зависит от количества маршрутизаторов или размера глобальной сети. Проблема может возникать каждый раз, когда незадействованное в процессе обмена маршрутной информацией устройство использует задействованное устройство для доставки сообщения. Чтобы понять, почему так происходит, рассмотрим пример, показанный на рис. 15.1.



*Рис. 15.1. Структура системы маршрутизации, при которой может возникнуть проблема дополнительных переходов. Неоптимальная маршрутизация происходит, когда подключенное к магистрали незадействованное в процессе обмена маршрутной информацией устройство прокладывает стандартный маршрут к задействованному маршрутизатору*

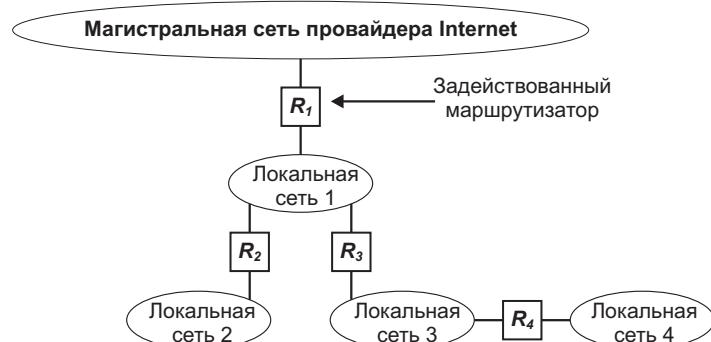
На рис 15.1 маршрутизаторы  $R_1$  и  $R_2$  подключены к локальным сетям 1 и 2, соответственно. Поскольку они принимают участие в процессе обмена маршрутной информацией, им обоим известно, как получить доступ к локальным сетям. Предположим, что незадействованный в процессе обмена информацией маршрутизатор  $R_3$  выбирает один из задействованных маршрутизаторов, например  $R_1$  в качестве стандартного. Это означает, что маршрутизатор  $R_3$  отсылает маршрутизатору  $R_1$  все дейтаграммы, предназначенные для тех сетей, к которым он непосредственно не подключен. В частности, маршрутизатор  $R_3$  отсылает через магистраль дейтаграммы, предназначенные для сети 2, маршрутизатору  $R_1$ , который затем должен переправить их обратно через магистраль к маршрутизатору  $R_2$ . Конечно, для достижения оптимальной маршрутизации требуется, чтобы маршрутизатор  $R_3$  передавал дейтаграммы, предназначенные для сети 2, непосредственно маршрутизатору  $R_2$ . Обратите внимание, что не имеет значения, какое именно из задействованных в процессе обмена маршрутной информацией устройств будет выбрано. Очевидно, что оптимальные маршруты будут проложены только к тем получателям, которые находятся в локальной сети, непосредственно подключенной к выбранному маршрутизатору. Все остальные маршруты, которые проходят к получателям через другой магистральный маршрутизатор, — неоптимальные. В последнем случае дейтаграммы будут повторно проходить по магистральной сети к нужному маршрутизатору, что является излишним. Также обратите внимание, что задействованные маршрутизаторы не могут с помощью ICMP-сообщения о перенаправлении сообщить маршрутизатору  $R_3$  о том, что он располагает информацией о неоптимальных маршрутах. Причина в том, что такие сообщения могут быть посланы только исходному отправителю дейтаграммы, а не ее промежуточному маршрутизатору.

Назовем аномалию в маршрутизации, показанную на рис. 15.1, *проблемой дополнительных переходов*. Эту проблему довольно трудно выявить, так как на первый взгляд система хорошо функционирует, и дейтаграммы доходят до получателей. Все же, поскольку маршрутизация не является оптимальной, вся система чрезвычайно неэффективна. Каждая совершающая дополнительный переход дейтаграмма потребляет ресурсы на промежуточном маршрутизаторе, а также вдвое повышает трафик в магистральной сети. Чтобы решить эту проблему, необходимо пересмотреть понятие о структуре маршрутизации.

*Использование группы маршрутизаторов, задействованных в процессе обновления маршрутной информации в качестве стандартного средства доставки, может создать дополнительный переход при пересылке программы конечному получателю. Поэтому возникает необходимость в создании механизма, который бы позволил незадействованным в процессе обмена маршрутной информации устройствам получить ее от задействованных маршрутизаторов и выбрать оптимальные из предоставленных маршрутов.*

## 15.5. Скрытые сети

Прежде чем приступить к изучению механизма, позволяющего маршрутизатору, который находится за пределами группы, получить маршрутную информацию, необходимо иметь представление о другом аспекте маршрутизации — так называемых скрытых сетях (т.е. сетях, информация о которых скрыта от маршрутизаторов группы). На рис. 15.2 показан пример, позволяющий проиллюстрировать это понятие.



*Рис. 15.2. Пример нескольких сетей и маршрутизаторов, подключенных к одной магистрали. В подобной структуре возникает необходимость в механизме передачи в ядро системы маршрутизации данных о достижимости добавляемых локальных сетей*

Как показано на рис. 15.2, сетевой центр состоит из нескольких локальных сетей, подключенных друг к другу посредством маршрутизаторов. Предположим, что в состав сетевого центра только что введена локальная сеть 4, которой выделен отдельный IP-адрес (для определенности предположим, что адрес сети выделил провайдер услуг Internet, которому не принадлежит магистральная сеть). Также предположим, что каждый из маршрутизаторов  $R_2$ ,  $R_3$  и  $R_4$  располагает правильной маршрутной информацией обо всех четырех локальных сетях данного центра. Кроме того, у этих маршрутизаторов существует также стандартный маршрут, согласно которому весь внешний трафик направляется к маршрутизатору  $R_1$  поставщика услуг Internet. Машины, непосредственно подключенные к локальной сети 4, могут напрямую обмениваться информацией, и компьютеры этой сети могут отсылать пакеты другим сетевым узлам в Internet. Однако, поскольку маршрутизатор  $R_1$  подключен только к локальной сети 1, он не располагает сведениями о локальной сети 4. Считается, что с точки зрения системы маршрутизации провайдера Internet, локальная сеть 4 скрыта за локальной сетью 1. Здесь напрашивается следующий важный вывод.

*Поскольку отдельная организация может иметь любое количество сетей, соединенных между собой посредством маршрутизаторов, очевидно, что ни один маршрутизатор, принадлежащий другой организации, не может непосредственно быть подключен сразу ко всем сетям отдельной организации. Таким образом, необходим механизм, который бы позволял незадействованным в процессе обмена маршрутной информацией устройствам уведомить другую группу маршрутизаторов о своих скрытых сетях.*

Теперь становится понятен основной принцип маршрутизации: *информация должна проходить в двух направлениях*. Другими словами, маршрутная информация должна проходить от группы задействованных маршрутизаторов до незадействованного маршрутизатора, а последний, в свою очередь, должен передать информацию о скрытых сетях обратно группе. В идеальном случае обе проблемы должны решаться с помощью одного механизма, создать который непросто. Здесь возникают две неявные проблемы: ответственность за предоставление информации и производительность. Выражаясь точнее, в каком месте иерархии должно располагаться устройство, отвечающее за предоставление группе маршрутизаторов информации о скрытых сетях? Если принять решение, что группу должен информировать один из незадействованных маршрутизаторов, то какой именно сможет это сделать? Обратимся снова к примеру. Маршрутизатор  $R_4$  ближе всех подключен к сети 4, но он находится на расстоянии двух переходов от ближайшего базового маршрутизатора. Таким образом, процесс доставки пакетов для сети 4 зависит не только от маршрутизатора  $R_4$ , но и от работы маршрутизатора  $R_3$ . Смысл заключается в том, что маршрутизатору  $R_4$  известно все о локальной сети 4, но он не может получать дейтаграммы непосредственно от маршрутизатора  $R_1$ . Маршрутизатор  $R_3$  расположен на расстоянии одного перехода от базового маршрутизатора  $R_1$  и может обеспечить доставку пакетов в Internet, но он не подключен непосредственно к локальной сети 4. Таким образом, кажется нецелесообразным возлагать на маршрутизатор  $R_3$  ответственность за предоставление информации о сети 4. Чтобы решить эту дилемму, необходимо ввести новое понятие. Оно будет рассмотрено в следующих разделах; там же будет описан протокол, необходимый для осуществления задуманного.

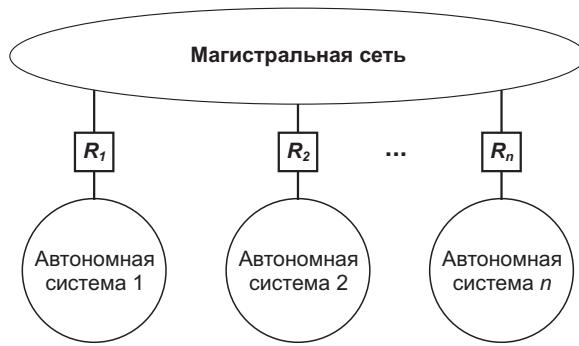
## 15.6. Понятие автономной системы

Разногласия по поводу того, какой именно маршрутизатор должен сообщать группе базовых маршрутизаторов информацию о скрытой локальной сети, возникают из-за того, что нами был рассмотрен только принцип действия системы маршрутизации в объединенной сети, а не административные вопросы. Структура соединений, образующаяся в результате усложнения структуры объединенной сети (рис. 15.2), не должна рассматриваться как несколько независимых сетей, подключенных к объединенной сети. Напротив, структура сети должна рассматриваться с точки зрения одной организации, в состав которой входит несколько сетей, управляемых из единого центра. Поскольку сети и маршрутизаторы находятся под контролем единого административного органа, гарантируется, что все внутренние маршруты остаются жизнеспособны и непротиворечивы. Кроме того, администрация может назначить один из своих маршрутизаторов в качестве машины, которая будет информировать внешний мир о всех внутренних сетях организации. Снова обратимся к примеру, показанному на рис. 15.2. Благодаря тому, что маршрутизаторы  $R_2$ ,  $R_3$  и  $R_4$  находятся под контролем одного административного органа, он может возложить на маршрутизатор  $R_3$  обязанности сообщать информацию о сетях 2, 3 и 4 во внешний мир (заметьте, что маршрутизатору  $R_1$  уже известно о сети 1, так как к ней он подключен непосредственно).

В контексте маршрутизации, группа сетей и маршрутизаторов, находящихся под контролем одного административного органа, называется *автономной системой* (AC). В пределах одной автономной системы маршрутизаторы могут по своему усмотрению выбирать механизм для обнаружения, распространения, утверждения и проверки согласованности маршрутов. Обратите внимание, что, согласно этому определению, система базовых маршрутизаторов оригинальной сети Internet представляла собой первую автономную систему. Любое изменение в протоколах маршрутизации в ней происходило без влияния на методы маршрутизации, используемые в других автономных системах. В предыдущей главе было отмечено, что в первоначальной системе базовой маршрутизации в Internet для обмена данными между устройствами использовался протокол GGP. В более поздних версиях системы маршрутизации использовался протокол SPREAD. В конечном счете, провайдеры Internet создали собственные магистральные сети, в которых использовались более новые протоколы. В следующей главе рассмотрены несколько протоколов, которые используются для распространения маршрутной информации в пределах автономной системы.

## 15.7. От ядра сети к независимым автономным системам

Теоретически идея создания автономной системы естественным образом вытекала из обобщения первоначальной структуры сети Internet. Если в структуре сети, показанной на рис. 14.1, заменить локальные сети на автономные системы, то получится топология, изображенная на рис. 15.3.



*Рис. 15.3. Структура объединенной сети, автономные системы которой подключены к сетевой магистрали. Каждая автономная система состоит из нескольких сетей и маршрутизаторов и управляетется одним административным органом*

Чтобы из любой точки сети Internet обеспечить доступ к сетям, скрытым внутри автономной системы, последняя должна сообщить о своих сетях другим автономным системам. Более того, это сообщение может быть отослано любой другой автономной системе. Однако в централизованной структуре крайне важно, чтобы каждая автономная система анонсировала информацию одному из маршрутизаторов, находящихся в автономной системе ядра сети.

Может показаться, что приведенное выше определение автономной системы довольно расплывчено. Однако на практике границы между автономными системами должны быть четко обозначены для того, можно было принимать решения о маршрутизации на основе автоматизированных алгоритмов. Например, автономная система, которая принадлежит корпорации, может принять решение не

отправлять пакеты через автономную систему, принадлежащую другой корпорации, даже если между ними существует прямое соединение. Для того чтобы автоматизированные алгоритмы маршрутизации могли различать автономные системы, каждой из них присваивается *номер*. Это осуществляется центральным административным органом, в обязанности которого входит назначение всех сетевых адресов в Internet. Если маршрутизаторы двух автономных систем обмениваются маршрутной информацией, то, согласно протоколу, в сообщения включается номер автономной системы, которую представляет каждый из маршрутизаторов. Подведем итог всему сказанному выше.

*В большой объединенной сети на основе протокола TCP/IP вводится дополнительная структура, определяющая административные границы сети. Суть ее состоит в том, что любой набор сетей и маршрутизаторов, управляемых одним административным органом, считается единой автономной системой. Администрация автономной системы может по своему усмотрению выбирать структуру внутренней системы маршрутизации и используемые при этом протоколы.*

Выше уже было отмечено, что в пределах одной автономной системы собирается информация о всех внутренних сетях и назначается один или несколько маршрутизаторов, ответственных за передачу этой информации другим автономным системам. В следующих разделах представлено детальное описание протокола, который используют маршрутизаторы для анонсирования достижимости своих сетей. В конце главы мы снова вернемся к рассмотрению структурных вопросов и обсудим одно важное ограничение, налагаемое на маршрутизацию структурой автономной системы.

## 15.8. Протокол внешнего шлюза

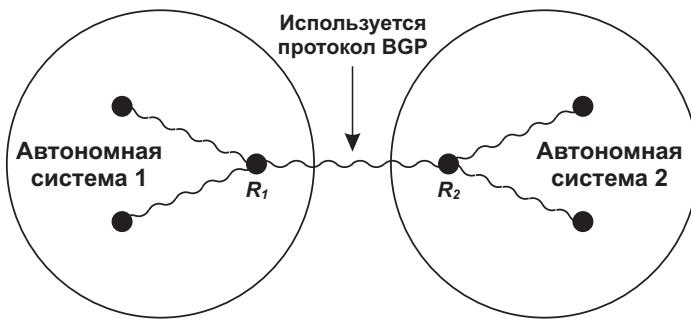
Специалисты, занимающиеся компьютерными технологиями, используют термин *протокол внешнего шлюза (EGP<sup>1</sup> — Exterior Gateway Protocol)* для обозначения протокола, который используется для передачи маршрутной информации между двумя автономными системами. В настоящее время в большинстве объединенных сетей на основе протокола TCP/IP используется один внешний протокол. Этот протокол, называемый *протоколом пограничного шлюза (BGP — Border Gateway Protocol)*, за время своей эволюции пересматривался четыре раза (причем каждый раз довольно основательно). Каждая версия имеет свой номер, который входит в официальное название протокола. Последняя, четвертая версия этого протокола называется *BGP-4*. Поэтому далее в книге, если не будет оговорено особо, под термином *BGP* мы будем понимать протокол *BGP-4*.

Для того чтобы две автономные системы могли обмениваться маршрутной информацией, в каждой из них должен быть выделен маршрутизатор<sup>2</sup>, который будет взаимодействовать по протоколу BGP с равным ему по рангу маршрутизатором другой автономной системы. Говорят, что эти два маршрутизатора составляют одноранговую BGP-систему (*BGP peers*). Поскольку маршрутизатор одной

<sup>1</sup> Первоначально аббревиатура EGP применялась для обозначения протокола, с помощью которого взаимодействовали машины, находящиеся внутри ядра сети Internet. Название этого протокола укоренилось, когда вместо термина *маршрутизатор (router)* использовался термин *шлюз (gateway)*.

<sup>2</sup> Хотя для этой цели подойдет любой компьютер, в большинстве автономных систем реализации протокола BGP запускаются именно на маршрутизаторе. Поэтому при рассмотрении всех примеров в данной книге мы будем подразумевать, что программа протокола BGP запущена на маршрутизаторе.

автономной системы должен по протоколу BGP обмениваться информацией с равным ему по рангу маршрутизатором другой автономной системы, целесообразно так выбрать эти машины, чтобы они находились как можно ближе к “краю” автономной системы. Поэтому в терминологии протокола BGP эти машины называются *пограничными шлюзами* (*border gateway*), или *пограничными маршрутизаторами* (*border router*). Эта идея изображена на рис. 15.4.



*Рис. 15.4. Концептуальная схема взаимодействия двух маршрутизаторов  $R_1$  и  $R_2$  по протоколу BGP. Получив от других маршрутизаторов информацию о внутренних сетях своей автономной системы, маршрутизаторы  $R_1$  и  $R_2$  обмениваются ею друг с другом. Маршрутизаторы, использующие протокол BGP, как правило, располагаются как можно ближе к внешней “границе” автономной системы.*

На рис 15.4 маршрутизатор  $R_1$  собирает информацию о сетях в автономной системе 1 и сообщает эту информацию маршрутизатору  $R_2$ , используя протокол BGP, в то время как маршрутизатор  $R_2$  сообщает информацию об автономной системе 2 маршрутизатору  $R_1$ .

## 15.9. Особенности протокола BGP

Протокол BGP — не совсем обычный протокол маршрутизации. Самое важное заключается в том, что его нельзя до конца отнести ни к классу дистанционно-векторных, ни к классу протоколов маршрутизации на основе состояния соединения. Ниже приведены характерные отличия протокола BGP от других протоколов маршрутизации.

- *Коммуникация между автономными системами.* Поскольку протокол BGP относится к протоколам внешнего шлюза, его основное назначение — обеспечить обмен информацией между двумя автономными системами.
- *Координирование работы нескольких спикеров BGP.* Если в состав автономной системы входит несколько маршрутизаторов, каждый из которых обменивается информацией с равным ему по рангу маршрутизатором внешней автономной системы (их называют *спикерами BGP*), протокол BGP может использоваться для координации работы всего набора маршрутизаторов. Это гарантирует, что маршрутизаторы распространяют непротиворечивую информацию.
- *Распространение информации о достижимости.* Протокол BGP позволяет автономной системе сообщить во внешний мир информацию о расположенных в ней получателях, а также о тех получателях, доступ к которым осуществляется через данную автономную систему. Кроме того, с помощью

протокола BGP подобную информацию можно получить от других автономных систем.

- *Принцип ближайшего перехода.* Подобно дистанционно-векторным протоколам маршрутизации, протокол BGP предоставляет информацию об адресе ближайшей точки перехода для каждого получателя.
- *Поддержка различной политики маршрутизации.* В отличии от многих дистанционно-векторных протоколов, которые сообщают только ту маршрутную информацию, которая находится в локальной таблице маршрутизации, протокол BGP обеспечивает различную политику маршрутизации, в зависимости от выбора администратора. В частности, маршрутизатор, работающий под управлением протокола BGP, можно настроить так, чтобы он различал получателей, доступ к которым осуществляется через компьютеры его автономной системы, и получателей, анонсированных другими автономными системами.
- *Надежный транспортный протокол.* Протокол BGP отличается от других протоколов, передающих информацию о маршрутизации тем, что он предполагает использование надежного транспортного протокола. Таким образом, для обмена информацией в протоколе BGP используется исключительно транспортный протокол TCP/IP.
- *Информация о маршруте.* Кроме указания списка возможных получателей и адреса ближайшей точки перехода для каждого из них, в сообщении протокола BGP анонсируется также маршрутная информация, которая позволяет узнать, через какие автономные системы проложен маршрут к конкретному получателю.
- *Передача обновлений.* Чтобы не создавать дополнительную нагрузку на сеть, в каждом сообщении протокола BGP об обновлении не передается полная маршрутная информация. Вместо этого обмен полной информацией происходит только один раз, а в следующих сообщениях передаются только изменения, которые называются *дельтами* (*deltas*).
- *Поддержка бесклассовой адресации.* Протокол BGP поддерживает CIDR-адреса. Это означает, что программа протокола BGP не полагается на методы идентификации IP-адресов, а вместе с каждым адресом отсылает и его маску.
- *Объединение маршрутов.* Чтобы не создавать дополнительной нагрузки на сеть, протокол BGP позволяет отправителю накапливать информацию о маршрутах и отсыпать в одном пакете данные сразу о нескольких, связанных между собой получателях.
- *Аутентификация.* Протокол BGP позволяет получателю удостоверить подлинность сообщений (т.е. подтвердить “личность” отправителя).

## 15.10. Функции протокола BGP и виды сообщений

В процессе взаимодействия по протоколу BGP выполняется три основных действия. Первое — получение согласия сторон на взаимодействие по протоколу BGP и аутентификацию. При этом два равноправных маршрутизатора устанавливают соединение по протоколу TCP и обмениваются сообщениями, которые подтверждают, что обе стороны согласны вступить в процесс обмена информацией. Второе действие составляет основу протокола: каждая сторона отсылает информацию о доступности или недоступности получателей. Это означает, что отправитель может сообщить о возможности доступа к одной или нескольким сетям

получателя (при этом указывается адрес ближайшей точки перехода для каждой сети) или, напротив, может заявить, что одна или несколько сетей, о которых сообщалось раньше, более недоступны. Третье действие — осуществление постоянного контроля над правильностью функционирования взаимодействующих пар маршрутизаторов и сетевых соединений.

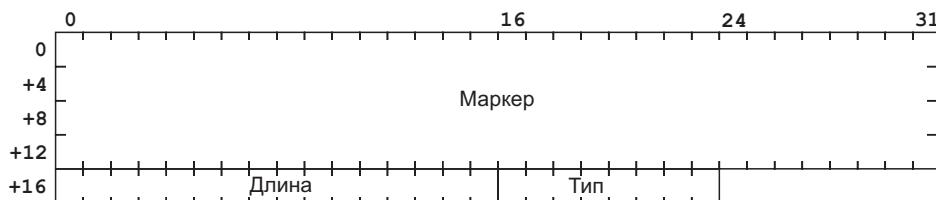
Для выполнения трех описанных выше действий в протоколе BGP определено четыре основных типа сообщений, которые приведены в табл. 15.1.

**Таблица 15.1. Основные типы сообщений протокола BGP-4**

Код типа сообщения	Тип сообщения	Выполняемые действия	Описание
1	OPEN	Открыть	Инициализирует процесс взаимодействия
2	UPDATE	Обновить	Анонсирует или аннулирует маршрутную информацию
3	NOTIFICATION	Известить	Ответ на неверное сообщение
4	KEEPALIVE	Проверить	Выполняется активная проверка возможности соединения между BGP-парами

## 15.11. Заголовок BGP-сообщения

В начале каждого сообщения протокола BGP расположен заголовок фиксированного формата, с помощью которого определяется тип сообщения. Формат заголовка показан на рис. 15.5.



*Рис. 15.5. Формат заголовка, который помещается перед каждым сообщением протокола BGP*

В поле маркера (его длина составляет 16 октетов) заносится значение, которое обе стороны “договорились” использовать в качестве метки начала сообщения. В поле длины, размер которого составляет 2 октета, указывается общая длина сообщения в октетах. Минимальный размер сообщения составляет 19 октетов (для типа сообщения, в котором после заголовка нет данных). Максимально допустимая длина сообщения составляет 4096 октетов. И наконец, в поле типа, длина которого составляет 1 октет, заносится одно из четырех значений, определяющих тип сообщения, указанных в табл. 15.1.

Наличие поля маркера является нехарактерным для сетевых протоколов. В исходном сообщении маркер состоит из всех единиц. Если взаимодействующие между собой маршрутизаторы “договорятся” об использовании механизма аутентификации, в поле маркера может содержаться информация об аутентификации. В любом случае обе стороны должны согласовать, какое значение будет внесено в это поле, чтобы его можно было в дальнейшем использовать для выполнения синхронизации. Чтобы понять, зачем нужна синхронизация, вспомним, что обмен всеми типами сообщений в протоколе BGP происходит через потоковую

транспортную службу (т.е. протокол TCP), в которой невозможно определить, где заканчивается одно сообщение и начинается другое. В такой среде простая ошибка, произошедшая на стороне одного из участников соединения, может иметь фатальные последствия. В частности, если отправитель или получатель неправильно подсчитает количество октетов в сообщении, произойдет *ошибка синхронизации*. Хуже всего то, что поскольку с помощью транспортного протокола невозможно определить границы сообщения, получатель так никогда и не узнает об ошибке. Таким образом, чтобы обеспечить синхронные действия отправителя и получателя, программа протокола BGP помещает в начало каждого сообщения хорошо известную обеим сторонам последовательность октетов и перед дальнейшей обработкой сообщения требует от получателя подтвердить, что данное значение не повреждено.

## 15.12. Сообщение об открытии BGP-соединения

Как только два взаимодействующих по протоколу BGP маршрутизатора установили TCP-соединение, каждый из них отсылает сообщение об открытии, в котором указывает номер своей автономной системы и устанавливает текущие параметры. Помимо стандартного заголовка, в сообщении об открытии соединения указывается значение *времени удержания*. Это время определяет максимальное количество секунд, которые могут пройти между успешным получением двух сообщений. Формат сообщения об открытии BGP-соединения показан на рис. 15.6.

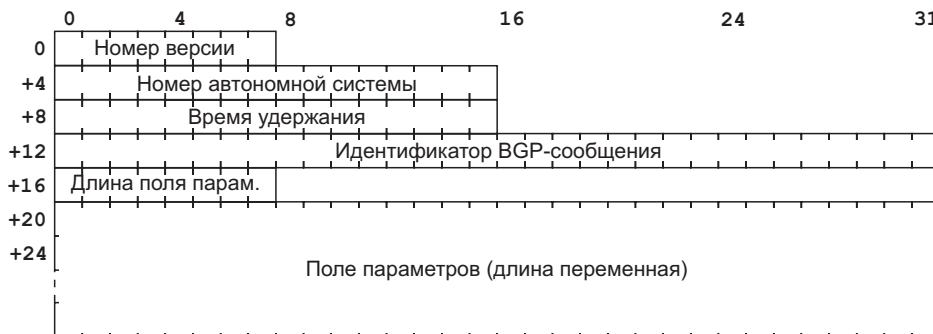


Рис. 15.6. Формат начального сообщения об открытии BGP-соединения, которое отсылается после установки TCP-соединения. Эти данные помещаются после стандартного заголовка BGP-сообщения

Назначение большинства полей — достаточно очевидно. Поле *номер версии* используется для идентификации используемой версии протокола (на рис. 15.6 показан формат сообщения для версии BGP-4). Вспомним, что каждой автономной системе присваивается свой уникальный номер. В поле *номера автономной системы* указывается номер автономной системы отправителя. В поле *времени удержания* определяется максимальный промежуток времени, в течение которого получатель может получить сообщение от отправителя. Установить таймер, используя это значение, должен получатель. Таймер сбрасывается каждый раз, когда приходит сообщение. Если в течение заданного промежутка времени от отправителя не получено сообщение, получатель делает вывод, что отправитель недоступен и прекращает отправку дейтаграмм по маршруту, указанному отправителем.

В поле *идентификатора BGP-сообщения* указывается целое 32-битовое число, которое однозначно определяет отправителя. Если машина задействована в нескольких BGP-соединениях (например, при обмене маршрутной информацией с

несколькими автономными системами), эта машина должна использовать один и тот же идентификатор для всех сеансов. В спецификации протокола определено, что в качестве идентификатора используется IP-адрес. Таким образом, маршрутизатор должен выбрать один из своих IP-адресов, который он будет использовать в качестве идентификатора сеанса связи с соответствующими маршрутизаторами по протоколу BGP.

Последнее поле параметров в сообщении об открытии BGP-соединения является необязательным. Если оно присутствует, то его длина в октетах помещается в соответствующее поле сообщения (на рис. 15.6 оно обозначено как *Длина поля параметров*). При этом само поле содержит список параметров. Поле параметров имеет переменную длину, поэтому его размер меняется от сообщения к сообщению. Если параметры присутствуют, каждому параметру в списке предшествует заголовок, состоящий из 2 октетов: первый — определяет тип параметра, а второй — длину самого параметра. Если параметры не указаны, то в поле длины параметров помещается нулевое значение, и текст сообщения на этом заканчивается.

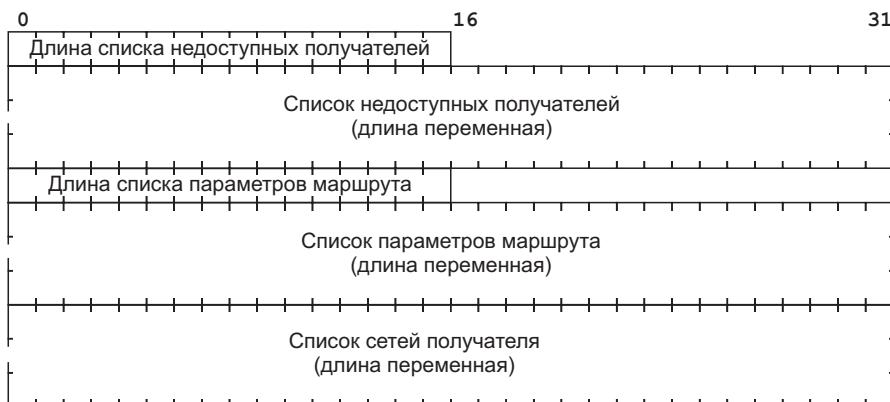
В первоначальном стандарте зарезервирован только один тип параметра под номером 1, который предназначен для аутентификации. Параметр аутентификации начинается с заголовка, определяющего тип аутентификации, после которого следуют соответствующие этому типу данные. Параметр аутентификацииведен для того, чтобы взаимодействующие по протоколу BGP маршрутизаторы могли самостоятельно выбирать механизм аутентификации. При этом сам механизм не должен являться частью стандарта BGP.

Получив сообщение об открытии BGP-соединения, машина-спикер должна ответить на него сообщением о поддержке соединения в активном состоянии (KEEPALIVE) (этот тип сообщения будет рассмотрен ниже). Прежде чем стороны смогут обменяться маршрутной информацией, каждый из маршрутизаторов должен послать сообщение об открытии соединения и получить на него ответ о возможности поддержки соединения в активном состоянии. Таким образом, сообщение о поддержке соединения является своего рода подтверждением получения сообщения об открытии.

### 15.13. BGP-сообщение об обновлении

Этот тип сообщения используется после того, как взаимодействующие по протоколу BGP стороны установили друг с другом TCP-соединение, отослали сообщения об открытии BGP-соединения и получили на них сигналы подтверждения приема. Сообщение об обновлении (UPDATE) позволяет анонсировать противоположной стороне о появившихся новых получателях, а также отменить предыдущие анонсы, если получатели стали недоступны. Формат сообщения об обновлении изображен на рис. 15.7.

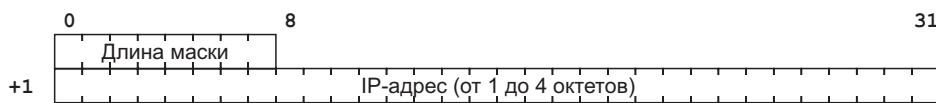
Как показано на рис. 15.7 каждое сообщение об обновлении разделено на две части: в первой части перечисляются анонсированные ранее получатели, которые стали недоступны, а во второй части указывается список новых анонсируемых получателей. Все списки имеют переменную длину, поэтому, если для некоторого сообщения об обновлении в информации заданного типа нет необходимости, соответствующее поле не включается в сообщение. Первыми в сообщении об обновлении указываются недоступные получатели, информация о которых аннулируется. Перед списком в специальном поле длиной 2 октета указывается его длина. Если в данном сообщении ни один из получателей не аннулируется, в поле длины списка помещается нулевое значение. Аналогично, в поле под названием *Длина списка параметров маршрута* определяется размер *атрибутов маршрута*, относящихся к новым получателям, о которых анонсирует маршрутизатор. Если новых получателей нет, в это поле помещается нулевое значение.



*Рис. 15.7. Формат BGP-сообщения об обновлении. Обратите внимание, что части сообщения, имеющие переменный размер, могут быть опущены. Эти данные помещаются после стандартного заголовка BGP-сообщения*

### 15.14. Сжатое представление пары “маска-адрес”

Как в списке недоступных получателей, так и в списке анонсируемых сетей получателя указываются IP-адреса сетей. Для поддержки бесклассовой адресации в протоколе BGP для каждого IP-адреса должна быть указана его маска. Как известно, длина IP-адреса и длина IP-маски составляют 32-бита. Поэтому, чтобы уменьшить размер сообщения, в протоколе BGP используется сжатое представление пары “маска-адрес” (рис. 15.8).



*Рис. 15.8. Сжатый формат, используемый в протоколе BGP для хранения адреса получателя и соответствующей ему маски*

Как следует из рис. 15.8, на самом деле в протоколе BGP битовая маска нигде не указывается — значение маски кодируется в одном октете, который указывается перед каждым IP-адресом. Октет маски содержит 8-разрядное двоичное целое число, которое определяет количество битов в маске (подразумевается, что биты в маске должны следовать непрерывно один за другим). IP-адрес, указываемый следом за октетом маски, также сжимается. Из него исключается поле адреса узла, определяемое маской (т.е. оставляется только поле адреса сети). Таким образом, если значение поля длины маски меньше или равно 8, то следом за ним указывается только один октет IP-адреса сети. Если значение поля длины маски находится в интервале от 9 до 16 — оставляется два октета IP-адреса, от 17 до 24 — три октета, и от 25 до 32 — IP-адрес указывается полностью. Интересно, что в этом стандарте значение поля длины маски может равняться нулю (при этом октеты IP-адреса не указываются вовсе). Нулевая длина маски также представляет очень полезную информацию, поскольку она соответствует стандартному маршруту следования пакетов.

## 15.15. Параметры маршрута протокола BGP

Как уже было сказано, протокол BGP нельзя считать истинным дистанционно-векторным протоколом, поскольку, кроме адреса ближайшей точки перехода, он анонсирует дополнительные данные. Дополнительная информация указывается в списке параметров маршрута сообщения об обновлении. Отправитель может использовать параметры маршрута для того, чтобы определить адрес ближайшей точки перехода для анонсированных получателей либо указать список автономных систем, расположенных по пути следования к получателям. С помощью параметров маршрута можно также определить, откуда именно была получена информация (от другой автономной системы или от источника, находящегося в пределах автономной системы отправителя).

Обратите внимание — параметры маршрута указываются так, чтобы уменьшить размер сообщения об обновлении. Это означает, что выбранные параметры относятся ко всем получателям, которые анонсируются в сообщении. Поэтому если нужно указать параметры, которые относятся только к определенной группе получателей, о них необходимо объявить в отдельном сообщении об обновлении.

Параметры маршрута очень важны для протокола BGP по трем причинам. Во-первых, информация о маршруте позволяет получателю проверить наличие маршрутных петель. При этом отправитель может установить точный маршрут следования к получателю и определить номера автономных систем, через которые проходят пакеты. Если какая-либо из автономных систем перечислена в списке более одного раза, это указывает на вероятность существования маршрутной петли. Во-вторых, информация о маршруте позволяет получателю установить определенные ограничения на уровне административной политики. Например, получатель может изучить маршруты следования пакетов и убедиться, что они не проходят через ненадежные автономные системы (например, через автономную систему конкурента). В третьих, информация о маршруте позволяет получателю узнать об источнике информации для всех маршрутов. При этом отправитель может определить, пришла ли информация из его автономной системы или от другой системы. Параметры маршрута позволяют также отправителю узнать, каким образом была собрана маршрутная информация: с помощью протокола внешнего шлюза (наподобие BGP), или с помощью протокола внутреннего шлюза (о нем речь пойдет в следующей главе). В результате каждый получатель сможет решить, стоит ли доверять маршрутной информации, полученной из внешних автономных систем (т.е. тех систем, к которым не принадлежат взаимодействующие по протоколу BGP маршрутизаторы).

По идеи в поле *параметров маршрута* должен содержаться список элементов, каждый из которых представлен тремя значениями:

(тип, длина, значение)

Таким образом, вместо полей фиксированного размера, разработчики выбрали гибкую систему кодирования, сокращающую до минимума пространство, которое занимает каждый из элементов. Как показано на рис. 15.9, информация о типе всегда занимает два октета, но размер остальных полей может быть разным. Описание битов поля типа приведено в табл. 15.2.

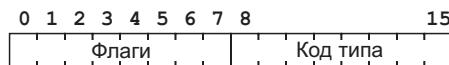


Рис. 15.9. Формат поля типа элемента параметра маршрута, размер которого составляет 2 октета. Это поле расположено всегда первым в начале каждого элемента

---

**Таблица 15.2. Описание битов поля типа элемента параметра маршрута**

---

<i>Номер бита</i>	<i>Описание</i>
0	0 — если атрибут обязательный, 1 — если нет
1	1 — если атрибут транзитивный, 0 — если нет
2	0 — если атрибут полный, 1 — если частичный
3	0 — если длина поля составляет один октет, 1 — если два
5–7	Не используются (должны быть нулевыми)

---

Для каждого элемента списка параметров маршрута после поля типа, размером 2 октета, следует поле длины атрибута, состоящее из одного или двух октетов. Размер поля длины определяется значением третьего бита поля типа (см. табл. 15.2 и рис. 15.9). Получатель использует поле типа, чтобы определить размер поля длины, а затем использует содержимое поля длины, чтобы определить длину атрибута.

Каждый элемент поля параметров маршрута может иметь один из семи возможных кодов типа (табл. 15.11).

---

**Таблица 15.11. Коды типа параметров маршрута протокола BGP**

---

<i>Код типа</i>	<i>Описание</i>
1	Определяет источник информации о маршруте
2	Список автономных систем, расположенных по маршруту к получателю
3	Адрес ближайшей точки перехода, используемой для достижения получателя
4	Дискриминатор, используемый для нескольких точек выхода АС
5	Предпочтение, используемое в пределах автономной системы
6	Признак завершения объединения маршрутов
7	Идентификатор автономной системы, которая выполнила объединение маршрутов

---

## **15.16. BGP-сообщение о поддержке соединения в активном состоянии**

Два взаимодействующих между собой по протоколу BGP маршрутизатора периодически обмениваются сообщениями о поддержке соединения в активном состоянии (KEEPALIVE) для того, чтобы проверить наличие сетевого соединения и убедиться, что обе стороны продолжают функционировать. Такое сообщение состоит только из стандартного заголовка без каких-либо дополнительных данных. Таким образом, общий размер сообщения составляет 19 октетов (минимальный размер BGP-сообщения).

Существует две причины, по которым в протоколе BGP используются сообщения о поддержке соединения в активном состоянии. Во-первых, периодический обмен сообщениями необходим, поскольку в протоколе BGP в качестве транспортного используется протокол TCP, а в протоколе TCP не предусмотрены средства для постоянной проверки доступности конечной точки соединения. Кроме того, если протокол TCP не может доставить данные, которые отсылает

определенная прикладная программа, он не информирует эту прикладную программу об ошибке передачи. Таким образом, не существует другого средства узнать, не произошел ли разрыв TCP-соединения, кроме как периодически посыпать тестовые сообщения. Во-вторых, по сравнению с другим типом сообщений, сообщения о поддержке соединения в активном состоянии практически не увеличивают поток данных в сети. Для проверки TCP-соединения во многих ранних протоколах маршрутизации использовался периодический обмен маршрутной информацией. Однако, поскольку маршрутная информация изменяется не так часто, редко изменяется и содержимое сообщения. Кроме того, поскольку сообщения о маршрутизации, как правило, имеют большой размер, повторная отправка одного и того же сообщения без надобности создает дополнительную нагрузку на сеть. Поэтому, чтобы избежать неэффективной работы системы, в протоколе BGP проверка наличия TCP-соединения осуществляется отдельно от пересылки сообщения об обновлении маршрутной информации (т.е. основных функций протокола). В результате участники BGP-соединения могут часто обмениваться короткими тестовыми сообщениями о поддержке соединения в активном состоянии. Большие по размеру сообщения об обновлении пересылаются только в тех случаях, когда изменяется информация о достижимости получателей.

Вспомним, что при открытии соединения спикер протокола BGP устанавливает значение таймера *удержания* (*hold timer*). Этот таймер определяет максимальное время, в течение которого программа протокола BGP может получить сообщение. В отдельных случаях значение интервала удержания может равняться нулю. Это означает, что сообщения о поддержке соединения в активном состоянии не будут использоваться. Если значение интервала удержания больше нуля, то в стандарте рекомендуется, чтобы за это время было послано не менее трех тестовых сообщений о поддержке соединения в активном состоянии. Однако в любом случае интервал между посылками тестовых пакетов в протоколе BGP не может быть меньше одной секунды. Следовательно, если значение интервала удержания больше нуля, то оно не может быть меньше трех секунд.

## 15.17. Информация для конечного получателя

В отличие от большинства других протоколов, с помощью которых распространяется маршрутная информация, протокол внешнего шлюза предназначен не только для анонсирования информации о наборе доступных получателей. Он должен предоставлять информацию, которая является правильной с точки зрения внешнего получателя. Здесь нужно рассмотреть два вопроса: методы административной политики и прокладывание оптимальных маршрутов. Вопрос о политике очевиден: маршрутизатору, находящемуся в пределах автономной системы, может быть разрешен доступ к данному получателю, тогда как маршрутизаторам, находящимся за пределами автономной системы, доступ к этому получателю может быть запрещен. Вопрос о маршрутизации означает, что маршрутизатор должен анонсировать адрес ближайшей точки перехода, которая является оптимальной с точки зрения внешнего получателя. Эта идея представлена на рис. 15.10.

На рис. 15.10 маршрутизатор  $R_2$  назначен в качестве спикера протокола BGP от имени автономной системы. Он должен сообщать во внешний мир о способе достижимости сетей 1–4. Однако, предоставляемая адрес ближайшей точки перехода, маршрутизатор  $R_2$  сообщает о том, что сеть 1 можно достичь через маршрутизатор  $R_1$ , сети 3 и 4 — через маршрутизатор  $R_3$ , а сеть 2 — через маршрутизатор  $R_2$ .

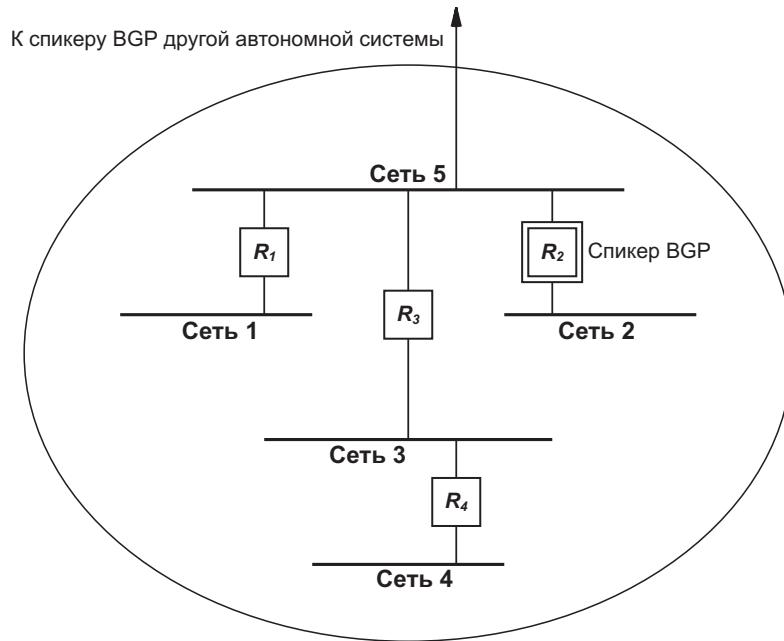


Рис. 15.10. Пример автономной системы. Маршрутизатор  $R_2$  работает под управлением протокола BGP и сообщает информацию, предназначенную для внешнего маршрутизатора, которая не обязательно должна содержаться в его таблице маршрутизации

## 15.18. Основное ограничение протоколов внешнего шлюза

Выше уже отмечалось, что, поскольку на протоколы внешнего шлюза могут налагаться административные ограничения, анонсируемые ими сети на самом деле могут представлять собой подмножество тех сетей, доступ к которым можно получить на основании предоставляемой маршрутной информации. Несмотря на это существует и более серьезное ограничение, которое налагается на процесс внешней маршрутизации.

*Протокол внешнего шлюза не обменивается метрикой расстояния и не интерпретирует ее, даже если такая метрика существует.*

Протоколы наподобие BGP позволяют спикеру объявить том, что некоторый получатель более недоступен, или предоставить список автономных систем, расположенных по маршруту, ведущему к получателю. Но они не могут передавать или сравнивать стоимости двух маршрутов, если информация об этих маршрутах не поступает из сетей, принадлежащих одной и той же автономной системе. В сущности, протокол BGP может только определить, существует ли маршрут к данному получателю, но он не может передать или вычислить кратчайший из двух маршрутов.

Теперь становится понятно, почему в протоколе BGP скрупулезно регистрируется источник информации, которая посыпается. Важно отметить следующее: когда маршрутизатор получает анонсы для одного и того же получателя от взаимодействующих по протоколу BGP маршрутизаторов, расположенных в разных автономных системах, он не может сравнивать стоимости маршрутов, проложенных через эти автономные системы. Таким образом, анонсировать доступность сети

с помощью протокола BGP равносильно тому, что сказать: “Маршрут к этой сети проходит через мою автономную систему”. Но BGP-маршрутизатор не может сказать: “Моя автономная система предоставляет лучший маршрут к этой сети, чем другая автономная система”.

Отсутствие возможности интерпретации метрики расстояния не позволяет использовать протокол BGP в качестве алгоритма маршрутизации. В частности, даже если маршрутизатору известны два маршрута к сети, он не может определить, какой из них короче, так как ему неизвестна стоимость маршрутов, пересекающих промежуточные автономные системы. Рассмотрим маршрутизатор, который использует протокол BGP для обмена информацией с двумя маршрутизаторами, находящимися в автономных системах  $r$  и  $f$ . Если маршрутизатор в автономной системе  $r$  анонсирует маршрут к данному получателю через автономные системы  $p$ ,  $q$  и  $r$ , а маршрутизатор в автономной системе  $f$  анонсирует маршрут к тому же получателю через автономные системы  $f$  и  $g$ , у получателя нет возможности сравнить длину этих двух маршрутов. Маршрут, ведущий через три автономные системы, может проходить через локальные сети в каждой из систем, в то время как маршрут, проходящий через две автономные системы, может быть проложен через несколько медленных последовательных каналов связи. Поскольку получатель не располагает полной информацией о маршрутах, он не может провести сравнительный анализ.

Поскольку в маршрутную информацию не входит метрика расстояния, маршрутизаторы автономной системы должны с особой тщательностью отбирать и сообщать только те маршруты, по которым должен следовать трафик. Формально можно сказать, что протокол внешнего шлюза является *протоколом информирования о достоверности сети*, а не протоколом маршрутизации. Итак, можно подвести итог сказанному.

*Поскольку протокол внешнего шлюза наподобие BGP только объявляет маршруты к сетям, получатель может установить административные ограничения на использование этой информации, но не может выбрать маршрут с наименьшей стоимостью. Поэтому отправитель должен анонсировать только те маршруты, по которым должен следовать трафик.*

Главный аспект здесь заключается в том, что в любой объединенной сети при использовании протокола BGP для предоставления маршрутной информации для внешних получателей последние должны либо полагаться на административную политику, либо предполагать, что пересечение любой автономной системы требует одинакового количества затрат. В результате безобидные на первый взгляд ограничения могут иметь непредсказуемые последствия.

1. Несмотря на то что протокол BGP позволяет анонсировать несколько маршрутов к одной сети, в нем не предусмотрена возможность одновременного использования нескольких маршрутов. Это означает, что в любой момент времени весь трафик, передаваемый от компьютера одной автономной системы в сеть другой автономной системы, будет проходит по одному маршруту, даже если между автономными системами существует несколько физических соединений. Обратите также внимание на то, что внешняя автономная система будет использовать только один обратный маршрут к отправителю, даже если системы отправителя разделяют исходящий трафик между двумя или более маршрутами. В результате скорость соединения и задержка передачи пакетов между парой машин в разных направлениях может быть различной. Это усложняет выявление и устранение ошибок в объединенной сети.

2. В протоколе BGP не поддерживается возможность распределения нагрузки между несколькими маршрутизаторами, соединяющими разные автономные системы. Если две автономные системы соединены друг с другом с помощью нескольких маршрутизаторов, естественно, возникает желание равномерно распределить трафик между ними. Протокол BGP позволяет автономным системам распределить нагрузку по сетям (например, разбить автономную систему на несколько подсетей и поручить разным маршрутизаторам анонсировать информацию о своих подсетях). Однако в протоколе BGP не предусмотрены средства для распределения нагрузки на более общем уровне.
3. Как исключение из п. 2, протокол BGP не обеспечивает оптимальной маршрутизации в структуре, которая имеет две или более глобальных сетей, соединенных друг с другом в нескольких местах. В подобных случаях администраторы должны вручную указать, какие из сетей будет анонсировать каждый из внешних маршрутизаторов.
4. Для достижения рациональной маршрутизации все автономные системы в объединенной сети должны обеспечить согласованную систему анонсирования достижимости получателей. Это означает, что протокол BGP не может гарантировать непротиворечивость маршрутной информации в глобальной сети.

### **15.19. Арбитражная система маршрутизации в объединенной сети**

Для того чтобы объединенная сеть правильно функционировала, маршрутная информация должна быть непротиворечивой по всей глобальной сети. Отдельные протоколы, такие как BGP, которые управляют процессом обмена информацией между парой маршрутизаторов, не могут гарантировать глобальной непротиворечивости. Таким образом, возникает необходимость в механизме, с помощью которого можно было бы обновлять маршрутную информацию по всей глобальной сети.

В первоначальной системе маршрутизации в Internet непротиворечивость маршрутной информации по всей глобальной сети гарантировала система базовых маршрутизаторов. Все дело в том, что в любой момент времени у нее имелись сведения только об одном маршруте к каждому получателю. Поэтому после отмены этой системы, возникла необходимость в новом механизме, позволявшем rationalизировать маршрутную информацию. Его называли *арбитражной системой маршрутизации* (*routing arbiter system*, или *RA-системой*). Основу этой системы составляла реплицируемая база данных, содержащая достоверную информацию о достижимости получателей. При этом перед обновлением информации в базе данных выполнялась ее проверка на подлинность. Такая система не позволяла любому маршрутизатору ошибочно анонсировать маршрут к некоторому получателю. Вообще говоря, анонсировать информацию о достижимости получателя имеет право только та автономная система, к которой принадлежит сеть получателя. Необходимость в подобной проверке стала очевидной еще в первоначальной системе базовой маршрутизации, в которой любой из маршрутизаторов мог анонсировать информацию о достижимости любой сети. При этом система не была застрахована от ошибок, которые возникали, когда одно из устройств случайно анонсировало неверную информацию о достижимости получателей. Ядро сети принимало эту информацию и соответствующим образом изменяло маршруты. В результате некоторые сети становились недоступными.

Чтобы понять, каким образом посторонние маршрутизаторы смогут получить доступ к базе данных арбитражной системы маршрутизации, рассмотрим существующую в настоящее время структуру сети Internet. Выше уже отмечалось, что крупные провайдеры Internet взаимодействуют между собой через специальные *точки доступа* (*Network Access Points*, или *NAP*). Таким образом, в плане маршрутизации точку доступа к сети (NAP) можно рассматривать как границу между несколькими автономными системами. При этом для обмена маршрутной информацией между каждой парой провайдеров через точку доступа можно было бы использовать протокол BGP. Однако такое решение неэффективно и может привести к возникновению противоречий. Поэтому в каждой точке доступа установлен специальный компьютер под названием *сервер маршрутизации* (*route server*, или *RS*), на котором хранится копия арбитражной базы данных и поддерживается протокол BGP. Каждый из провайдеров назначает один из своих маршрутизаторов, ближайших к точке доступа, в качестве пограничного BGP-маршрутизатора. Выбранный пограничный маршрутизатор устанавливает соединение с сервером маршрутизации и обменивается с ним маршрутной информацией по протоколу BGP. Таким образом провайдер анонсирует информацию о достижимости своих сетей, а также сетей своих клиентов и получает информацию о сетях, принадлежащих другим провайдерам.

Одно из главных преимуществ использования протокола BGP для получения доступа к серверу маршрутизации заключается в том, что с помощью этого протокола можно передавать информацию как о доступности, так и о недоступности получателей. Когда получатель становится недоступен, провайдер информирует об этом сервер маршрутизации, который, в свою очередь, сообщает об этом другим провайдерам. Распространение информации о недоступности получателя позволяет уменьшить объем ненужного трафика, поскольку дейтаграммы, предназначенные для недостижимых получателей, могут быть аннулированы еще до того, как они перейдут от одного провайдера к другому<sup>3</sup>.

## 15.20. Формат уведомляющих BGP-сообщений

Помимо сообщений об открытии и обновлении, описанных выше, в протоколе BGP предусмотрены специальные *уведомляющие сообщения* (*notification messages*). Они используются для передачи служебной информации, а также для уведомлений о возникших ошибках. Следует отметить, что ошибки в сети возникают постоянно. Поэтому, обнаружив проблему, модуль протокола BGP отсылает противоположной стороне уведомляющее сообщение и закрывает TCP-соединение. Формат уведомляющего BGP-сообщения показан на рис. 15.11.

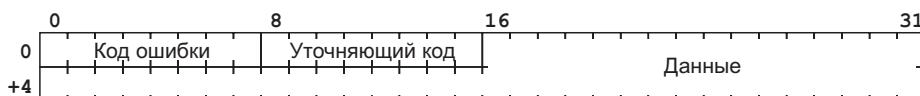


Рис. 15.11. Формат уведомляющего BGP-сообщения. Эти данные помещаются после стандартного заголовка сообщения

В первый октет уведомляющего сообщения помещается код ошибки, а во второй — код, уточняющий эту ошибку. Для каждого типа ошибки существует свой уточняющий код. Коды ошибки и соответствующие им уточняющие коды приведены в табл. 15.3.

<sup>3</sup> В системе арбитражной маршрутизации, также как и в базовой системе маршрутизации, которую она заменила, не назначаются стандартные маршруты. Поэтому иногда ее называют *зоной, свободной от стандартных маршрутов* (*default-free zone*).

**Таблица 15.3. Коды ошибки и уточняющие их коды протокола BGP**

<i>Код ошибки</i>	<i>Уточняющий код</i>	<i>Описание</i>
1	—	Ошибка в заголовке сообщения
	1	Срыв синхронизации соединения
	2	Ошибочная длина сообщения
	3	Неверный тип сообщения
2	—	Ошибка в сообщении об открытии
	1	Указанный номер версии не поддерживается
	2	Некорректный номер автономной системы противоположной стороны
	3	Неправильный идентификатор BGP-соединения
	4	Один из необязательных параметров BGP-сообщения не поддерживается
	5	Ошибка аутентификации
	6	Недопустимое время удержания
3	—	Ошибка в сообщении об обновлении
	1	Ошибка в списке атрибутов
	2	Неизвестный атрибут
	3	Атрибут пропущен
	4	Ошибка во флагах атрибута
	5	Неправильная длина атрибута
	6	Неправильный атрибут источника
	7	Маршрутная петля в автономной системе
	8	Ошибкаочный адрес ближайшей точки перехода
	9	Ошибка в необязательных атрибутах
	10	Ошибка в поле адреса сети
	11	Ошибка в пути к автономной системе
4	—	Значение таймера удержания исчерпано
5	—	Ошибка конечного автомата
6	—	Завершение (разорвать соединение)

## 15.21. Децентрализация структуры объединенной сети

Нам осталось рассмотреть два важных вопроса, относящихся к структуре объединенной сети. Первый — касается централизации: как изменить структуру объединенной сети, чтобы устраниТЬ зависимость от централизованной системы маршрутизации? Второй вопрос связан с уровнем доверительных отношений: можно ли структуру объединенной сети расширить таким образом,

чтобы обеспечить более тесные (доверительные) отношения между некоторыми автономными системами?

Устранить зависимость от централизованной системы маршрутизации и при этом сохранить доверительные отношения между автономными системами нелегко. Несмотря на то что структура семейства протоколов TCP/IP продолжает развиваться, все же во многих его протоколах можно заметить следы этой зависимости. Если бы централизации не существовало вовсе, то каждому провайдеру пришлось бы обмениваться информацией о достоверности сетей со всеми другими провайдерами, к которым он подключен. Следовательно, объем трафика маршрутизации был бы намного больше, чем при использовании арбитражной системы маршрутизации. В конечном счете централизация играет важную роль в рационализации маршрутов и обеспечении доверительных отношений. Кроме сохранности базы данных о достоверности сетей, арбитражная система маршрутизации гарантирует непротиворечивость информации в глобальной сети и является надежным источником информации.

## 15.22. Резюме

Для того, чтобы ограничить объем служебного трафика маршрутизаторы необходимо разбить на группы. Современная структура сети Internet состоит из совокупности автономных систем, связанных друг с другом. Каждая автономная система состоит из набора маршрутизаторов и нескольких сетей, управляемых из единого административного центра. Для сообщения маршрутов другим автономным системам используется протокол внешнего шлюза. Прежде чем к внутренним сетям автономной системы смогут получить доступ другие автономные системы, первая должна анонсировать во внешний мир информацию о достоверности своих сетей.

Самым распространенным из используемых протоколов внешнего шлюза является протокол пограничного шлюза BGP. В данной главе были рассмотрены три типа сообщений этого протокола, которые используются для инициирования соединения (сообщение об открытии), отправки информации о достоверности (сообщение об обновлении) и для извещения о возникших проблемах (уведомляющее сообщение). Каждое сообщение начинается со стандартного заголовка, в котором может находиться информация об аутентификации (она не является обязательной). Для отправки BGP-сообщений используется протокол TCP. Чтобы убедиться в том, что обе стороны BGP-соединения продолжают оставаться на связи, в протоколе BGP используется специальный механизм отслеживания активного состояния.

В глобальной сети Internet каждый крупный провайдер относится к отдельной автономной системе. Основная граница между автономными системами проходит в точках доступа к сети (NAP), через которые несколько провайдеров взаимодействуют между собой. В каждой точке доступа к сети установлен отдельный сервер маршрутизации. Это позволяет исключить обмен маршрутной информацией по протоколу BGP между парами провайдеров. Для обмена информацией с сервером маршрутизации каждый провайдер использует протокол BGP. Это позволяет ему анонсировать достоверность как своих сетей, так и сетей своих клиентов, а также получать информацию о достоверности сетей других провайдеров.

## Материал для дальнейшего изучения

Информацию о первых системах маршрутизации в сети Internet можно найти в [RFC 827, 888, 904 и 975]. Рехтер (Rekhter) и Ли (Li) в [RFC 1771] описывают версию 4 протокола пограничного шлюза (BGP-4). Протокол BGP основательно

пересматривался и исправлялся три раза. Ранние версии этого протокола описаны в [RFC 1163, 1267 и 1654]. Траина (Traina) в [RFC 1773] делится с читателями опытом работы с протоколом BGP-4. В [RFC 1774] тот же Траина анализирует объем возникающего трафика маршрутизации. И наконец, Вилламайзер (Villamizar) и др. в [RFC 2439] рассматривает проблему колебаний маршрута.

## Упражнения

- 15.1. Если в вашем сетевом центре используется один из протоколов внешнего шлюза, например BGP, выясните, сколько маршрутов анонсирует сеть NSFNET?
- 15.2. В некоторых реализациях протокола BGP используется механизм замораживания (“hold down”). Суть его заключается в задержке на определенный промежуток времени обработки сообщения об открытии BGP-соединения, которое поступило сразу после получения сообщения о разрыве BGP-соединения от этого же маршрутизатора. Какую проблему помогает решить механизм замораживания?
- 15.3. Какие из маршрутизаторов, изображенных на рис. 15.2, должны поддерживать протокол BGP и почему?
- 15.4. В официальное описание протокола BGP включена модель конечного автомата, объясняющая принцип работы этого протокола. Начертите диаграмму состояний конечного автомата протокола BGP и отметьте на ней переходы из состояния в состояние.
- 15.5. Что произойдет, если маршрутизатор, принадлежащий к одной из автономных систем, отправит маршрутизатору другой автономной системы BGP-сообщение об обновлении, в котором будет указано, что через первый маршрутизатор можно получить доступ к любому получателю объединенной сети?
- 15.6. Могут ли два маршрутизатора, принадлежащие к разным автономным системам, создать маршрутную петлю, обмениваясь друг с другом BGP-сообщениями об обновлении? Обоснуйте свой ответ.
- 15.7. Должен ли маршрутизатор, использующий протокол BGP для анонсирования маршрутов, по-разному интерпретировать маршруты, хранящиеся в его локальной таблице маршрутизации и полученные по протоколу BGP? В частности, должен ли вообще маршрутизатор анонсировать информацию о достижимости некоторой сети, если информация о ней отсутствует в его локальной таблице маршрутизации? Обоснуйте свой ответ. (*Подсказка.* Прочтайте RFC).
- 15.8. Возвращаясь к предыдущему упражнению, внимательно прочитайте спецификацию протокола BGP-4. Имеет ли право маршрутизатор анонсировать информацию о достижимости некоторого получателя, если этот получатель отсутствует в его локальной таблице маршрутизации?
- 15.9. Если вы являетесь сотрудником большой корпорации, выясните, на сколько автономных систем разбита ее сеть предприятия. Если автономных систем несколько, узнайте обмениваются ли они маршрутной информацией?
- 15.10. В чем заключается главное преимущество разделения сети большой, многонациональной корпорации на несколько автономных систем? В чем заключается главный недостаток?

- 15.11.** Корпорации  $A$  и  $B$  используют для обмена маршрутной информацией протокол BGP. Чтобы помешать компьютерам корпорации  $B$  получить доступ к машинам, расположенным в одной из своих сетей,  $N$ , сетевой администратор корпорации  $A$  изменяет настройки протокола BGP своего маршрутизатора и исключает сеть  $N$  из анонсов, отсылаемых корпорации  $B$ . Можно ли при этом считать, что сеть  $N$  находится в полной безопасности? Обоснуйте свой ответ.
- 15.12.** Поскольку в протоколе BGP используется надежный транспортный протокол TCP, сообщения о поддержке соединения в активном состоянии не могут затеряться. Имеет ли смысл устанавливать между отправкой тестовых пакетов интервал, равный одной третьей от значения таймера удержания?
- 15.13.** Обратитесь к RFC за подробной информацией о формате поля *параметров маршрута*. Какой минимальный размер BGP-сообщения об обновлении?



# 16

## *Маршрутизация внутри автономной системы (RIP, OSPF, HELLO)*

### **16.1. Введение**

В предыдущей главе было введено понятие автономной системы, а также был рассмотрен один из протоколов внешнего шлюза (BGP), который используется маршрутизаторами для анонсирования информации о сетях своей системы другим автономным системам. В этой главе завершается общее рассмотрение методов маршрутизации в объединенной сети. Здесь будут описаны методы, с помощью которых маршрутизатор может узнать о других сетях, принадлежащих к его автономной системе.

### **16.2. Сравнение статических и динамических методов внутренней маршрутизации**

Два маршрутизатора, находящиеся в одной автономной системе, называют *внутренними* (*interior*) по отношению друг к другу. Например, два маршрутизатора, расположенных в университетской сети, считаются внутренними по отношению друг к другу, если машины этой сети принадлежат к одной автономной системе.

Каким же образом маршрутизаторы, относящиеся к одной автономной системе, узнают о сетях, принадлежащих этой автономной системе? В небольших, редко изменяющихся объединенных сетях, администраторы могут устанавливать и модифицировать маршруты вручную. При этом администратор самостоятельно ведет таблицу сетей и обновляет данные, содержащиеся в этой таблице, каждый раз, когда к автономной системе подключается новая сеть, или удаляется старая. В качестве примера рассмотрим небольшую корпоративную объединенную сеть, топология которой изображена на рис. 16.1.

Выполнить маршрутизацию в объединенной сети, топология которой изображена на рис. 16.1, — задача весьма тривиальная, поскольку между любыми двумя узлами этой сети существует только один возможный маршрут. Учитывая это, сетевой администратор может вручную сконфигурировать систему маршрутизации во всех узлах сети и маршрутизаторах. Если в топологии объединенной сети происходят изменения (например, к ней подключается новая сеть), администратор должен будет опять же вручную переконфигурировать систему маршрутизации на всех машинах.

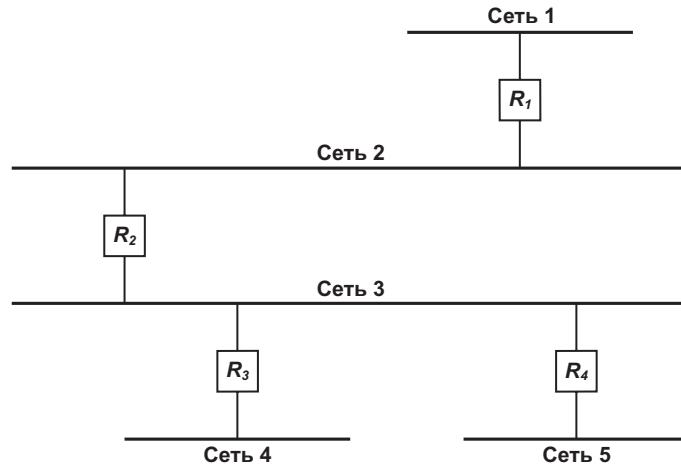


Рис. 16.1. Пример небольшой корпоративной объединенной сети, состоящей из пяти локальных сетей Ethernet и четырех маршрутизаторов. Заметьте, что между любыми двумя узлами этой объединенной сети существует только один возможный маршрут

Недостатки управления системой маршрутизации вручную очевидны: человек не может справиться с быстрым ростом или быстрыми изменениями, происходящими в объединенной сети. В больших, быстро изменяющихся средах, таких как глобальная сеть Internet, человек просто не может достаточно быстро реагировать на изменения иправляться с возникающими проблемами. Нужно как-то автоматизировать этот процесс, чтобы повысить надежность и сократить время восстановления системы после сбоя, особенно в небольших объединенных сетях, имеющих альтернативные маршруты. Чтобы понять суть процесса, рассмотрим, что произойдет, если к объединенной сети, изображенной на рис. 16.1, добавить еще один дополнительный маршрутизатор. В результате мы получим объединенную сеть, топология которой показана на рис. 16.2.

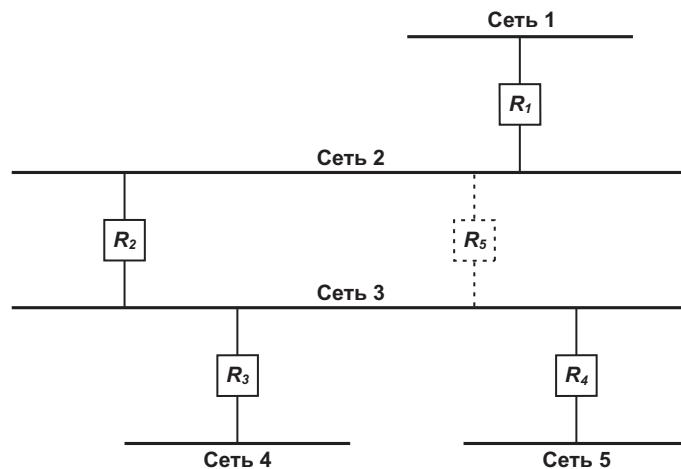


Рис. 16.2. Добавление маршрутизатора  $R_5$  создает альтернативный маршрут между сетями 2 и 3. При этом программы маршрутизации могут быстро отреагировать на сбой и автоматически переключится на альтернативный маршрут

В структурах объединенной сети, имеющих несколько физических маршрутов, администраторы, как правило, выбирают один из них в качестве основного. Если один или несколько маршрутизаторов, расположенных на основном маршруте, выйдут из строя, система должна изменить маршрутизацию и переключить трафик на альтернативный маршрут. Ручное изменение маршрутов не только занимает много времени, но и чревато ошибками. Поэтому даже в небольших объединенных сетях необходимо использовать автоматизированную систему, чтобы обеспечить быстрое и надежное изменение маршрутов.

Чтобы обеспечить автоматическое сохранение точной информации о достижимости сетей, внутренние маршрутизаторы, как правило, периодически вступают друг с другом во взаимодействие, обмениваясь либо данными о достижимости сетей, либо сетевой маршрутной информацией, из которой легко можно получить данные о достижимости. Как только будет собрана информация о достижимости сетей по всей автономной системе, один из маршрутизаторов этой системы может анонсировать ее другим автономным системам, используя для этой цели протокол внешнего шлюза.

В отличие от взаимодействия между внешними маршрутизаторами, для осуществления которого используется широко распространенный протокол BGP, ни один из существующих протоколов не рассчитан для использования в рамках автономной системы. Одна из причин такого отличия заключается в разнообразии топологий и сетевых технологий, используемых в автономных системах. Вторая причина является следствием компромисса между простотой и функциональностью системы: протоколы, которые легко установить и сконфигурировать, как правило, не выполняют сложных функций. Поэтому популярность приобрели лишь несколько протоколов. В большинстве небольших автономных систем для распространения маршрутной информации внутри системы используется только один из этих протоколов. В более крупных автономных системах часто используется несколько протоколов.

Поскольку единого стандарта не существует, для описания алгоритма, используемого внутренними маршрутизаторами для обмена информацией о достижимости сетей и маршрутной информацией, используется специальный обобщенный термин — *протокол внутреннего шлюза (Interior Gateway Protocol, или IGP)*. Например, в последнем варианте базовой системы маршрутизации в качестве протокола внутреннего шлюза использовался протокол под названием *SPREAD*. В некоторых автономных системах в качестве протокола внутреннего шлюза используется протокол BGP, хотя это редко имеет смысл для небольших автономных систем, состоящих из нескольких локальных сетей, поддерживающих широковещательный режим передачи.

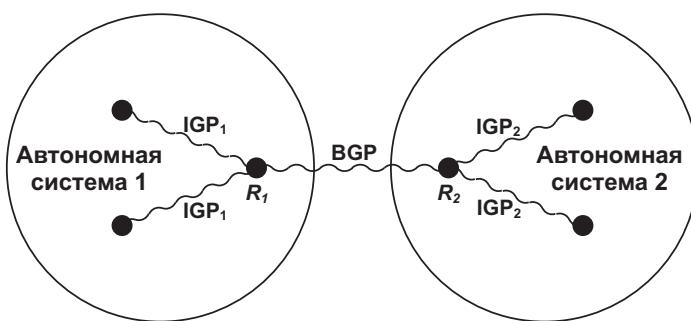


Рис. 16.3. Схематичное изображение двух автономных систем, в каждой из которых используется собственный внутренний протокол IGP. Для взаимодействия между внешними маршрутизаторами двух автономных систем используется протокол BGP

На рис. 16.3 изображены две автономные системы, в каждой из которых для распространения маршрутной информации между внутренними маршрутизаторами используется протокол IGP.

На рис. 16.3 аббревиатурой  $IGP_1$  обозначен внутренний протокол маршрутизации, используемый в пределах автономной системы 1, а  $IGP_2$  — аналогичный протокол, используемый в пределах автономной системы 2. С помощью рис. 16.3 проиллюстрирован важный принцип системы маршрутизации, который заключается в следующем.

*Один и тот же маршрутизатор может использовать одновременно два различных протокола: один — для анонсирования информации за пределы своей автономной системы, а другой — для взаимодействия с другими маршрутизаторами своей автономной системы.*

В частности, маршрутизаторы, которые поддерживают протокол BGP для анонсирования информации о достижимости сетей, как правило, вынуждены также поддерживать протокол IGP, чтобы получить информацию от источников, расположенных в пределах своих автономных систем.

## 16.3. Протокол маршрутной информации (RIP)

### 16.3.1. История возникновения протокола RIP

Одним из наиболее широко распространенных протоколов IGP является *протокол маршрутной информации (Routing Information Protocol, или RIP)*, который в системе UNIX реализован в виде программы `routed`<sup>1</sup>. Программа `routed` была создана в Калифорнийском университете в Беркли. Ее разработчики хотели сделать так, чтобы процесс маршрутизации в локальных сетях проходил согласованно, а информация о достижимости получателей была непротиворечивой. Для быстрого обмена маршрутной информацией в программе `routed` использовался широковещательный режим передачи в физической сети. Программа изначально не была предназначена для работы в больших распределенных сетях (хотя в настоящее время разработчики программного обеспечения продают адаптированные версии программ, поддерживающих протокол RIP и предназначенных специально для использования в глобальных сетях).

В основу работы программы `routed` были положены результаты исследований, проведенных в исследовательском центре Пало Альто корпорации Херох (Palo Alto Research Center, или PARC). Разработчики программы реализовали протокол, производный от протокола *NS RIP* фирмы Херох. В результате обобщения им удалось охватить несколько семейств сетей.

Несмотря на незначительные усовершенствования по сравнению со своими предшественниками, популярность RIP в качестве протокола IGP не стала следствием только лишь его технических преимуществ. Скорее всего это результат включения программы `routed` в состав популярной операционной системы 4BSD UNIX, созданной в Калифорнийском университете в Беркли. Таким образом, многие сетевые центры, в которых использовалось семейство протоколов TCP/IP, просто устанавливали программу `routed` (т.е. начинали использовать протокол RIP, даже не задумываясь над его техническими преимуществами или ограничениями). В результате была сформирована система локальной

<sup>1</sup> Это название соответствует принятому в системе UNIX соглашению по именованию фоновых процессов (демонов), согласно которому к названию программы прибавляется латинская буква “d” (от слова “daemon”). Оно произносится как “route-d”.

маршрутизации на основе протокола RIP. Поэтому исследовательским группам ничего не оставалось, как адаптировать этот протокол для использования в более крупных сетях.

Возможно, самый поразительный факт, касающийся протокола RIP, заключается в том, что он был создан и широко распространен задолго до написания его официального стандарта. Большинство реализаций протокола были построены на основе программы, разработанной в Беркли. Их способность к взаимодействию зависела от того, насколько правильно программист уловил тонкости протокола и реализовал недокументированные возможности. С появлением новой версии возникло и больше трудностей. Документ RFC со стандартом протокола RIP появился лишь в июне 1988 года. Благодаря ему разные производители смогли состыковать свои программы.

### 16.3.2. Функционирование протокола RIP

Протокол RIP по сути является прямой реализацией дистанционно-векторного принципа маршрутизации для локальных сетей. В нем участники маршрутизации подразделяются на *активных* и *пассивных* (т.е. *молчаливых*). Активные участники анонсируют свои маршруты другим участникам; пассивные — прослушивают сообщения протокола RIP и используют их для обновления своих таблиц маршрутизации, но не анонсируют маршруты. Использовать протокол RIP в активном режиме может только маршрутизатор, узел сети должен работать в пассивном режиме.

Маршрутизатор, поддерживающий протокол RIP в активном режиме, каждые 30 секунд рассыпает в широковещательном режиме сообщения об обновлении маршрутной информации. В этом сообщении содержится текущая информация, полученная из локальной таблицы маршрутизатора. Каждое сообщение об обновлении состоит из набора парных значений. В каждой из пар указывается IP-адрес сети и расстояние до этой сети, выраженное целым числом. Для измерения расстояний в протоколе RIP используется принцип подсчета *количество переходов*. Согласно принятой в протоколе RIP системе измерений, маршрутизатор находится на расстоянии одного перехода от непосредственно подключенной в нему сети<sup>2</sup>, на расстоянии двух переходов — от сети, доступ к которой можно получить через другой маршрутизатор, и т.д. Таким образом, *количество переходов* или *счетчик числа переходов* по маршруту от отправителя до получателя указывает на количество маршрутизаторов, через которые проходит дейтаграмма, следующая по этому маршруту. Само собой разумеется, что вычисление кратчайших маршрутов методом подсчета количества переходов не всегда дает оптимальный результат. Например, передача по маршруту с количеством переходов 3, который проложен через три локальные сети Ethernet, может осуществляться намного быстрее, чем по маршруту с количеством переходов 2, но проходящему через два спутниковых канала. Поэтому, чтобы компенсировать различия в технологиях передачи данных, в большинстве реализаций протокола RIP администратор может искусственно установить большое количество переходов при анонсировании маршрутов к низкоскоростным сетям. Таким образом, число переходов выступает в роли весового коэффициента маршрута. Чем он выше, тем сложнее и дороже маршрут. Поэтому весовые коэффициенты иногда называют *стоимостью маршрута* (*route cost*).

И активные, и пассивные участники протокола RIP прослушивают все широковещательные сообщения и обновляют информацию в своих таблицах в соот-

<sup>2</sup> В других протоколах маршрутизации считается, что непосредственно подключенная сеть находится на нулевом расстоянии от маршрутизатора.

вествии с описанным в предыдущей главе дистанционно-векторным алгоритмом. Например, в объединенной сети, топология которой показана на рис. 16.2, маршрутизатор  $R_1$  посыпает по сети 2 широковещательное сообщение, которое содержит пару чисел  $(1, 1)$ . Это означает, что маршрутизатор может обеспечить доступ к сети 1 за 1 переход. Маршрутизаторы  $R_2$  и  $R_5$  получают это сообщение и устанавливают маршрут к сети 1 через маршрутизатор  $R_1$  (при этом количество переходов уже равно 2). Затем маршрутизаторы  $R_2$  и  $R_5$  включают в свои сообщения протокола RIP пару чисел  $(1, 2)$  и рассыпают их по сети 3. В конечном счете все маршрутизаторы и узлы автономной системы будут иметь информацию о маршруте к сети 1.

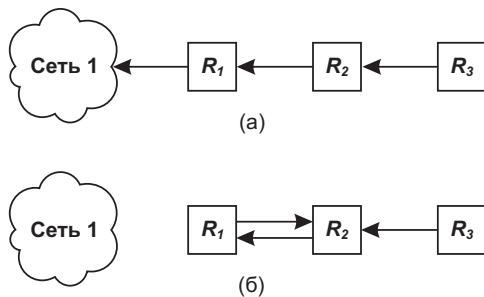
Для повышения надежности и улучшения функциональных возможностей протокола RIP в нем определено несколько правил. Например, как только маршрутизатор узнает о маршруте от другого маршрутизатора, он должен применить к новой информации принцип *гистерезиса*. Это означает, что информация о маршрутах с одинаковыми весовыми коэффициентами (или одинаковой стоимостью) не должна заменяться. В нашем примере, если оба маршрутизатора —  $R_2$  и  $R_5$  — анонсируют сеть 1 с весовыми коэффициентами 2, маршрутизаторы  $R_3$  и  $R_4$  устанавливают маршрут к этой сети через тот из них, который первым предоставит им информацию о сети 1. Таким образом можно подвести итог всему сказанному выше.

*Чтобы предотвратить колебание между маршрутами с одинаковой стоимостью, в протоколе RIP принято правило, согласно которому информацию о существующих маршрутах необходимо сохранять до тех пор, пока не появится маршрут, стоимость которого будет значительно ниже стоимости существующих маршрутов.*

Что же произойдет, если первый маршрутизатор, который должен анонсировать маршрутную информацию, не сможет этого сделать (например, если он выйдет из строя)? Согласно протоколу RIP, все слушатели информации должны отслеживать тайм-ауты для маршрутов, о которых они узнают через протокол RIP. Как только произвольный маршрутизатор заносит информацию о маршруте в свою таблицу, он запускает специальный таймер. Таймер перезапускается каждый раз, когда маршрутизатор получает следующее сообщение протокола RIP, анонсирующее этот же маршрут. Маршрут становится недействительным, если на протяжении 180 секунд не поступает повторное сообщение, анонсирующее его.

Протокол RIP должен обрабатывать три вида ошибок, вызванных использованием лежащего в его основе алгоритма. Во-первых, поскольку базовый алгоритм не может явно обнаружить маршрутных петель, в протоколе RIP нужно либо сделать допущение, что его участникам можно доверять, либо предпринять меры, предотвращающие образование таких петель. Во-вторых, чтобы система на основе протокола RIP работала стабильно, для максимально возможного расстояния нужно назначить небольшой весовой коэффициент (в протоколе RIP используется значение 16). Таким образом, объединенную сеть, в которой допустимое количество переходов достигает 16, администратор должен разбить на части или использовать другой протокол маршрутизации. В-третьих, в дистанционно-векторном алгоритме, используемом в протоколе RIP, может возникнуть проблема *медленной сходимости* (ее также называют проблемой *подсчета до бесконечности*), когда противоречия в маршрутной информации возникают из-за того, что сообщения об обновлении медленно распространяются по сети. Частично проблему медленной сходимости можно решить, ограничив значение, соответствующее бесконечности (в случае протокола RIP это 16). Однако при этом проблема полностью не исчезает.

Проблема противоречивости данных в таблице маршрутизации, характерна не только для протокола RIP. Это основная проблема, возникающая при использовании любого дистанционно-векторного протокола, в котором сообщения об обновлении содержат пары чисел, соответствующие адресу сети получателя и расстоянию до нее. Чтобы понять суть проблемы, рассмотрим набор маршрутизаторов, изображенных на рис. 16.4. На этом рисунке показаны маршруты в объединенной сети, представленной на рис. 16.2, которые ведут к сети 1.



*Рис. 16.4. Иллюстрация проблемы медленной сходимости. В случае (а) каждый из трех маршрутизаторов располагает информацией о маршруте, ведущем к сети 1. В случае (б) соединение с сетью 1 разорвалось, однако это не мешает маршрутизатору  $R_2$  анонсировать эту сеть, что приводит к образованию маршрутной петли*

Как показано на рис. 16.4(а), маршрутизатор  $R_1$  непосредственно подключен к сети 1. Следовательно, в его таблице маршрутизации есть маршрут с весовым коэффициентом 1, который будет включен в периодические широковещательные сообщения. Маршрутизатор  $R_2$  “узнает” об этом маршруте от маршрутизатора  $R_1$ , заносит информацию о нем в свою таблицу маршрутизации, и анонсирует этот маршрут с весовым коэффициентом 2. И наконец, маршрутизатор  $R_3$  узнает об этом маршруте от маршрутизатора  $R_2$  и анонсирует его с весовым коэффициентом 3.

Теперь предположим, что соединение между маршрутизатором  $R_1$  и сетью 1 разрывается. Маршрутизатор  $R_1$  сразу же обновляет свою таблицу маршрутизации, изменив в ней весовой коэффициент с 1 (непосредственное подключение) на 16 (бесконечность). Поэтому в следующем широковещательном сообщении маршрутизатора  $R_1$  маршрут к сети 1 будет иметь большую стоимость. Однако, если не принять дополнительных мер, какой-нибудь другой маршрутизатор может послать широковещательное сообщение с маршрутной информацией раньше маршрутизатора  $R_1$ . В частности, предположим, что маршрутизатор  $R_2$  анонсирует свои маршруты сразу после разрыва соединения между маршрутизатором  $R_1$  и сетью 1. Тогда маршрутизатор  $R_1$  получит сообщение от маршрутизатора  $R_2$  и будет руководствоваться обычным дистанционно-векторным алгоритмом. Поскольку маршрутизатор  $R_2$  уже анонсировал маршрут с более низкой стоимостью к сети 1, маршрутизатор  $R_1$  определяет, что теперь для достижения сети 1 требуется 3 перехода (2 перехода — для маршрутизатора  $R_2$ , чтобы достичь сети 1, и 1 переход — для достижения маршрутизатора  $R_2$ ). После этого маршрутизатор  $R_1$  заносит в свою локальную таблицу адрес маршрутизатора  $R_2$  в качестве ближайшей точки перехода для сети 1. Результат этого показан на рис. 16.4(б). Если один из маршрутизаторов  $R_1$  или  $R_2$  получит дейтаграмму, предназначенную для сети 1, возникнет маршрутная петля. Дейтаграммы будут курсировать в цикле между маршрутизаторами  $R_1$  и  $R_2$  до тех пор, пока не истечет время их жизни.

Последующие широковещательные сообщения протокола RIP, посылаемые двумя маршрутизаторами, не помогут быстро решить проблему. В следующем цикле обмена маршрутной информацией устройство  $R_1$  пересыпает содержащиеся в его таблице данные. Когда маршрутизатору  $R_2$  станет известно, что для

достижения сети 1 через маршрутизатор  $R_1$ , требуется выполнить 3 перехода, он вычислит новое расстояние до сети 1, которое станет равным 4. В третьем цикле маршрутизатор  $R_1$  получит сообщение от маршрутизатора  $R_2$ , в котором будет указано уже увеличенное расстояние до сети 1, после чего он у себя увеличит на единицу расстояние до сети 1, и оно станет равным 5. В результате два маршрутизатора в своих таблицах каждый раз будут увеличивать на единицу значение расстояния до сети 1. Циклы повторяются до тех пор, пока не будет достигнуто максимальное значение расстояния — 16, принятое в протоколе RIP, которое соответствует значению бесконечности.

### 16.3.3. Решение проблемы медленной сходимости

В приведенном на рис. 16.4 примере проблему медленной сходимости алгоритма RIP можно решить, используя для этого метод, получивший название *обновление разделенного горизонта* (*split horizon update*). При использовании этого метода маршрутизатор не должен распространять маршрутную информацию обратно по тому же интерфейсу, из которого она была получена. В рассматриваемом примере разделение горизонта препятствует маршрутизатору  $R_2$  повторно анонсировать маршрут, ведущий к сети 1, устройству  $R_1$ . Поэтому, если соединение между маршрутизатором  $R_1$  и сетью 1 разрывается, маршрутизатор  $R_2$  должен прекратить анонсирование маршрута к этой сети. Благодаря применению метода разделения горизонта в изображеной на рис. 16.4 сети никогда не возникнет маршрутных петель. Более того, после выполнения нескольких циклов обмена сообщениями об обновлении маршрутной информации все участвующие в протоколе RIP маршрутизаторы приходят к выводу, что сеть 1 больше недоступна. Как показано в одном из упражнений в конце главы, эвристический метод разделения горизонта не может устраниć образование маршрутных петель во всех возможных топологиях объединенной сети.

Другой способ решения проблемы медленной сходимости алгоритма RIP — рассмотрение ее с точки зрения информационного потока. Если маршрутизатор анонсирует короткий маршрут, ведущий к определенной сети, то все маршрутизаторы, получившие информацию об этом маршруте, быстро реагируют на нее и заносят ее в свои локальные таблицы. Если маршрутизатор прекращает анонсировать какой-то маршрут, то прежде чем маршрут будет признан недостижимым, должно пройти определенное время, зависящее от установленной величины тайм-аута. По истечении тайм-аута маршрутизатор обязан найти альтернативный маршрут и начать анонсировать о нем информацию. К сожалению, маршрутизатор не может “знать”, зависит ли новый маршрут от только что исчезнувшего маршрута. Следовательно, отрицательная информация не всегда распространяется быстро. Саму идею и объяснение этого феномена можно выразить с помощью следующей короткой эпиграммы.

*Хорошая новость распространяется быстро, а плохая — медленно.*

Другой метод, который применяют для решения проблемы медленной сходимости, заключается в использовании приема *замораживания изменений* (*hold down*), или временного отказа от приема сообщений об изменениях. Участвующий в обмене информацией маршрутизатор, получив сообщение о недоступности сети, должен какое-то время игнорировать поступающую об этой сети информацию. Как правило, период замораживания изменений устанавливается равным 60 секунд. Смысл этого метода заключается в том, что по прошествии достаточно большого периода времени можно будет гарантировать, что все машины получат информацию о недоступности сети. В результате по ошибке не будет принято устаревшее сообщение. Необходимо отметить, что все участвующие

в информационном обмене по протоколу RIP машины должны использовать идентичные методы замораживания изменений, чтобы предотвратить образование маршрутных петель. Недостаток метода замораживания изменений заключается в том, что при возникновении маршрутных петель они сохраняются на протяжении всего периода замораживания. И, что самое неприятное, на период замораживания изменений сохраняются все неверные маршруты, даже если существуют альтернативные маршруты.

Последний прием, используемый для решения проблемы медленной сходимости, называется *обратное исправление* (*poison reverse*). Когда обрывается связь с сетью, анонсирующий ее маршрутизатор сохраняет в своей таблице данные об этой сети на время посылки нескольких периодических сообщений об обновлении. При этом в широковещательных сообщениях указывается бесконечная стоимость маршрута к сети, с которой отсутствует связь. Чтобы обеспечить максимальную эффективность метода обратного исправления, его необходимо применять совместно с методом *мгновенного изменения* (*triggered updates*). Суть второго метода состоит в том, что, получив информацию о каком-либо сбое, маршрутизатор должен сразу же отослать срочное сообщение, а не ждать наступления момента отсылки следующего периодического сообщения. Таким образом, отсылая без промедления сообщение об обновлении, маршрутизатор до минимума сокращает время, на протяжении которого возможно распространение некорректной информации.

К сожалению, хотя все описанные приемы (мгновенное изменение, обратное исправление, замораживание изменений, разделение горизонта) позволяют решить целый ряд основных проблем, их применение вызывает массу побочных явлений. Например, рассмотрим, что произойдет при использовании метода мгновенных изменений в объединенной сети, к которой подключено большое количество маршрутизаторов. Всего лишь одно широковещательное сообщение может изменить таблицы всех маршрутизаторов, что вызовет новый цикл рассылки широковещательных сообщений. Если в результате второго цикла рассылки сообщений будут также внесены изменения в таблицы маршрутизаторов, это даст толчок для рассылки еще большего количества сообщений. В результате возникнет целая лавина широковещательных сообщений<sup>3</sup>.

Использование как широковещательного режима передачи сообщений об обновлении, который может спровоцировать возникновение петель маршрутизации, так и метода замораживания изменений для предотвращения медленной сходимости может значительно снизить эффективность функционирования протокола RIP в глобальной сети. Передача сообщений в широковещательном режиме всегда вызывает большой трафик в сети. Даже если при этом не возникнет лавины сообщений, сам факт периодической передачи всеми маршрутизаторами сети сообщений в широковещательном режиме означает, что вместе с увеличением количества маршрутизаторов будет происходить увеличение объема трафика в сети. Если пропускная способность канала связи ограничена, повышается вероятность возникновения маршрутных петель, что может привести к очень опасным последствиям. Пакеты, посылаемые по маршрутной петле, могут вызвать перегрузку канала связи. При этом затрудняется (или даже становится невозможным) обмен маршрутной информацией между маршрутизаторами, необходимой для разрыва петель маршрутизации. Кроме того, в глобальной сети периоды замораживания изменений делятся очень долго. В результате может истечь время на таймерах, используемых высокогорневыми протоколами, что

<sup>3</sup> Чтобы избежать возникновения коллизий в сети, в протоколе RIP предусмотрена небольшая задержка перед отправкой сообщения о мгновенном изменении. Ее значение выбирается случайным образом каждым маршрутизатором.

приведет к разрыву соединения. Несмотря на эти хорошо известные проблемы, большинством административных групп протокол RIP продолжает использоваться в глобальных сетях в качестве протокола IGP.

#### 16.3.4. Формат сообщения протокола RIP1

Сообщения протокола RIP можно разделить на две большие категории: содержащие маршрутную информацию и используемые для запроса информации. Оба типа сообщений имеют одинаковый формат и состоят из стандартного заголовка, после которого следует необязательный список пар, каждая из которых включает адрес сети и расстояние до нее. На рис. 16.5 показан формат сообщения, которое используется в протоколе RIP версии 1, называемой *RIP1*.

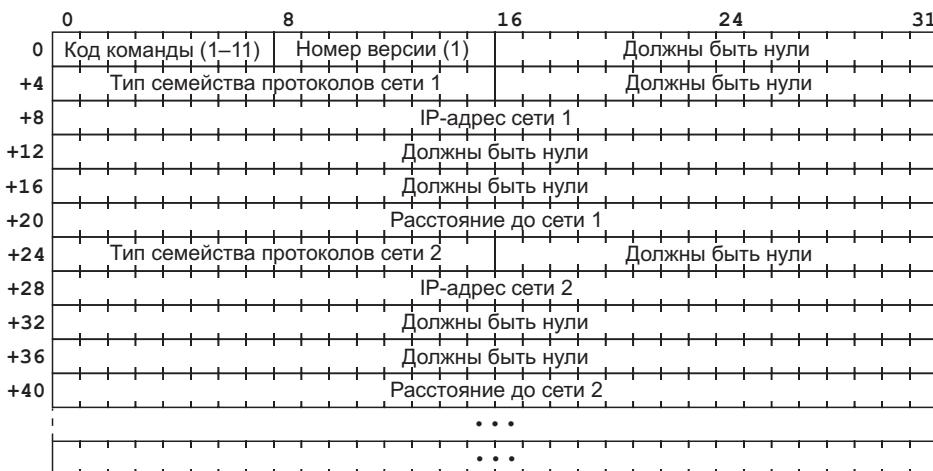


Рис. 16.5. Формат сообщения протокола RIP версии 1. За 32-битовым заголовком следует последовательность пар, каждая из которых состоит из IP-адреса сети и выраженного целым числом расстояния до этой сети

В поле *кода команды* определяется тип операции, которая будет выполнена маршрутизатором при получении этого сообщения. Коды команд приведены в табл. 16.1.

**Таблица 16.1. Коды команд сообщений протокола RIP1**

Код команды	Описание
1	Запрос на частичную или полную маршрутную информацию
2	Ответ на запрос, содержащий пары чисел ( <i>адрес сети, расстояние</i> ), взятые из таблицы маршрутизации отправителя сообщения
3	Включить режим трассировки (устаревший)
4	Отключить режим трассировки (устаревший)
5	Зарезервирован фирмой Sun Microsystems для внутреннего использования
9	Запрос на обновление (используется со схемами запроса)
10	Ответ на запрос на обновление (используется со схемами запроса)
11	Подтверждение запроса на обновление (используется со схемами запроса)

Маршрутизатор или узел сети могут получить информацию у другого маршрутизатора, послав сообщение, содержащее код команды *запроса*. В ответ на него маршрутизатор пришлет сообщение, содержащее код команды *ответа*. Однако в большинстве случаев маршрутизаторы по собственной инициативе периодически рассылают в широковещательном режиме сообщения, содержащие коды команд *ответа*. В поле *номера версии* находится номер версии используемого протокола RIP (в данном случае 1). Это поле используется получателем для подтверждения того, что он правильно будет обрабатывать сообщение.

### 16.3.5. Соглашения об адресах протокола RIP1

Об универсальности протокола RIP можно судить по способу пересылки им сетевых адресов. Принятый в протоколе RIP формат спецификации адреса может использоваться не только совместно с протоколом TCP/IP, но и со многими другими сетевыми протоколами. Как показано на рис. 16.5, указываемый в сообщении протокола RIP адрес может иметь длину до 14 октетов. Конечно, IP-адреса занимают только 4 октета. Согласно протоколу RIP, остальные октеты должны быть равны нулю<sup>4</sup>. В поле *типа семейства протоколов сети N* указывается код семейства сетевых протоколов, согласно которому необходимо интерпретировать сетевой адрес. В протоколе RIP используются значения, присвоенные семействам адресов в операционной системе 4BSD UNIX (например, семейству IP-адресов присваивается значение 2).

Кроме обычных IP-адресов, в протоколе RIP используется соглашение, согласно которому адрес 0.0.0.0 обозначает *стандартный маршрут*. В протоколе RIP к каждому анонсируемому маршруту (включая стандартные) добавляется метрика расстояния. Поэтому можно сделать так, чтобы два маршрутизатора анонсировали свои стандартные маршруты (т.е. маршруты к остальным узлам объединенной сети) с разной метрикой. При этом один из маршрутов будет использоваться в качестве основного, а другой — в качестве резервного.

В последнем поле каждого элемента сообщения протокола RIP (на рис. 16.5 оно обозначено как *Расстояние до сети N*) указывается расстояние до определенной сети, выраженное целым числом. Расстояния измеряются количеством переходов через маршрутизатор, но их значения могут находиться только в пределах от 1 до 16. Причем значение 16 используется для обозначения бесконечного расстояния до указанной сети (т.е. когда соединение с указанной сетью отсутствует).

### 16.3.6. Интерпретация и агрегация маршрутов в протоколе RIP1

Поскольку первоначально протокол RIP был разработан для работы с классовыми адресами, в версии 1 не было предусмотрено использование маски подсети. После того как в протокол IP была добавлена возможность адресации подсетей, спецификация версии 1 протокола RIP была расширена таким образом, чтобы маршрутизаторы могли обмениваться адресами подсети. Однако поскольку в сообщениях об обновлении протокола RIP1 не было предусмотрено поля, содержащего маску подсети, в спецификацию этой версии протокола внесено важное ограничение: маршрутизатор может включать в свои сообщения об обновлении маршрутной информации адреса узлов сети или подсетей при условии, что все получатели смогут интерпретировать эти адреса однозначно. В частности, маршруты к подсети могут быть включены только в те сообщения об обновлении, рассылающиеся по сети с одинаковым префиксом IP-адреса, в которой используется одна

<sup>4</sup> С целью выравнивания IP-адреса на 32-х битовую границу, разработчики поместили его в октеты 3–6 поля адреса.

маска подсети. По сути, это ограничение означает, что протокол RIP1 не может использоваться для распространения маршрутной информации о подсетях, адреса которых имеют переменную длину, т.е. в случае бесклассовых адресов. Таким образом, можно сделать следующий вывод.

*Поскольку в протоколе RIP1 не предусмотрено явное использование маски подсети, маршрутизатор может анонсировать маршруты к подсетям только при условии, что их получатели смогут однозначно интерпретировать адреса в соответствии с доступной им локальной маской подсети. Поэтому протокол RIP1 может использоваться только с классовыми адресами или адресами подсетей фиксированной длины.*

Что же произойдет, если маршрутизатор, поддерживающий протокол RIP1, подключить к одной или нескольким сетям, имеющим один и тот же префикс IP-адреса  $N$ , а также к одной или нескольким сетям, имеющим другой, отличный от  $N$ , префикс адреса? Маршрутизатор должен подготовить разные сообщения об обновлении для двух типов интерфейса. В сообщения, которые рассылаются по интерфейсам, подключенными к подсетям с префиксом адреса  $N$ , может быть включена маршрутная информация об этих подсетях. Однако в сообщения, предназначенные для других интерфейсов, эти маршруты не могут быть включены. Следовательно, рассылая свои сообщения через другие интерфейсы, маршрутизатор должен собрать всю информацию о всех подсетях сети  $N$  и анонсировать только один маршрут к сети  $N$  (т.е. выполнить агрегацию маршрутов к подсетям сети  $N$ ).

### 16.3.7. Расширения протокола RIP2

Описанное выше ограничение на интерпретацию адреса означает, что версия 1 протокола RIP не может использоваться для распространения маршрутной информации о подсетях переменной длины, а также содержащей бесклассовые адреса CIDR. После принятия версии 2 протокола RIP (*RIP2*) появилась возможность вместе с каждым адресом сети явно указывать маску подсети. Кроме того, в сообщения об обновлении протокола RIP2 явно включается информация об адресе ближайшей точки перехода. Это позволяет предотвратить образование маршрутных петель и возникновение проблемы медленной сходимости. Благодаря этому протокол RIP2 обладает расширенными функциональными возможностями, а также менее восприимчив к ошибкам.

### 16.3.8. Формат сообщения протокола RIP2

Формат сообщений, используемых в протоколе RIP2, немного расширен по сравнению с форматом аналогичных сообщений протокола RIP1. При этом дополнительная информация помещается в неиспользованные октеты поля адреса. В частности, в спецификации каждого адреса явно указывается адрес ближайшей точки перехода, а также маска подсети (рис. 16.6).

В каждый из элементов сообщения протокола RIP2 также добавляется 16-битовое поле *признака маршрута до сети N*. При передаче маршрутной информации маршрутизатор должен отослать в сообщении такое же значение признака, какое он получил. Таким образом, с помощью признака можно распространять дополнительную информацию, например, об источнике маршрута. В частности, если маршрутизатор, поддерживающий протокол RIP2, получил информацию о маршруте от устройства, принадлежащего к другой автономной системе, он может использовать значение поля *признака маршрута*, чтобы распространить номер этой автономной системы.

	0	8	16	24	31
0	Код команды (1–11)	Номер версии (2)		Должны быть нули	
+4	Тип семейства протоколов сети 1			Признак маршрута до сети 1	
+8		IP-адрес сети 1			
+12		Маска подсети для сети 1			
+16		Адрес ближайшей точки перехода для сети 1			
+20		Расстояние до сети 1			
+24	Тип семейства протоколов сети 2		Признак маршрута до сети 2		
+28		IP-адрес сети 2			
+32		Маска подсети для сети 2			
+36		Адрес ближайшей точки перехода для сети 2			
+40		Расстояние до сети 2			
		• • •			
		• • •			

Рис. 16.6. Формат сообщения протокола RIP2. В каждый из элементов сообщения, кроме IP-адреса сети и выраженного целым числом расстояния до этой сети, входит маска подсети и адрес ближайшей точки перехода

Поскольку поле номера версии в сообщении протокола RIP2 занимает тот же октет, что и в протоколе RIP1, обе версии протокола можно одновременно использовать на одном маршрутизаторе. Прежде чем обработать входящее сообщение, программа протокола RIP анализирует номер его версии.

### 16.3.9. Передача сообщений протокола RIP

В сообщении протокола RIP не предусмотрено специальное поле для указания его длины или количества элементов. Поэтому в протоколе RIP предполагается, что длину входящего сообщения должен сообщить получателю базовый механизм доставки. В частности, при использовании семейства протоколов TCP/IP сообщения протокола RIP передаются с помощью протокола UDP, который информирует получателя о длине сообщения. Протокол RIP использует порт протокола UDP под номером 520. Хотя запрос протокола RIP может отправляться из портов UDP с другими номерами, подобные сообщения всегда направляются на порт 520 протокола UDP машины получателя. При этом в широковещательных сообщениях протокола RIP в качестве порта отправителя также указывается порт 520.

### 16.3.10. Недостатки метода подсчета количества переходов в протоколе RIP

Использование RIP в качестве внутреннего протокола маршрутизации является сдерживающим фактором при выполнении маршрутизации по двум причинам. Во-первых, в протоколе RIP маршрутизация основана на подсчете количества переходов. Во-вторых, поскольку для обозначения бесконечного количества переходов (или разрыва соединения) используется небольшое целое число, размер объединенной сети при использовании протокола RIP ограничен. В частности, в протоколе RIP ограничивается *протяженность (span)* объединенной сети (т.е. максимальное расстояние от одного конца сети до другого, измеряемое в количестве сетей) числом 16. Это означает, что при использовании протокола RIP в объединенной сети между любыми двумя ее узлами может находиться не более 15 маршрутизаторов.

Обратите внимание, что в ограничении, наложенном на протяженность сети, не регламентируется ни общее количество маршрутизаторов, ни плотность их расположения. На самом деле большинство университетских сетей имеют небольшую протяженность, несмотря на то, что общее количество маршрутизаторов в них довольно велико. Все дело в том, что топологически такие сети строятся в виде одной большой иерархической структуры. В качестве примера рассмотрим типичную корпоративную объединенную сеть. В большинстве таких сетей используется иерархическая топология, т.е. высокоскоростную магистральную сеть формируют несколько маршрутизаторов. К каждому из этих маршрутизаторов подключена определенная рабочая группа, в состав которой, как правило, входит одна локальная сеть. Хотя в корпорацию могут входить десятки рабочих групп, протяженность подобной иерархической объединенной сети равна всего лишь 2. Даже если расширить локальную сеть каждой рабочей группы и подключить к ней по одному маршрутизатору, что даст возможность подключить одну или несколько дополнительных локальных сетей, протяженность корпоративной сети увеличится только до 4. Точно так же расширение иерархии еще на один уровень увеличивает протяженность объединенной сети только до 6. Таким образом, налагаемое протоколом RIP ограничение влияет только на большие автономные системы или автономные системы, в которых отсутствует иерархическая структура.

Но даже в самом идеальном случае по количеству переходов можно лишь приблизительно оценить пропускную способность сети или скорость ее передачи данных. Таким образом, на основе подсчета количества переходов нельзя проложить маршрут, обеспечивающий минимальную задержку при доставке сообщений или максимальную пропускную способность. Кроме того, вычисление маршрутов на основании минимального количества переходов имеет большой недостаток: это делает процесс маршрутизации относительно статичным. В результате маршрутизатор не сможет достаточно быстро отреагировать на изменение нагрузки в сети и изменить соответствующим образом маршруты пакетов. В следующих разделах рассматриваются альтернативные методы оценки стоимости маршрутов, а также объясняется, почему метод подсчета количества переходов продолжает пользоваться популярностью, несмотря на все его недостатки.

## 16.4. Протокол HELLO

Протокол HELLO представляет собой разновидностью протокола IGP, в котором стоимость маршрутов оценивается иначе, чем просто подсчетом количества переходов. Хотя протокол HELLO уже устарел, он сыграл важную роль в истории Internet, так как представлял собой протокол IGP, который использовался между fuzz-ball-маршрутизаторами<sup>5</sup> первой магистральной сети NSFNET. Протокол HELLO представляет собой значительный интерес, поскольку является примером протокола, использующего методику оценки стоимости маршрутов на основе величины задержки распространения пакета.

Протокол HELLO выполняет две функции. Во-первых, он синхронизирует показания часов между несколькими машинами. Во-вторых, он позволяет каждой машине определить маршруты, имеющие кратчайшее время задержки. Таким образом, в сообщениях протокола HELLO передаются временная метка и маршрутная информация. Основной принцип функционирования протокола HELLO довольно прост: каждая машина, принимающая участие в обмене сообщениями

<sup>5</sup> Термином fuzz-ball-маршрутизатор обозначается некоммерческий маршрутизатор, созданный на основе компьютера PDP11, на котором запущена специализированная программа поддержки протокола маршрутизации.

по протоколу HELLO, ведет собственную таблицу наилучших оценок значений задержек распространения пакетов до соседних машин. До передачи информационного пакета в него помещается временная метка, которая соответствует текущему показанию часов компьютера. После доставки пакета получатель оценивает текущее время задержки передачи пакета по данному каналу связи. Он вычисляет время отправки пакета, полученное из входящего пакета, из текущего значения времени, установленного на локальной машине. Периодически машины проводят упорядоченный опрос соседних машин, чтобы заново вычислить значения задержек.

Сообщения протокола HELLO также позволяют участвующим в обмене информацией машинам вычислять новые маршруты. Этот протокол использует модифицированную дистанционно-векторную систему, которая вместо подсчета числа переходов применяет методику вычисления задержки. Таким образом, каждая машина периодически отсылает соседним машинам таблицу получателей, доступ к которым она может получить, а также предполагаемую задержку для каждого из них. При поступлении сообщения от машины  $X$  получатель внимательно просматривает каждый его компонент и изменяет адрес ближайшей точки перехода к этой машине, если проходящий через нее маршрут имеет меньшую стоимость по сравнению с текущим маршрутом. Считается, что маршрут имеет меньшую стоимость, если суммарная задержка прохождения сообщения до машины  $X$ , а затем от машины  $X$  до конечного получателя будет меньше, чем задержка прохождения пакета к данному получателю по текущему маршруту.

## 16.5. Оценка задержек и стабильность работы программ маршрутизации

На первый взгляд может показаться, что использование метода оценки задержек при выборе маршрутов приводит к лучшим результатам, чем использование простого метода подсчета количества переходов. К тому же протокол HELLO хорошо зарекомендовал себя еще в самой первой магистрали Internet. Однако существует причина, из-за которой метод оценки задержек не используется в большинстве протоколов. Эта причина — нестабильность получаемых результатов.

Нестабильную работу системы может вызвать любой протокол, быстро изменяющий маршруты под воздействием внешних факторов, даже если два маршрута имеют одинаковые характеристики. Нестабильность возникает из-за того, что величина задержки, в отличие от числа переходов, не является фиксированной величиной. Незначительный разброс при измерении времени задержки происходит в результате отклонений в показаниях внутренних часов компьютера, изменения нагрузки на центральный процессор во время измерений или задержкой передачи битов данных, вызванной синхронизацией на уровне канала связи. Таким образом, если протокол маршрутизации будет быстро реагировать на малейшие изменения в задержке, может возникнуть эффект двухступенчатых колебаний, когда трафик все время переключается между двумя альтернативными маршрутами. Объясняется все очень просто. Сначала маршрутизатор определяет, что задержка доставки сообщения по маршруту 1 незначительно меньше, чем по маршруту 2, и сразу же переключает трафик на этот маршрут. Затем маршрутизатор обнаруживает, что на маршруте 2 задержка уменьшилась, и снова переключает трафик на исходный маршрут.

Чтобы избежать колебаний маршрутов, в программах протоколов, где решение принимается на основе времени задержки, применяют несколько эври-

стических алгоритмов. Во-первых, для предотвращения быстрого изменения маршрутов применяется метод *замораживания изменений*, о котором уже говорилось выше. Во-вторых, в программе протокола *не* применяется метод точного измерения значений задержек и их непосредственного сравнения. Вместо этого полученные значения округляются до большего кратного числа либо вычисляется разность двух чисел, которая затем сравнивается с заранее заданным *пороговым значением*. В-третьих, в программах протоколов *не* сравнивается каждое новое значение задержки. Вместо этого определяется скользящее *среднее* последних нескольких значений либо применяется альтернативный алгоритм “*K из N*”. Согласно этому алгоритму, прежде чем маршрут будет изменен, не менее *K* из последних *N* измерений задержки должны быть меньше текущего значения задержки.

Однако даже при использовании эвристических алгоритмов стабильность работы программы протокола маршрутизации может быть нарушена при оценке величин задержки для маршрутов, которые имеют разные характеристики. Чтобы понять, почему так происходит, необходимо иметь в виду, что на значение величины задержки может существенно повлиять интенсивность трафика в сети. При отсутствии трафика задержка в сети будет равна времени передачи битового потока из одной точки сети в другую. Но с увеличением интенсивности трафика, задержки в сети начинают возрастать. Вспомним, что маршрутизаторы помещают прибывающие пакеты в очередь для отправки, которая при увеличении трафика разрастается. А если интенсивность трафика хотя бы немного превышает предельную пропускную способность канала связи, размер очереди может увеличиваться до бесконечности. Это означает, что до бесконечности будет увеличиваться величина текущей задержки. Подведем итог.

*Величина текущей задержки в сети зависит от интенсивности трафика. Если она достигает предельной пропускной способности сети, происходит быстрый рост задержки.*

Поскольку величина задержки сильно зависит от интенсивности трафика в сети, нестабильность работы программ маршрутизации, в которых используется метод оценки значений задержки, может быть вызвана переходом программы в цикл положительной обратной связи. Образование цикла связано с малейшими внешними изменениями в загрузке сети (например, один из компьютеров создал выброс дополнительного трафика). Увеличение трафика приводит к повышению задержки, что побуждает программу протокола изменить маршрут. Однако, поскольку изменение маршрута оказывает влияние на загрузку сети, оно может привести к еще большему увеличению времени задержки. А это означает, что программа протокола выполнит очередное вычисление маршрутов. Поэтому программы протоколов, в которых используется метод оценки значений задержки, должны содержать механизмы уменьшения колебаний.

Мы уже рассматривали эвристические методы, которые могут предотвратить возникновение колебаний в самых простых случаях, когда маршруты имеют одинаковую пропускную способность и трафик в сети не превышает предельного значения. Однако применение эвристических методов может оказаться неэффективным, если альтернативные маршруты имеют разную задержку и пропускную способность. В качестве примера рассмотрим задержку, возникающую на двух маршрутах, один из которых проходит через спутниковый канал, а другой — через линию последовательной передачи данных с низкой пропускной способностью (например, 33600 бит/с). На первом этапе работы протокола, когда оба маршрута свободны, задержка в линии последовательной передачи окажется значительно ниже, чем в спутниковом канале связи. Поэтому в качестве маршрута для трафика будет выбрана линия последовательной

передачи. Поскольку линия имеет низкую пропускную способность, в ней быстро наступит перегрузка, а значит, резко возрастет и значение задержки. На втором этапе задержка в линии последовательной передачи будет значительно превышать задержку в спутниковом канале, поэтому программа маршрутизации переключит трафик на спутниковый канал. Трафик, который вызвал перегрузку в линии последовательной передачи, не создаст значительной нагрузки на спутниковый канал связи, поскольку он имеет большую пропускную способность. А это означает, что при росте трафика задержка в спутниковом канале связи не будет изменяться. В следующем цикле задержка в разгруженной линии последовательной передачи снова окажется намного меньше, чем задержка в спутниковом канале. Поэтому программа маршрутизации снова переключит трафик на линию последовательной передачи данных, и описанный выше процесс повторится сначала. Такие колебания часто возникают на практике. Как видно из рассмотренного примера, ими довольно сложно управлять, поскольку трафик, оказывающий незначительное влияние на один канал связи, может вызвать перегрузку в другом канале.

## 16.6. Совместное использование протоколов RIP, HELLO и BGP

Как уже было сказано, на маршрутизаторе одновременно может работать протокол как внутреннего, так и внешнего шлюза. Причем первый из них используется для сбора информации в пределах своей автономной системы, а второй — для анонсирования маршрутов другой автономной системе. Казалось бы, легко создать один программный продукт, который объединил бы в себе оба протокола, чтобы без человеческого вмешательства собиралась и анонсировалась маршрутная информация. Однако на практике осуществление этой идеи усложняется различными техническими и политическими препятствиями.

С формальной точки зрения протоколы внутреннего шлюза, такие как RIP и HELLO, являются протоколами маршрутизации. Маршрутизатор использует такие протоколы для обновления своей таблицы маршрутизации, построенной на основе информации, которая получена от других маршрутизаторов в пределах автономной системы. Таким образом, программа UNIX под названием `routed`, реализующая протокол RIP, анонсирует информацию из локальной таблицы маршрутизации и вносит в нее изменения при получении сообщений об обновлении. В протоколе RIP принято, что маршрутизаторы, находящиеся в пределах одной автономной системы, должны доверять информации, полученной друг от друга и предоставлять правильную маршрутную информацию.

В протоколах внешнего шлюза, таких как BGP, напротив, считается, что маршрутизатор не может доверять информации, полученной от других автономных систем. Поэтому протоколы внешнего шлюза не должны анонсировать все возможные маршруты, информация о которых имеется в локальной таблице маршрутизации. Такие протоколы ведут базу данных о достижимости сетей, а при пересылке или получении информации на нее налагаются установленные административные ограничения. Игнорирование подобных административных ограничений может сильно повлиять на процесс маршрутизации, поскольку некоторые участки объединенной сети могут стать недостижимыми. Например, если маршрутизатор некоторой автономной системы, поддерживающий протокол RIP, сообщит маршрут с очень низкой стоимостью до сети университета Пердью (а на самом деле у него не будет информации о подобном маршруте), другие маршрутизаторы, поддерживающие протокол RIP, примут этот маршрут и занесут его в свои таблицы маршрутизации. После этого весь трафик, следующий в указанную университетскую сеть, будет направлен маршрутизатору,

предоставившему неверные данные. В результате компьютеры этой автономной системы скорее всего не получат доступ к университетской сети. Проблема усугубляется, если в программах протокола внешнего шлюза не реализована поддержка административных ограничений. Например, если пограничный маршрутизатор в автономной системе использует протокол BGP для объявления недопустимого маршрута другим автономным системам, сеть университета Пердью может стать недостижимой из некоторых частей объединенной сети.

## 16.7. Маршрутизация между автономными системами

Выше уже шла речь о том, что протоколы внешнего шлюза, такие как BGP, позволяют одной автономной системе анонсировать информацию о доступности сетей другой автономной системе. Однако было бы также полезно обеспечить *маршрутизацию между автономными системами*, т.е. чтобы маршрутизаторы сами могли бы выбирать маршруты с наименьшей стоимостью. Для реализации этой идеи необходимо ввести дополнительные меры безопасности и повысить надежность доверительных отношений между устройствами объединенной сети. Расширить доверительные отношения на несколько автономных систем сложно. Самый простой подход — объединить автономные системы в группы по иерархическому принципу. Представьте себе, например, что три автономные системы, относящиеся к трем разным факультетам, подключены к большой университетской сети. Вполне естественно, что их можно объединить в одно целое, поскольку между этими автономными системами существуют общие административные связи. Причина объединения сетей в иерархические группы заключается в основном в установке разной степени доверительных отношений между маршрутизаторами. Маршрутизаторы в пределах группы больше доверяют друг другу, чем маршрутизаторы, принадлежащие к разным группам.

Объединение автономных систем в группы требует расширения спецификации протоколов маршрутизации. Речь идет о том, что, если сети относятся к разным группам, то значение расстояния до этих сетей необходимо увеличить. Такой прием назвали *трансформацией метрики*, а значения расстояния разделили на три категории. Предположим, что в маршрутизаторах, относящихся к одной автономной системе, используются значения расстояния, меньше 128. Тогда можно установить следующее правило: при передаче информации о расстоянии через границу автономных систем, относящихся к одной группе, значения расстояния необходимо трансформировать и переместить в диапазон от 128 до 191. А при передаче информации о расстоянии через границу двух групп, значения необходимо трансформировать и переместить в диапазон от 192 до 254<sup>6</sup>. Результат таких трансформаций очевиден: для произвольной сети получателя, любой маршрут, полностью проходящий в пределах одной автономной системы, обязательно будет иметь меньшую стоимость, чем маршруты, пересекающие границы между автономными системами. Более того, стоимость маршрута также будет меньше, если он пересекает несколько автономных систем, принадлежащих к одной группе, а не к разным. Основное преимущество метода трансформации метрики заключается в том, что благодаря ему каждая автономная система может самостоятельно выбирать протокол внутреннего шлюза IGP. При этом появляется возможность для других автономных систем сравнивать стоимости маршрутов.

<sup>6</sup> Для описания групп автономных систем используется специальный термин *автономная конфедерация* (*autonomous confederation*). Границам автономной конфедерации соответствуют значения расстояния большие чем 191 (после преобразования).

## 16.8. Программа *gated* и обмен информацией между автономными системами

Для обеспечения взаимодействия между автономными системами была создана специальная программа, которую называли *gated*<sup>7</sup>. Она поддерживает несколько протоколов маршрутизации, относящихся как к внутреннему, так и внешнему шлюзам (в частности, BGP), а также обеспечивает соблюдение административных ограничений. Например, программа *gated* может принять сообщения протокола RIP и изменить локальную таблицу маршрутизации компьютера, точно так же, как это делает программа *routed*. Она также может анонсировать маршруты, полученные от источников, принадлежащих к ее автономной системе, используя для этого протокол BGP. В программе *gated* разработчики предусмотрели набор правил, задавая которые системный администратор может точно определить, какие сети должна анонсировать программа *gated* и каким образом она должна сообщать расстояния к этим сетям. Поэтому, хотя программа *gated* не относится к программам поддержки протокола внутреннего шлюза, она играет важную роль в процессе маршрутизации. На ее примере показано, что можно создать механизм автоматического взаимодействия между протоколами внутреннего и внешнего шлюза, и обеспечить при этом приемлемую систему защиты.

Поскольку в программе *gated* поддерживается метод трансформации метрики, она также выполняет еще одну важную функцию. Речь идет об использовании этой программы для взаимодействия между двумя автономными системами, а также между двумя группами маршрутизаторов, в каждой из которых поддерживается протокол внутреннего шлюза.

## 16.9. Открытый протокол SPF (OSPF)

В главе 14, “Основы маршрутизации”, уже упоминалось, что алгоритмы маршрутизации, отслеживающие состояние соединения, которые используются в протоколе SPF для поиска кратчайшего пути, лучше приспособлены к расширению топологии сети, чем дистанционно-векторные алгоритмы маршрутизации. Поэтому, чтобы внедрить технологию маршрутизации с отслеживанием состояния соединения, инженерная группа IETF создала спецификацию протокола внутреннего шлюза, использующего эту технологию. Этот протокол, названный *открытым протоколом SPF (OSPF)*, позволяет решать следующие задачи.

- Как предполагает название протокола, его спецификацию можно найти в открытой печати. Создание открытого стандарта предполагает, что любой пользователь может создать его программную реализацию, не платя при этом за лицензию. Это побудило большинство поставщиков программного обеспечения поддерживать протокол OSPF, который стал широко использоваться вместо их собственных нестандартных протоколов.
- Протокол OSPF поддерживает маршрутизацию с учетом *типа обслуживания* (*type of service routing*). Это позволяет администраторам устанавливать несколько маршрутов с разными приоритетами или типом обслуживания, ведущих к одному получателю. При выборе маршрута дейтаграмм маршрутизатор, поддерживающий протокол OSPF, использует адрес получателя, а также значение поля типа обслуживания, взятое из заголовка

---

<sup>7</sup> Название *gated* произносится как “гейтди”, от сочетания “gate daemon”.

IP-пакета. Протокол OSPF — один из первых протоколов семейства TCP/IP, поддерживающий маршрутизацию с учетом типа обслуживания.

- Протокол OSPF обеспечивает *распределение нагрузки* (*load balancing*). Если администратор определяет несколько маршрутов с одинаковой стоимостью, ведущих к одному получателю, программа протокола OSPF поровну распределяет трафик между всеми маршрутизаторами. Опять же протокол OSPF был одним из первых открытых протоколов IGP, в которых предлагалась методика распределения нагрузки. Напомним, что во многих протоколах внутреннего шлюза, таких как RIP, считается, что к каждому получателю существует только один маршрут.
- Для облегчения управления сетями и обеспечения возможности их расширения протокол OSPF позволяет администрации сетевого центра разбить свою сеть и маршрутизаторы на подмножества, называемые *зонами* (*areas*). Все зоны изолированы друг от друга. Это означает, что сведения о топологической схеме зоны можно скрыть от других зон. В результате несколько групп, находящихся в пределах одного сетевого центра, могут использовать протокол OSPF для маршрутизации пакетов между собой. При этом каждая из групп может независимо от других групп изменять топологию своей внутренней сети.
- Согласно протоколу OSPF весь обмен информацией между маршрутизаторами может *аутентифицироваться*. Протокол OSPF поддерживает несколько схем аутентификации, причем для двух зон можно даже выбрать разные схемы аутентификации. Процесс аутентификации введен для того, чтобы гарантировать участие в распространении маршрутной информации только надежных маршрутизаторов. Чтобы понять суть проблемы, рассмотрим, что может произойти при использовании протокола RIP1, в котором не предусмотрена аутентификация. Не исключена ситуация, когда какой-нибудь злоумышленник с помощью своего персонального компьютера разошлет RIP-сообщения, в которых будут анонсироваться маршруты с низкой стоимостью. Тогда маршрутизаторы и узлы сети, поддерживающие протокол RIP, изменят свои локальные таблицы маршрутизации и начнут посыпать дейтаграммы персональному компьютеру злоумышленника.
- Протокол OSPF поддерживает как классовую, так и бесклассовую маршрутизацию, а также маршрутизацию для отдельного узла сети и для подсети. Все перечисленные типы маршрутизации могут понадобиться в большой объединенной сети.
- Для поддержки сетей с множественным доступом, таких как Ethernet, в стандарте протокола OSPF расширена спецификация протокола SPF, который был описан в главе 14, “Основы маршрутизации”. Там же был рассмотрен алгоритм поиска кратчайшего пути, использующий направленный граф, и отмечалось, что маршрутизатор, поддерживающий протокол SPF, должен периодически рассыпать широковещательные сообщения, содержащие информацию о состоянии соединения с каждым достижимым соседним маршрутизатором. Таким образом, если  $K$  маршрутизаторов подключены к локальной сети Ethernet, они будут отсылать  $K^2$  широковещательных сообщений о достижимости. Для уменьшения количества широковещательных сообщений в протоколе OSPF принята более сложная топологическая схема графа, в которой каждый узел представляет собой либо маршрутизатор, либо сеть. Поэтому в протоколе OSPF для каждой сети с множественным доступом назначается *выделенный шлюз* (*designated gateway*) (или *выделенный маршрутизатор*), который должен отсылать сообщения о состоянии канала связи от имени всех подключенных к сети

маршрутизаторов. Такие сообщения информируют подключенные к сети маршрутизаторы о состоянии всех внешних соединений. Если позволяет сетевое оборудование, для рассылки широковещательных сообщений протокола OSPF о состоянии канала связи используются возможности аппаратного обеспечения.

- Чтобы обеспечить максимальную гибкость, протокол OSPF позволяет администраторам описать топологию виртуальной сети, которая абстрагируется от деталей физических соединений. Например, администратор может сконфигурировать виртуальное соединение между двумя маршрутизаторами в графе маршрутизации, даже если между ними нет прямого физического соединения (т.е. связь между этими маршрутизаторами осуществляется через третью сеть).
- Протокол OSPF позволяет маршрутизаторам обмениваться маршрутной информацией, полученной от других (внешних) сетевых центров. Если один или несколько маршрутизаторов имеют соединения с другими сетевыми центрами и получают от них маршрутную информацию, они могут включить полученные данные в свои сообщения об обновлении. В формате сообщений предусмотрено, что информация, полученная из внешних источников помещается отдельно от информации, полученной от маршрутизаторов своего сетевого центра. Таким образом снимается неопределенность, связанная с источником информации и достоверностью маршрутов.

#### 16.9.1. Формат OSPF-сообщения

В начале каждого OSPF-сообщения располагается заголовок фиксированного формата, размер которого составляет 24 октета (рис. 16.7).

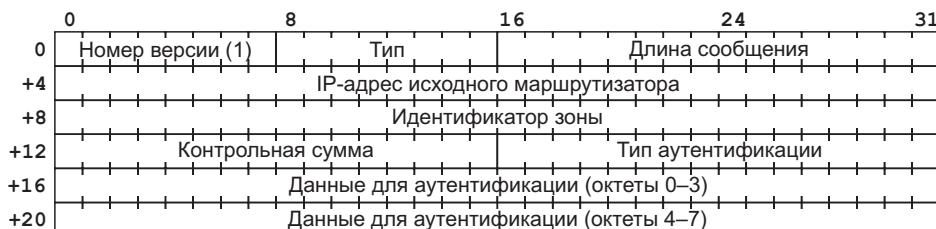


Рис. 16.7. Формат заголовка фиксированного формата OSPF-сообщения

В поле *номера версии* определяется версия используемого протокола. В поле *типа* указывается тип сообщения (табл. 16.2).

Таблица 16.2. Типы OSPF-сообщений

Тип сообщения	Описание
1	Сообщение HELLO (используется для проверки достижимости)
2	Описание базы данных (топология)
3	Запрос о состоянии соединения
4	Обновление состояния соединения
5	Уведомление о состоянии соединения связи

В поле под названием *IP-адрес исходного маршрутизатора* указывается адрес сообщения отправителя, а в поле под названием *Идентификатор зоны* — 32-битовый идентификационный номер зоны, к которой относится маршрутизатор.

Поскольку в каждое сообщение может быть включена информация об аутентификации, в поле под названием *Тип аутентификации* указывается тип используемой схемы аутентификации (пока используется два значения: 0 обозначает отсутствие аутентификации, а 1 означает, что для аутентификации используется открытый текстовый пароль).

### 16.9.2. Формат сообщения HELLO протокола OSPF

В протоколе OSPF сообщения HELLO периодически посылаются по каждому соединению для того, чтобы определить, достижима ли соседняя машина. Формат этого сообщения показан на рис. 16.8.

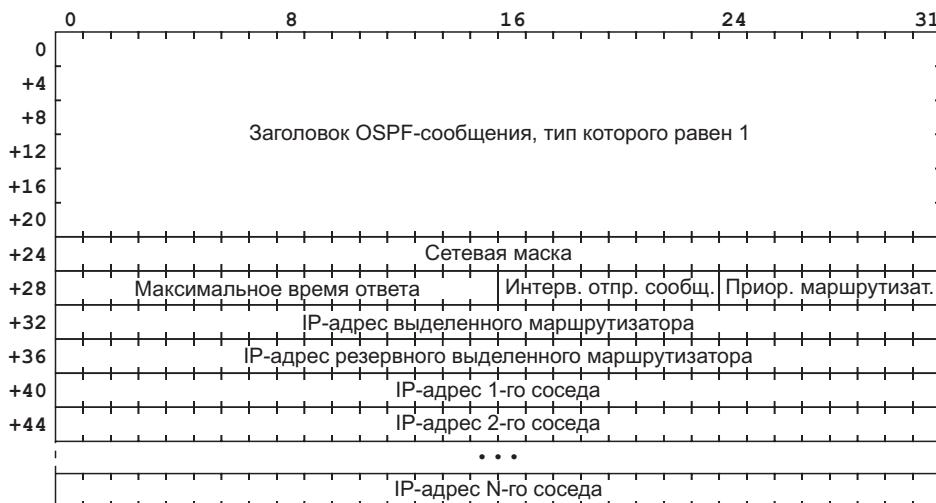


Рис. 16.8. Формат сообщения HELLO протокола OSPF. Пара расположенных по соседству маршрутизаторов обменивается этими сообщениями с целью проверки достижимости друг друга

В поле *сетевой маски* указывается маска для той сети, по которой было отослано сообщение (подробнее о сетевых масках говорилось в главе 10, “Бесклассовая адресация и подсети (CIDR)”). В поле *максимального времени ответа* указывается время в секундах, по истечении которого не отвечающий на запросы маршрутизатор считается вышедшим из строя. В поле *Интерв. отпр. сообщ.* указывается интервал времени в секундах, через который обычно отправляются сообщения HELLO. В поле *Приор. маршрутизат.* указывается целочисленное значение приоритета маршрутизатора, отправившего сообщение; оно используется для выбора резервного выделенного маршрутизатора. В полях под названием *IP-адрес выделенного маршрутизатора* и *IP-адрес резервного выделенного маршрутизатора* указываются IP-адреса одноименных маршрутизаторов для сети, по которой отсылается сообщение, с точки зрения отправителя. И, наконец, в полях под названием *IP-адрес N-го соседа* указываются IP-адреса всех соседних маршрутизаторов, от которых за последнее время отправитель получил сообщения HELLO.

### 16.9.3. Формат OSPF-сообщения описания базы данных

Маршрутизаторы обмениваются OSPF-сообщениями описания базы данных для инициализации своей базы данных, содержащей информацию о топологии сети. В обмене сообщениями один из маршрутизаторов выполняет роль главного устройства, а другой — подчиненного. Подчиненный маршрутизатор после получения каждого сообщения описания базы данных посыпает главному маршрутизатору сигнал подтверждения приема. Формат сообщения описания базы данных показан на рис. 16.9.

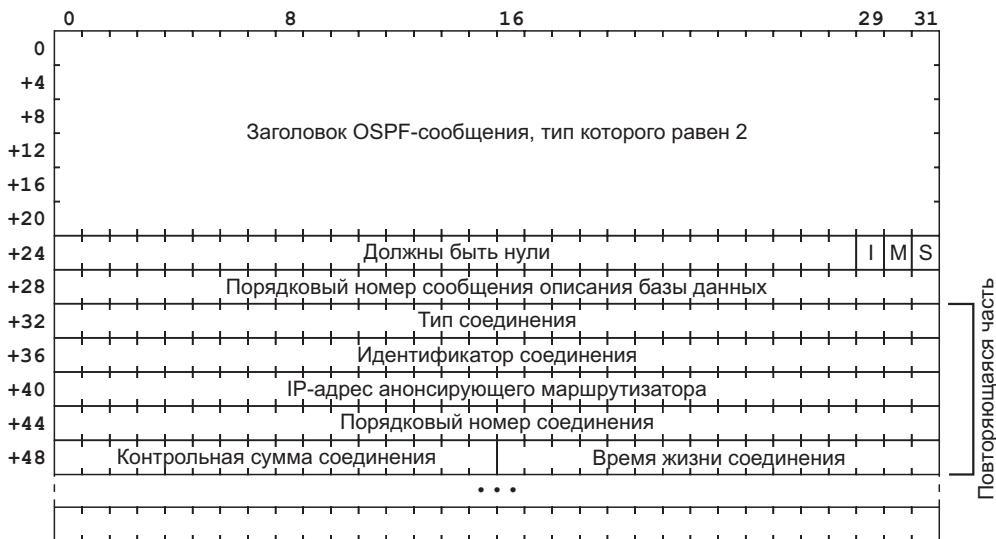


Рис. 16.9. Формат OSPF-сообщения описания базы данных. Для каждого определяемого соединения указывается своя группа полей, начиная с поля типа соединения

Поскольку база данных, содержащая информацию о топологии сети, может достигать больших размеров, ее описание можно разделить на несколько сообщений. Для этого предназначены биты *I* и *M*. В первом (исходном) сообщении биту *I* присваивается значение 1. Биту *M* присваивается значение 1 только в том случае если за этим сообщением последует еще одно сообщение. Бит *S* показывает, кем именно было послано сообщение: главным (1) или подчиненным (0) маршрутизатором. В поле *порядкового номера сообщения описания базы данных* помещаются последовательные числа, чтобы получатель смог определить, все ли сообщения ему были доставлены. В этом поле исходного сообщения помещается случайное целое число *R*, а во все последующие сообщения — последовательность целых чисел, начиная с *R+1*.

Поля, начиная от поля *типа соединения* и заканчивая полем *времени жизни соединения*, описывают одно соединение в топологии сети. Эта группа полей повторяется для каждого соединения. В поле *типа соединения* указывается одно из значений, приведенных в табл. 16.3.

Поле *идентификатора соединения* используется для обозначения соединения (в зависимости от типа соединения в нем указывается либо IP-адрес маршрутизатора либо IP-адрес сети).

В поле *IP-адреса анонсирующего маршрутизатора* указывается адрес маршрутизатора, анонсирующего соединение. Поле *порядкового номера соединения* содержит целое значение, которое гарантирует, что сообщения не затеряются в пути

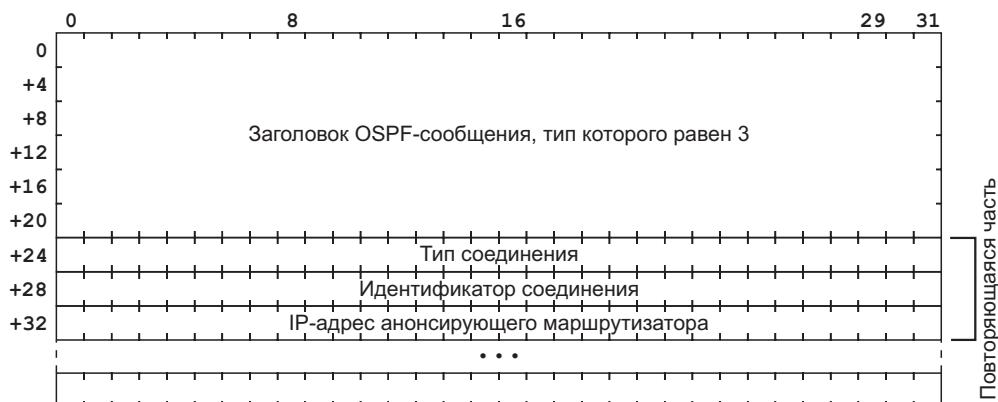
и будут доставлены в правильном порядке. С помощью поля *контрольной суммы информации о соединении* получатель может проверить целостность информации о соединении. И, наконец, поле *времени жизни соединения* помогает упорядочить полученные сообщения. В нем указывается время в секундах, которое прошло с момента установки соединения.

**Таблица 16.3. Типы OSPF-соединений**

Тип соединения	Описание
1	Соединение с маршрутизатором
2	Сетевое соединение
3	Суммарное соединение (IP-сеть)
4	Суммарное соединение (соединение с граничным маршрутизатором)
5	Внешнее соединение (соединение с другим сетевым центром)

#### 16.9.4. Формат OSPF-запроса о состоянии соединения

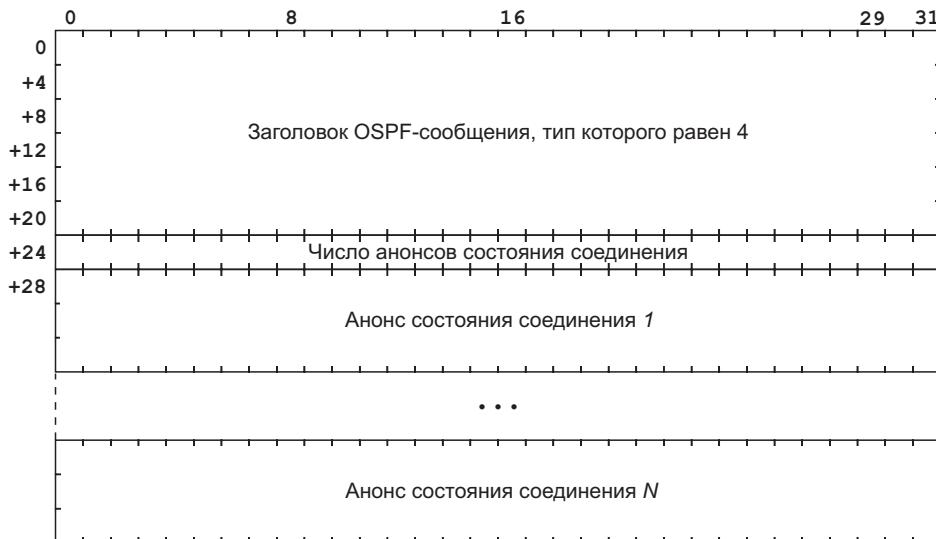
После обмена сообщениями об описании базы данных с соседним устройством маршрутизатор может обнаружить, что отдельные элементы его базы данных уже устарели. Чтобы получить от соседнего устройства обновленную информацию, маршрутизатор отсылает ему *запрос о состоянии соединения*. В таком запросе перечисляются определенные виды соединений, как показано на рис. 16.10. Соседний маршрутизатор отсылает в ответ на полученный запрос самую свежую информацию, которой он располагает об этих соединениях. Показанные на рис. 16.10 три поля повторяются для каждого вида соединения, состояние которого запрашивается. Если список соединений длинный, он разбивается на несколько сообщений.



*Рис. 16.10. Формат OSPF-запроса о состоянии связи. Маршрутизатор отсылает этот запрос соседнему маршрутизатору, чтобы получить от него свежую информацию об определенных соединениях*

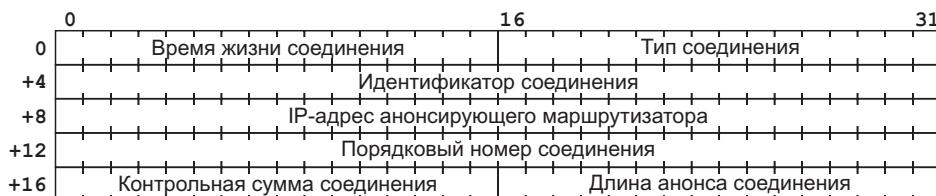
### 16.9.5. Формат OSPF-сообщения об обновлении состояния соединения

Маршрутизаторы отсылают сообщение об обновлении состояния соединения в широковещательном режиме. Каждое такое сообщение состоит из списка анонсов, как показано на рис. 16.11.



*Рис. 16.11. Формат OSPF-сообщения об обновлении состояния соединения. Маршрутизатор отсылает это сообщение в широковещательном режиме для того, чтобы сообщить другим маршрутизаторам информацию о состоянии соединений, которые непосредственно к нему подключены*

Каждый анонс состояния соединения начинается с заголовка стандартного формата, который показан на рис. 16.12. Назначение полей этого заголовка такое же, как и в сообщении описания базы данных.



*Рис. 16.12. Формат заголовка, который используется для всех анонсов состояния соединения*

После заголовка следует информация, описывающая состояние данного соединения, которая может быть представлена в одном из четырех возможных форматов, в зависимости от типа соединения:

- между маршрутизатором и указанной зоной сети;
- между маршрутизатором и указанной сетью;
- между маршрутизатором и физической сетью, состоящей из одной IP-подсети (см. главу 10, “Бесклассовая адресация и подсети (CIDR)”);
- между маршрутизатором и сетью другого сетевого центра.

Во всех случаях тип используемого формата определяется значением поля *типа соединения* заголовка анонса. Поэтому маршрутизатор, который получает сообщение об обновлении состояния соединения точно знает, какой из описанных в сообщении получателей находится внутри сетевого центра, а какой за его пределами.

## 16.10. Маршрутизация при неполной информации

Мы начали обсуждение структуры маршрутизации в объединенной сети с рассмотрения концепции маршрутизации при неполной информации. Выше отмечалось, что узлы сети могут осуществлять маршрутизацию, располагая неполной информацией, поскольку они почти всегда посылают свои пакеты маршрутизаторам. Однако, должно быть уже понятно, что не все маршрутизаторы объединенной сети могут располагать полной маршрутной информацией. Большинство автономных систем имеет только один маршрутизатор, который соединяет эту автономную систему с другими автономными системами. Например, если сетевой центр подключен к глобальной сети Internet, то хотя бы один из его маршрутизаторов должен иметь соединение с провайдером Internet. Маршрутизаторы автономной системы располагают полной информацией о получателях, принадлежащих к этой автономной системе. При отправке дейтаграмм за пределы автономной системы, они посылают их по стандартному маршруту, ведущему к провайдеру Internet.

Процесс маршрутизации при неполной информации становится понятным при анализе содержимого локальных таблиц маршрутизации. Маршрутизаторы, расположенные в ядре глобальной сети Internet, располагают полным набором маршрутов ко всем возможным получателям, информацию о которых они получили от арбитражной системы маршрутизации. В таких маршрутизаторах не используются стандартные маршруты. Следовательно, если адрес сети получателя почему-то отсутствует в базе данных арбитражной системы маршрутизации, то это можно объяснить только двумя причинами:

- адрес является недействительным IP-адресом сети получателя;
- адрес действителен, но получатель по этому адресу почему-то недоступен (например, в результате выхода из строя ведущих к этому получателю каналов связи или маршрутизаторов).

Маршрутизаторы, которые находятся за пределами ядра сети Internet, как правило, не располагают полным набором маршрутов. Поэтому, они должны отправлять пакеты, адресованные неизвестным им получателям, по стандартному маршруту.

Использование маршрутизаторами стандартных маршрутов приводит к двум последствиям. Во-первых, это означает, что ошибки в локальной системе маршрутизации могут остаться незамеченными. Например, если вместо локального маршрутизатора автономной системы одна из машин по ошибке направит пакет во внешнюю автономную систему, последняя отошлет этот пакет обратно (возможно, даже через другое соединение). Поэтому соединение с получателем может быть установлено даже при совершено неправильной маршрутизации. Эта проблема может оказаться не очень серьезной для небольших автономных систем, состоящих из высокоскоростных локальных сетей. Однако в глобальной сети неверные маршруты могут привести к пагубным последствиям. Во-вторых (положительный момент), использование стандартных маршрутов там где это возможно, позволяет уменьшить размер сообщений об обновлении маршрутной информации, которыми обмениваются большинство маршрутизаторов, чем если бы в них содержалась полная информация.

## 16.11. Резюме

Способ обмена маршрутной информацией между локальными маршрутизаторами автономной системы устанавливают сетевые администраторы. Ручное управление маршрутной информацией возможно только в небольших, не претерпевающих частых изменений объединенных сетях с минимальным количеством внутренних соединений. В большинстве случаев возникает необходимость в создании автоматического механизма, с помощью которого автоматически обнаруживаются и обновляются маршруты. Для обмена маршрутной информацией между двумя маршрутизаторами, находящимися под контролем одного административного центра, используется протокол внутреннего шлюза IGP.

Для реализации протокола IGP используется либо дистанционно-векторный алгоритм, либо алгоритм маршрутизации, отслеживающий состояние соединения, который называется алгоритмом поиска кратчайшего пути (Shortest Path First, SPF). В этой главе были рассмотрены три разных протокола IGP: RIP, HELLO и OSPF. Протокол RIP является одним из самых распространенных протоколов внутреннего шлюза. Он относится к классу дистанционно-векторных протоколов и реализован в виде программы для системы UNIX, которая называется *routed*. Для устранения маршрутных петель и решения проблемы медленной сходимости в протоколе RIP используются методы разделения горизонта, замораживания изменений и обратного исправления. Хотя протокол HELLO уже устарел, он представляет определенный интерес с познавательной точки зрения. Дело в том, что в нем применяется дистанционно-векторный алгоритм, в котором расстояние до сети получателя оценивается не по количеству переходов через маршрутизаторы, а по задержке передачи пакетов. Выше были рассмотрены недостатки метода оценки значений задержки, используемой в качестве метрики расстояния, основным из которых является нарушение стабильной работы системы маршрутизации. Мы отмечали, что эвристические алгоритмы могут устранить эту нестабильность при условии, что все маршруты имеют одинаковую пропускную способность. Если же пропускная способность маршрутов разная, система маршрутизации может работать нестабильно на протяжении длительного времени. И наконец, в протоколе OSPF используется алгоритм, отслеживающий состояние соединения.

Также было отмечено, что программа *gated* создана для обеспечения взаимодействия между протоколом внутреннего шлюза (такого как RIP) и протоколом внешнего шлюза (такого как BGP). Она позволяет автоматизировать процесс сбора маршрутной информации, поступающей от источников, которые находятся в пределах автономной системы, и анонсировать их другим автономным системам.

## Материал для дальнейшего изучения

Алгоритмы, используемые для обмена маршрутной информацией, в общих чертах описаны Хедриком (Hedrick) в [RFC 1058]. Там же приведена спецификация стандарта протокола RIP1. Протокол RIP2 описан Малкиным (Malkin) в [RFC 2453]. Протокол HELLO описан Миллсом (Mills) в [RFC 891]. Проблемы преобразования метрик задержки и подсчета числа переходов рассмотрены в статье Миллса и Брауна (Braun) [88]. Достаточно длинное описание протокола OSPF сделано Мойем (Moy) в [RFC 1583]. Там же обсуждаются причины его возникновения. Программа *gated* описана в статье Федора (Fedor) [51].

## Упражнения

- 16.1. Какие семейства сетей поддерживает протокол RIP? (*Подсказка.* Прочтите сетевой раздел руководства программиста BSD UNIX версии 4.3.)
- 16.2. Предположим, что в большой автономной системе используется протокол внутреннего шлюза (типа HELLO), в котором используется алгоритм оценки задержек. Какие проблемы могут возникнуть в такой автономной системе, если в маршрутизаторах одной из подгрупп будет решено использовать протокол RIP?
- 16.3. В RIP-сообщении каждый IP-адрес выравнивается по 32-битовой границе. Будут ли такие адреса оставаться выровненными по 32-битовой границе, если IP-дейтаграмма, передающая это сообщение, располагается с 32-битовой границы?
- 16.4. Размеры автономной системы могут меняться в очень широких пределах: от одной локальной сети, до большой распределенной сети. Почему такой разброс размеров создает трудности в выборе стандартного протокола IGP?
- 16.5. Опишите условия, при которых проблема медленной сходимости может быть решена с помощью методики разделения горизонта.
- 16.6. Предположим, что объединенная сеть состоит из большого количества локальных сетей, в которых в качестве протокола IGP используется протокол RIP. Опишите условия, при которых после получения информации о недостижимости сети может возникнуть маршрутная петля, хотя в программе маршрутизации используется прием замораживания изменений.
- 16.7. Должна ли программа, поддерживающая протокол RIP, работать в активном режиме на обычном узле сети? Поясните свой ответ.
- 16.8. При каких условиях использование метода подсчета числа переходов приведет к вычислению более эффективных маршрутов, чем при использовании метода оценки значения задержки?
- 16.9. При каких условиях администрация автономной системы может принять решение не анонсировать все свои сети? (*Подсказка.* Рассмотрите университетскую сеть).
- 16.10. Говоря в общем, можно сказать, что протокол RIP распространяет информацию из локальной таблицы маршрутизации, а протокол BGP — информацию о сетях и маршрутизаторах, используемых для их достижения (т.е. маршрутизатор может отослать BGP-сообщение, информация в котором не обязательно должна совпадать с информацией, содержащейся в его локальной таблице маршрутизации). Опишите преимущества каждого из подходов.
- 16.11. Опишите алгоритм функции, используемой для преобразования метрик задержки и подсчета числа переходов. Можете ли вы назвать характеристики этой функции, достаточные для предотвращения образования петель маршрутизации. Являются ли эти характеристики необходимыми? (*Подсказка.* Прочтайте статью Миллса и Брауна [88].)
- 16.12. При каких условиях протокол SPF может вызвать образование маршрутных петель? (*Подсказка.* Рассмотрите метод негарантированной доставки сообщений).

- 16.13.** Напишите программу, которая отсылает RIP-запрос маршрутизатору и отображает полученную от него информацию.
- 16.14.** Внимательно прочтайте спецификацию протокола RIP. Может ли информация, присланная маршрутизатором в ответ на запрос, отличаться от информации, содержащейся в сообщении об обновлении маршрутов? Если да, то почему?
- 16.15.** Внимательно прочтайте спецификацию протокола OSPF. Как администратор может использовать возможность установки виртуального соединения?
- 16.16.** Протокол OSPF позволяет администраторам самостоятельно назначать большинство своих идентификаторов. В результате их значения могут дублироваться в разных сетевых центрах. Значения каких идентификаторов придется изменить при объединении двух сетевых центров, в которых используется протокол OSPF?
- 16.17.** Сравните версии программ, реализующих протоколы OSPF и RIP в операционной системе 4BSD UNIX. Какие различия существуют в размере исходных программ? В размере объектного кода? В размере области данных? Какие выводы из этого можно сделать?
- 16.18.** Можно ли использовать ICMP-сообщения о переадресации для обмена маршрутной информацией между *внутренними* маршрутизаторами? Поясните свой ответ.
- 16.19.** Напишите программу, которая вводит информацию о топологии объединенной сети вашей организации, посыпает запросы по протоколу RIP и получает от маршрутизаторов маршрутную информацию, после чего выводит сообщения о возникших противоречиях.
- 16.20.** Если в вашем сетевом центре используется программа `gated`, проанализируйте ее файл конфигурации и объясните назначение каждого параметра.



# 17

## Режим многоадресатной передачи в объединенной сети

### 17.1. Введение

В предыдущих главах была описана первоначальная структура классовой адресации протокола IP, а также ее расширения — адресация подсетей и бесклассовая адресация. В этой главе исследуется еще одна возможность системы адресации протокола IP, которая обеспечивает доставку одной дейтаграммы нескольким получателям (*multipoint delivery*, или *многоточечная доставка*). В начале главы приводится краткий обзор сетевого оборудования, благодаря которому становится возможной многоадресатная передача. В последующих разделах описана система IP-адресации, обеспечивающая многоточечную доставку, а также протоколы, используемые маршрутизаторами для распространения необходимой маршрутной информации.

### 17.2. Аппаратное широковещание

В большинстве технологий сетевого аппаратного обеспечения предусмотрены механизмы, позволяющие отправить один пакет одновременно (или почти одновременно) нескольким получателям. В главе 2, “Обзор основных сетевых технологий”, было рассмотрено несколько таких технологий, а также описана самая распространенная форма многоточечной доставки — *широковещание (broadcasting)*. Широковещательная доставка сообщений означает, что сетевое оборудование доставляет каждому получателю копию одного и того же пакета. В шинных сетевых технологиях наподобие Ethernet широковещательная доставка может выполняться посредством передачи одного пакета сразу всем получателям. В сетях, состоящих из коммутаторов, соединенных двухточечными выделенными каналами связи, широковещательный режим передачи реализуется на уровне сетевого программного обеспечения. При этом копии пакета отправляются по каждому сетевому соединению и доставляются всем коммутаторам.

В большинстве технологий сетевого оборудования режим широковещательной доставки активизируется самим компьютером отправителя. При этом пакет посыпается по специальному адресу получателя, который называется *широковещательным (broadcast address)*. Например, длина физического адреса Ethernet составляет 48 бит. Если значение каждого бита адреса равно единице, то такой адрес считается широковещательным. Сетевое оборудование, установленное на каждой машине, опознает свой аппаратный адрес, а также широковещательный адрес, и принимает входящие пакеты, если в качестве адреса получателя указан любой из этих адресов.

Главный недостаток широковещательного режима передачи заключается в привлечении дополнительных сетевых ресурсов. Рассылка каждого широковещательного сообщения снижает пропускную способность сети, а также требует затрат времени центрального процессора на всех машинах. Например, можно создать альтернативное семейство протоколов IP, в котором для доставки дейтаграмм по локальной сети использовался бы широковещательный режим передачи. При этом дейтаграммы, не предназначенные для локальной машины, можно было бы отвергать программно на уровне протокола IP. Однако такая структура была бы крайне неэффективной, поскольку все компьютеры локальной сети получали и обрабатывали бы каждую дейтаграмму. Очевидно, что при этом большинство поступивших дейтаграмм отвергалось бы. Поэтому разработчики семейства протоколов TCP/IP использовали одноадресатный режим маршрутизации (unicast routing) и механизмы привязки адресов наподобие ARP, чтобы исключить широковещательную доставку пакетов.

### 17.3. Аппаратная поддержка режима многоадресатной передачи

Существует сетевое оборудование, поддерживающее вторую, менее распространенную форму многоточечной доставки пакетов, которая называется *многоадресатной передачей (multicasting)*. В отличие от широковещания, многоадресатная передача позволяет каждому узлу сети самостоятельно решать, стоит ли участвовать в подобной рассылке. Как правило, для обеспечения режима многоадресатной передачи на уровне сетевого оборудования резервируется большой набор физических адресов. Поэтому если компьютеры группы захотят обменяться между собой данными, они должны выбрать один из зарезервированных многоадресатных *адресов (multicast address)* и использовать его для обмена информацией. Если плату сетевого интерфейса компьютера сконфигурировать так, чтобы она могла распознавать выбранный многоадресатный адрес, то можно сформировать группу машин, которые будут получать по копии посланного по этому адресу пакета.

На абстрактном уровне многоадресатную адресацию можно рассматривать как обобщенную форму всех остальных методов адресации. Например, обычный *одноадресатный адрес (unicast address)* можно считать частным случаем многоадресатной адресации, при которой многоадресатная группа состоит только из одного компьютера. В то же время направленную широковещательную адресацию (*directed broadcast addressing*) можно считать формой многоадресатной передачи, при которой все компьютеры в определенной сети являются членами *многоадресатной группы*. Другие многоадресатные адреса могут соответствовать произвольным совокупностям машин.

Несмотря на явно общий характер многоадресатная передача не может заменить обычные формы адресации, поскольку между базовыми механизмами, реализующими пересылку и доставку пакетов, существуют некоторые отличия. Одноадресатный и широковещательный адреса определяют компьютер или совокупность компьютеров, подключенных к одному физическому сегменту, поэтому пересылка пакетов (*forwarding*) зависит от топологии сети. Многоадресатный адрес определяет произвольный набор получателей, поэтому механизм пересылки пакетов должен распространить данный пакет всем сегментам сети. В качестве примера рассмотрим два сегмента локальной сети, соединенных адаптивным мостом, который располагает информацией обо всех адресах своих узлов. Если узел сети, расположенный в сегменте 1, отсылает одноадресатный фрейм (*unicast frame*) другому узлу в сегменте 1, мост не перешлет этот фрейм в сегмент 2.

Но если узел сети использует многоадресатный адрес, мост должен переслать этот фрейм в сегмент 2. Таким образом, можно сделать следующий вывод.

*Хотя многоадресатная адресация, которая является обобщенной формой адресации (т.е. одноадресатные и широковещательные адреса являются ее частным случаем) имеет свои преимущества, базовые механизмы пересылки и доставки пакетов могут уменьшить эффективность ее использования.*

## 17.4. Многоадресатная передача в сети Ethernet

Локальная сеть Ethernet представляет собой хороший пример поддержки многоадресатной передачи пакетов на уровне сетевого оборудования. При этом для многоадресатной передачи зарезервирована половина физических адресов Ethernet. Обычные адреса отличаются от многоадресатных адресов значением младшего бита старшего октета физического адреса. Если значение этого бита равно 0, адрес считается одноадресатным, если 1 — многоадресатным. В шестнадцатеричной точечной форме записи<sup>1</sup> бит групповой передачи можно представить в следующем виде:

01.00.00.00.00.00<sub>16</sub>

После инициализации платы сетевого интерфейса Ethernet начинает принимать пакеты, отправленные по ее физическому либо широковещательному адресу. Однако программа драйвера устройства может реконфигурировать плату сетевого интерфейса так, чтобы она могла распознавать также один или несколько многоадресатных адресов. Предположим, что драйвер задал сетевой плате Ethernet следующий многоадресатный адрес:

01.5E.00.00.00.01<sub>16</sub>

После завершения конфигурации плата сетевого интерфейса будет принимать пакеты, посланные по ее физическому, широковещательному или по указанному выше многоадресатному адресу. При этом плата будет игнорировать пакеты, отправленные по другим многоадресатным адресам. В следующих разделах будет описано, как в протоколе IP осуществляется режим многоадресатной передачи пакетов с помощью сетевого оборудования, а также приведена информация о многоадресатных адресах специального назначения.

## 17.5. Многоадресатная передача в протоколе IP

Режим многоадресатной передачи в протоколе IP представляет собой обобщенную форму многоадресатной передачи, поддерживаемой на уровне сетевого оборудования, которая распространена по всей объединенной сети. При этом разработчики хотели обеспечить передачу одной копии дейтаграммы произвольному подмножеству узлов, которые могут быть расположены в произвольной физической сети, подключенной к объединенной сети. В терминологии протокола IP такое подмножество узлов называется *многоадресатной группой*. Многоадресатная передача в протоколе IP имеет следующие особенности.

- *Многоадресатный адрес.* Каждой многоадресатной группе назначается уникальный IP-адрес класса D. Существует несколько глобальных многоадресатных IP-адресов, определенных центральным административным

<sup>1</sup> В этой форме записи каждый октет представляется в виде двух шестнадцатеричных цифр, разделенных точками. Нижний индекс<sub>16</sub> может быть опущен при условии, что из контекста ясно, о чём идет речь.

органом сети Internet как постоянные. Они соответствуют группам, которые существуют постоянно, даже если в текущий момент в них нет участников. Остальные многоадресатные адреса являются временными, и могут использоваться в частных сетях.

- *Количество групп.* В протоколе IP одновременно может существовать до  $2^{28}$  многоадресатных групп. Однако количество групп определяется не возможностями адресации, а действующими ограничениями, которые налагаются на размер таблицы маршрутизации.
- *Динамическое подключение к группе.* Сетевой узел может в любой момент присоединиться к многоадресатной группе протокола IP или отключиться от нее. Кроме того, сетевой узел может быть членом любого количества многоадресатных групп.
- *Использование аппаратного обеспечения.* Если базовое сетевое аппаратное обеспечение поддерживает режим многоадресатной передачи, то в протоколе IP используется аппаратная поддержка отсылки многоадресатных IP-пакетов. Если аппаратное обеспечение не поддерживает многоадресатную передачу, протокол IP использует для доставки многоадресатных IP-пакетов режим широковещательной или одноадресатной передачи.
- *Пересылка пакетов между сетями.* Поскольку члены многоадресатной группы протокола IP могут быть подключены к разным физическим сетям, возникает необходимость в создании особых многоадресатных маршрутизаторов (*multicast routers*) для пересылки многоадресатных IP-пакетов. Как правило, режим многоадресатной маршрутизации поддерживается обычными маршрутизаторами.
- *Негарантированная доставка пакетов.* При многоадресатной передаче в протоколе IP используется такой же принцип негарантированной доставки, как и при передаче любой другой IP-дейтаграммы. Это означает, что многоадресатная дейтаграмма может не достичь одного или нескольких получателей, прийти к получателю с задержкой или в неправильном порядке, а также в сети может появиться одна или несколько ее копий.
- *Членство и передача.* Произвольный узел сети может отсыпать дейтаграммы любой многоадресатной группе. Понятие членства группы используется только для того чтобы определить, получил ли узел сети отосланную этой группе дейтаграмму.

## 17.6. Три аспекта многоадресатной передачи

Создавая универсальную систему многоадресатной передачи в объединенной сети, необходимо продумать три следующих аспекта:

- структура многоадресатной адресации;
- эффективный механизм уведомления и доставки;
- эффективный механизм пересылки пакетов между сетями.

При создании общей структурной схемы системы необходимо учесть множество противоречивых факторов и стоящих перед разработчиками целей. Например, кроме резервирования достаточного числа адресов для большого количества групп, система многоадресатной адресации должна удовлетворять двум взаимоисключающим условиям: дать возможность локальным административным группам самостоятельно назначать адреса узлам сети и в то же время обеспечить поддержку глобальных адресов, действительных по всей сети Internet. Точно так

же узлам сети необходим *механизм уведомления* маршрутизаторов о многоадресатных группах, членами которых они являются, а маршрутизаторам, в свою очередь, необходим *механизм доставки* многоадресатных пакетов узлам сети. И снова существует два противоречия: система должна эффективно использовать поддержку *многоадресатной* передачи на уровне сетевого оборудования (если, конечно, она там реализована) и одновременно обеспечивать многоадресатную доставку IP-пакетов по сетям, в которых отсутствует аппаратная поддержка *многоадресатной* передачи. В конечном счете создание *механизма пересылки многоадресатных пакетов* представляет для разработчиков самую трудную задачу: целью является создание структуры, одновременно и эффективной, и динамичной. Такая система должна передавать многоадресатные пакеты по кратчайшим маршрутам, не посылать копию дейтаграммы по маршруту, если он не ведет к члену группы, и позволять узлам сети в любой момент присоединяться к группам и отключаться от них.

Многоадресатная передача протокола IP включает в себя все три упомянутых выше аспекта. Она определяет многоадресатную IP-адресацию; устанавливает, каким образом узлы сети отсылают и получают многоадресатные дейтаграммы; и описывает протокол, который используют маршрутизаторы для определения членов *многоадресатной* группы в произвольной сети. Далее в этой главе каждый из этих аспектов рассматривается более подробно, а начнем мы со структуры адресации.

## 17.7. Многоадресатные IP-адреса

Как уже было сказано, многоадресатные IP-адреса делятся на два типа: те, что присваиваются на длительный период, и доступные только для временного использования. Постоянные адреса называются *общезвестными* и используются для работы основных служб сети Internet, а также для поддержки ее инфраструктуры (например, протоколов многоадресатной маршрутизации). Другие многоадресатные адреса соответствуют *временным* многоадресатным группам, которые создаются по мере необходимости и аннулируются, если количество членов группы достигает нуля.

Как и в случае многоадресатной передачи, поддерживаемой на уровне сетевого оборудования, режим многоадресатной передачи в протоколе IP задается отправителем дейтаграммы путем помещения специального (многоадресатного) IP-адреса в поле адреса получателя заголовка IP-дейтаграммы. Для многоадресатной передачи в протоколе IP зарезервированы адреса класса D, формат которых показан на рис. 17.1.

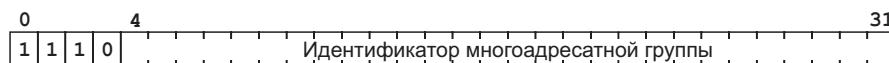


Рис. 17.1. Формат IP-адресов класса D, которые используются для многоадресатной передачи. Биты 4–31 идентифицируют многоадресатную группу

Значение первых четырех битов равно 1110, оно определяет адрес как многоадресатный. Остальные 28 бит идентифицируют многоадресатную группу. Биты идентификатора группы не разбиваются на компоненты. В частности, поле идентификатора группы не разбивается на биты, которые определяют происхождение группы или ее владельца, а также не содержит служебной информации, например, о том, находятся ли все члены группы в одной физической сети.

В точечной десятичной форме записи многоадресатные адреса выражаются значениями, лежащими в диапазоне от 224.0.0.0 до 239.255.255.255. Часть адресов из указанного диапазона выделена для специального использования. Например, самый

нижний адрес 224.0.0.0 зарезервирован и не может быть назначен ни одной из групп. Кроме того, остальные адреса до 224.0.0.255 применяются только при многоадресатной маршрутизации и в протоколах управления группами. Маршрутизатору запрещается пересыпать дейтаграммы, посланные по любому из адресов указанного диапазона. В табл. 17.1 показано несколько примеров постоянных многоадресатных адресов. Большинство других адресов имеют особые значения.

**Таблица 17.1. Несколько примеров постоянных групповых IP-адресов**

<b>Адрес</b>	<b>Описание</b>
224.0.0.0	Базовый адрес (зарезервирован)
224.0.0.1	Все компьютеры в данной подсети
224.0.0.2	Все маршрутизаторы в данной подсети
224.0.0.3	Свободен
224.0.0.4	Маршрутизаторы протокола DVMRP
224.0.0.5	Все маршрутизаторы протокола OSPFIGP
224.0.0.6	Выделенные маршрутизаторы протокола OSPFIGP
224.0.0.7	Маршрутизаторы ST
224.0.0.8	Узлы сети ST
224.0.0.9	Маршрутизаторы протокола RIP2
224.0.0.10	Маршрутизаторы протокола IGRP
224.0.0.11	Мобильные агенты
224.0.0.12	Сервер протокола DHCP/ретранслирующий агент
224.0.0.13	Все маршрутизаторы протокола PIM
224.0.0.14	Инкапсуляция протокола RSVP
224.0.0.15	Все маршрутизаторы протокола CBT
224.0.0.16	Выделенные маршрутизаторы протокола SBM
224.0.0.17	Все маршрутизаторы протокола SBM
224.0.0.18	VRBP
224.0.0.19	Свободные адреса
...	
224.0.0.255	
224.0.1.21	Поддержка протокола DVMRP на основе протокола MOSPF
224.0.1.84	Анонс Jini
224.0.1.85	Запрос Jini
239.192.0.0	Блок многоадресатных адресов, относящихся к одной организации
...	
239.251.255.255	
239.252.0.0	Блок многоадресатных адресов, относящихся к одному сетевому центру
...	
239.255.255.255	

Ниже мы увидим, что два из приведенных в табл. 17.1 адреса имеют важное значение для работы механизма доставки многоадресатных пакетов. Адреса 224.0.0.1 и 224.0.0.2 являются постоянными. Первый — соответствует *группе всех машин данной сети*, а второй — *группе всех маршрутизаторов*. В группу *всех машин* включены все узлы и маршрутизаторы данной сети, которые принимают участие в многоадресатной передаче по протоколу IP, а в группу *всех маршрутизаторов* включены только участвующие в многоадресатной передаче маршрутизаторы. В общем обе группы используются для работы протоколов управления группой, а не для обычной доставки данных. Кроме того, посланные по этим адресам дейтаграммы достигнут только тех машин, которые подключены к той же локальной сети, что и отправитель. Не существует многоадресатных IP-адресов, которые относятся ко всем машинам или всем маршрутизаторам объединенной сети.

## 17.8. Особенности использования многоадресатного адреса

В протоколе IP многоадресатные адреса интерпретируются иначе, чем обычные одноадресатные адреса. Например, многоадресатный адрес может использоваться только в поле адреса получателя. Поэтому в поле адреса отправителя заголовка дейтаграммы многоадресатный адрес указывать нельзя. Точно так же его нельзя указывать в области параметров маршрутизации от источника или регистрации маршрута IP-дейтаграммы. Кроме того, в процессе доставки многоадресатных дейтаграмм не могут генерироваться ICMP-сообщения об ошибке (например, о недоступности получателя, подавлении источника, ответе на запрос эха или истечении тайм-аута.) Поэтому запрос программы ping, направленный по многоадресатному адресу, останется без ответа.

Правило, запрещающее генерацию ICMP-сообщений об ошибках, в некоторой степени неожиданно, поскольку IP-маршрутизаторы все же обрабатывают значение поля времени жизни заголовка многоадресатной дейтаграммы. При передаче дейтаграммы каждый маршрутизатор, как обычно, вычитает единицу из счетчика времени жизни дейтаграммы и “молч” (не генерируя ICMP-сообщений) аннулирует дейтаграмму, если значение счетчика становится равным нулю. Ниже будет показано, что в некоторых протоколах значение счетчика времени жизни дейтаграммы используется для ограничения ее распространения по объединенной сети.

## 17.9. Преобразование многоадресатного IP-адреса в физический адрес Ethernet

Хотя стандарт многоадресатной передачи протокола IP не рассчитан на все виды сетевого оборудования, в нем все же определено, как преобразовать многоадресатный IP-адрес в физический многоадресатный адрес локальной сети Ethernet. Такое преобразование является эффективным, поскольку его очень легко осуществить.

*Чтобы преобразовать многоадресатный IP-адрес в соответствующий ему физический многоадресатный адрес сети Ethernet, необходимо поместить младшие 23 бита IP-адреса в младшие 23 бита особого физического многоадресатного адреса Ethernet 01.00.5E.00.00.00<sub>16</sub>.*

Например, многоадресатный IP-адрес 224.0.0.2 преобразуется в физический многоадресатный адрес Ethernet 01.00.5E.00.00.02<sub>16</sub>.

Что интересно, такое преобразование не является однозначным. Поскольку многоадресатные IP-адреса имеют 28 значимых битов, которые определяют многоадресатную группу, одному и тому же физическому многоадресатному адресу Ethernet может соответствовать одновременно несколько многоадресатных IP-адресов. Разработчики выбрали такой алгоритм преобразования в качестве компромиссного решения. С одной стороны, использование 23 из 28 битов для аппаратного адреса означает, что в него включена большая часть многоадресатного IP-адреса. Набор многоадресатных IP-адресов довольно большой, поэтому вероятность того, что две группы изберут адреса с абсолютно одинаковыми 23-мя младшими битами, невелика. С другой стороны, применение в протоколе IP фиксированной части многоадресатного физического адресного пространства Ethernet значительно облегчает выявление неполадок в сети и исключает создание взаимных помех между протоколом IP и другими протоколами, работающими в локальной сети Ethernet. Подобный подход к использованию многоадресатной адресации может привести к тому, что некоторые многоадресатные дейтаграммы будут доставлены узлам сети, которые не входят в заданную группу. Таким образом, программное обеспечение протокола IP должно проверять адреса во всех входящих дейтаграммах и аннулировать ненужные дейтаграммы.

## 17.10. Сетевые узлы и многоадресатная доставка

Выше уже отмечалось, что многоадресатная передача в протоколе IP может осуществляться как в одной физической сети, так и во всей объединенной сети. В первом случае узел сети может непосредственно отослать дейтаграмму узлу получателя, поместив ее в физический фрейм и указав в качестве адреса получателя физический адрес группы, к которой подключен получатель. Во втором случае доставкой многоадресатных дейтаграмм между сетями будут заниматься специальные *многоадресатные маршрутизаторы*. Поэтому узел сети должен отослать такую дейтаграмму многоадресатному маршрутизатору. Удивительно то, что узлу сети не нужно “знать” маршрут к многоадресатному маршрутизатору или использовать для этой цели стандартный маршрут, установленный по умолчанию. Прием, используемый узлами сети для отсылки многоадресатной дейтаграммы маршрутизатору, не похож на тот, который используется для поиска маршрутов при отправке одноадресатных и широковещательных дейтаграмм. Для отсылки многоадресатных дейтаграмм узел сети использует возможности многоадресатной передачи сетевого оборудования. Многоадресатные маршрутизаторы обрабатывают все полученные многоадресатные IP-дейтаграммы. Поэтому, если к данной физической сети подключен многоадресатный маршрутизатор, он получит дейтаграмму и при необходимости перешлет ее в другую сеть. Таким образом, главное отличие между локальной и глобальной многоадресатной передачей заключается в устройстве многоадресатных маршрутизаторов, а не узлов сети.

## 17.11. Область действия многоадресатной рассылки

Под *областью действия* (scope) многоадресатной группы подразумевается пространство сети, занимаемое членами этой группы. Если все члены находятся в одной физической сети, считается, что область действия группы ограничена одной сетью. Аналогично, если все члены группы размещены в пределах одной организации, считается, что область действия группы ограничена одной организацией.

Помимо области действия группы, для каждой многоадресатной дейтаграммы определена область многоадресатной рассылки. Эта область определяется набором сетей, по которым будет распространяться данная дейтаграмма. Иногда область многоадресатной рассылки дейтаграммы называют *размахом* (*range*).

Для контроля над областью многоадресатной рассылки в протоколе IP используется два метода. В первом методе используется значение поля *времени жизни* (TTL) дейтаграммы. Устанавливая в поле TTL небольшое значение, узел сети может ограничить расстояние, на которое будет передаваться дейтаграмма. Например, согласно стандарту в поле TTL дейтаграмм, которые используются для пересылки служебных сообщений между узлом и маршрутизатором в пределах одной сети, должно помещаться значение 1. В результате маршрутизатор никогда не отправит дейтаграмму со служебной информацией во внешний мир, поскольку значение счетчика TTL при переходе ее через этот маршрутизатор становится равным нулю. А это значит, что маршрутизатор попросту аннулирует такую дейтаграмму. Аналогично, если две прикладные программы, запущенные на одном компьютере, будут использовать режим многоадресатной рассылки по протоколу IP для межпроцессного взаимодействия (например, для тестирования программного обеспечения), они должны установить в поле TTL нулевое значение, чтобы дейтаграмма не покинула этот узел сети. Для расширения области многоадресатной рассылки дейтаграммы нужно просто последовательно увеличивать значение поля TTL. Например, некоторые производители предлагают сконфигурировать маршрутизаторы сетевого центра так, чтобы дейтаграммы не покидали пределов сетевого центра, если значение в поле TTL не превышает 15. Таким образом, значение поля TTL заголовка дейтаграммы можно использовать для грубого контроля над областью многоадресатной рассылки дейтаграммы.

Второй метод, используемый для управления областью многоадресатной рассылки, называется *административным*. Суть его заключается в резервировании частей адресного пространства для групп, которые являются локальными по отношению к данному сетевому центру или организации в целом. Согласно стандарту маршрутизаторам в сети Internet запрещается пересыпать дейтаграммы, посланные по многоадресатным адресам, принадлежащим к пространству ограниченных адресов. Таким образом, чтобы в процесс многоадресатного обмена информацией случайно не вовлекались посторонние машины, организация может присвоить группе адрес с локально ограниченной областью рассылки. В табл. 17.1 приведены примеры диапазонов адресов, соответствующие различным административным ограничениям.

## 17.12. Поддержка многоадресатной передачи узлом сети

Узел сети может участвовать в многоадресатной передаче по протоколу IP на одном из трех уровней, как показано в табл. 17.2.

Таблица 17.2. Три уровня участия узла сети в многоадресатной передаче по протоколу IP

Уровень	Описание
0	Сетевой узел не может ни посылать, ни получать многоадресатные IP-дейтаграммы
1	Сетевой узел может посылать, но не может получать многоадресатные IP-дейтаграммы
2	Сетевой узел может как посылать, так и получать многоадресатные IP-дейтаграммы

Внести соответствующие изменения в программу протокола IP, которые позволяют сетевому узлу отсылать многоадресатные IP-дейтаграммы, несложно. Программное обеспечение протокола IP должно позволять прикладной программе задавать в качестве IP-адреса получателя многоадресатный адрес, а драйвер сетевого интерфейса должен уметь конвертировать этот IP-адрес в соответствующий ему физический многоадресатный адрес (или использовать широковещательный режим, если сетевое оборудование не поддерживает многоадресатную передачу).

Расширить программное обеспечение узла сети так, чтобы он мог получать многоадресатные IP-дейтаграммы, гораздо сложнее. В модуле протокола IP должен быть предусмотрен специальный программный интерфейс, с помощью которого прикладная программа сможет объявить о присоединении к определенной многоадресатной группе или отключении от нее. Если к одной и той же группе подключаются несколько прикладных программ, модуль протокола IP должен “не забыть” пересыпать каждой из прикладных программ, входящих в группу, по копии предназначенные для этой группы дейтаграмм. Если все прикладные программы данного узла отключаются от группы, модуль протокола IP должен сделать так, чтобы узел сети больше не являлся членом группы. Кроме того, как будет показано в следующем разделе, узел сети должен поддерживать протокол, с помощью которого он сможет сообщить локальным многоадресатным маршрутизаторам о своем статусе членства группы. Основная сложность описанной выше задачи заключается в следующем.

*Узлы сети могут присоединяться к определенным многоадресатным группам в разных сетях.*

Это значит, что узел, подключенный к нескольким сетям, может присоединиться к определенной многоадресатной группе в одной сети, а не в другой. Чтобы понять, зачем членов группы соотносят с определенными сетями, вспомним, что многоадресатную передачу по протоколу IP можно использовать между локальными наборами машин. Поэтому на узле сети может быть запущена прикладная программа, использующая режим многоадресатной передачи для взаимодействия с машинами только в одной физической сети.

Поскольку понятие членства группы связано с определенными сетями, программное обеспечение должно вести отдельные списки адресов групп для каждой сети, к которой подключена данная машина. Кроме того, прикладная программа должна указывать определенную сеть, когда она посылает запрос на присоединение к многоадресатной группе или отключение от нее.

### 17.13. Межсетевой протокол управления группами (IGMP)

Для участия в многоадресатной передаче по протоколу IP в локальной сети на узле сети должна быть запущена специальная программа, которая позволит ему отсылать и получать многоадресатные дейтаграммы. Чтобы участвовать в многоадресатной передаче, охватывающей несколько сетей, узел сети должен сообщить об этом локальным многоадресатным маршрутизаторам. Локальные маршрутизаторы связываются с другими многоадресатными маршрутизаторами, передают им информацию о членах группы и устанавливают соответствующие маршруты. Ниже будет показано, что этот подход сходен с тем, который использовался при распространении обычной маршрутной информации между маршрутизаторами объединенной сети.

Прежде чем маршрутизатор сможет распространять информацию о членах многоадресатной группы, он должен определить, какие узлы локальной сети принадлежат к этой группе. Для этого многоадресатные маршрутизаторы и узлы сети, поддерживающие многоадресатную передачу, должны использовать *межсетевой протокол управления группами (IGMP)*. С его помощью распространяется информация о членах группы. Поскольку текущая версия протокола IGMP имеет номер 2 (и именно он рассматривается ниже), его официально обозначают аббревиатурой *IGMPv2*.

Протокол IGMP аналогичен протоколу ICMP<sup>2</sup>. Подобно протоколу ICMP, для передачи управляющих сообщений в нем используются IP-дейтаграммы. Как и

<sup>2</sup> Протокол межсетевых управляющих ICMP-сообщений подробно описан в главе 9, “Протокол IP: обработка ошибок и управляющие сообщения (ICMP)”.

ICMP, протокол IGMP используется самим протоколом IP. Таким образом, можно сделать следующий вывод.

*Хотя в протоколе IGMP для передачи управляющих сообщений используется IP-дейтаграммы, он считается неотъемлемой частью протокола IP, а не отдельным протоколом.*

Кроме того, протокол IGMP является стандартом для семейства протоколов TCP/IP. Он должен поддерживаться на всех машинах, которые хотят получать многоадресатные IP-дейтаграммы (т.е. на всех узлах сети и маршрутизаторах, которые участвуют в передаче на уровне 2).

Работа протокола IGMP состоит из двух этапов. Первый этап заключается в следующем: когда произвольный узел сети присоединяется к новой многоадресатной группе, он отсылает IGMP-сообщение по адресу этой группы, объявляя тем самым о своем присутствии в группе. Локальные многоадресатные маршрутизаторы получают это сообщение, настаивающим образом маршруты и распространяют информацию о членах группы другим многоадресатным маршрутизаторам по объединенной сети. Второй этап: поскольку состав членов группы часто меняется, многоадресатные маршрутизаторы должны периодически опрашивать узлы локальной сети, чтобы установить, продолжают ли они являться членами группы. Если какой-нибудь из узлов отвечает от имени данной группы, маршрутизатор оставляет группу в активном состоянии. Если после нескольких запросов маршрутизатора ни один из узлов не сообщает о том, что является членом данной группы, многоадресатный маршрутизатор считает, что группы больше не существует и прекращает анонсировать информацию о членах этой группы другим многоадресатным маршрутизаторам.

## 17.14. Реализация протокола IGMP

Структура протокола IGMP была тщательно продумана для того, чтобы избежать образования избыточного трафика, который может вызвать перегрузку в сетях. В частности, поскольку в одной сети может существовать несколько многоадресатных маршрутизаторов, а также все узлы сети могут принимать участие в многоадресатной передаче, в протоколе IGMP предусмотрено, чтобы не все участники группы создавали служебный трафик. Существует несколько способов, с помощью которых можно максимально снизить влияние служебного трафика протокола IGMP на сеть.

1. Весь обмен информацией между узлами сети и многоадресатными маршрутизаторами осуществляется с помощью режима многоадресатной передачи протокола IP. Это означает, что после инкапсуляции IGMP-сообщения в IP-дейтаграмму для последующей передачи по сети, в качестве IP-адреса получателя указывается многоадресатный адрес. Маршрутизаторы отсылают общие IGMP-запросы по многоадресатному адресу, соответствующему всем компьютерам сети (224.0.0.1), а узлы сети отсылают IGMP-сообщения по многоадресатному адресу, соответствующему всем маршрутизаторам этой сети (224.0.0.2). Однако специфичные для группы IGMP-сообщения отправляются и узлами сети, и маршрутизаторами исключительно по адресу этой группы. Кроме того, дейтаграммы, передающие IGMP-сообщения, отсылаются с помощью многоадресатной передачи на уровне сетевого оборудования, если, конечно, она поддерживается. Поэтому в сетях, которые поддерживают многоадресатную передачу на уровне сетевого оборудования, узлы сети, не участвующие в многоадресатной передаче по протоколу IP, никогда не получают IGMP-сообщений.

2. При опросе узлов сети с целью установления их принадлежности к определенной группе, многоадресатный маршрутизатор отсылает только один запрос на получение информации обо всех группах, а не отдельные запросы к каждой группе<sup>3</sup>. По умолчанию опрос проводится каждые 125 секунд. Поэтому протокол IGMP не создает большого объема трафика.
3. Если к одной сети подключены несколько многоадресатных маршрутизаторов, то для проведения опроса на предмет определения членов группы выбирается только один из маршрутизаторов. Причем процедура выборов происходит быстро и эффективно. Следовательно, по мере подключения к сети дополнительных многоадресатных маршрутизаторов объем создаваемого служебного трафика не увеличивается.
4. Узлы сети не должны одновременно отвечать на поступивший от маршрутизатора IGMP-запрос. Поэтому в каждом запросе указывается число  $N$ , которое определяет максимальное время ответа (по умолчанию его значение рано 10 секундам.) При поступлении запроса узел сети случайным образом выбирает время задержки в диапазоне от 0 до  $N$ , по истечении которого он посыпает ответ на запрос. Если же данный узел является членом нескольких групп, то для каждой группы выбирается свое время задержки. Поэтому ответы узла сети на запрос маршрутизатора будут распределены случайным образом во временном интервале 0 до  $N$ .
5. Каждый участник группы прослушивает ответы, поступающие от других членов группы, и не посыпает ненужных ответных сообщений.

Чтобы понять, почему на запрос многоадресатного маршрутизатора не должны отвечать все члены группы, вспомним, что в маршрутизаторе не хранится список их адресов. Сообщения пересыпаются всей группе с помощью многоадресатной передачи, поддерживаемой на уровне сетевого оборудования. Следовательно, маршрутизатору необходимо только установить, является ли хотя бы один из узлов сети членом группы. Поскольку каждый член группы получает запрос, посланный по многоадресатному адресу, соответствующему всем компьютерам данной сети, перед отправкой ответа каждый узел случайным образом выбирает время задержки и переходит в режим ожидания. Первым посыпает ответ тот узел, у которого время задержки было наименьшим. Поскольку ответ отсылается по адресу группы, его копию получают все ее члены, а также многоадресатный маршрутизатор. При получении ответа другие члены группы сбрасывают таймер задержки и отменяют передачу. Поэтому на практике только один участник каждой группы отвечает на запрос маршрутизатора.

## 17.15. Переходы состояния членов группы

Модуль протокола IGMP узла сети должен отслеживать состояние каждой группы, к которой принадлежит этот узел (например, состояние группы, от которой узел сети получаетдейтограммы)<sup>4</sup>. Предполагается, что узел сети ведет таблицу, где отмечает свою принадлежность к той или иной группе. В исходном состоянии все элементы таблицы пусты. Как только прикладная программа, запущенная на узле сети присоединяется к новой группе, модуль протокола IGMP создает в таблице новый элемент и заполняет его информацией об этой группе. Помимо других данных, в таблице указывается счетчик обращений к группе,

<sup>3</sup> В протоколе IGMP предусмотрен специальный тип сообщения, с помощью которого маршрутизатор при необходимости может опросить конкретную группу.

<sup>4</sup> Группа, в которую входят все компьютеры данной сети (ее адрес 224.0.0.1) является исключением, поскольку ни один узел сети никогда не сообщает о принадлежности к ней.

значение которого изначально устанавливается в единицу. Значение счетчика увеличивается на единицу, каждый раз, когда другая прикладная программа присоединяется к той же группе. При завершении работы программы или при поступлении явного запроса на отключение от группы значение счетчика уменьшается на единицу. Если значение счетчика становится равным нулю, узел сети сообщает многоадресатным маршрутизаторам о том, что он отключается от многоадресатной группы.

Действия, предпринимаемые программой поддержки протокола IGMP в ответ на разные события, происходящие в сети, можно схематично изобразить с помощью диаграммы перехода состояний, показанной на рис. 17.2.

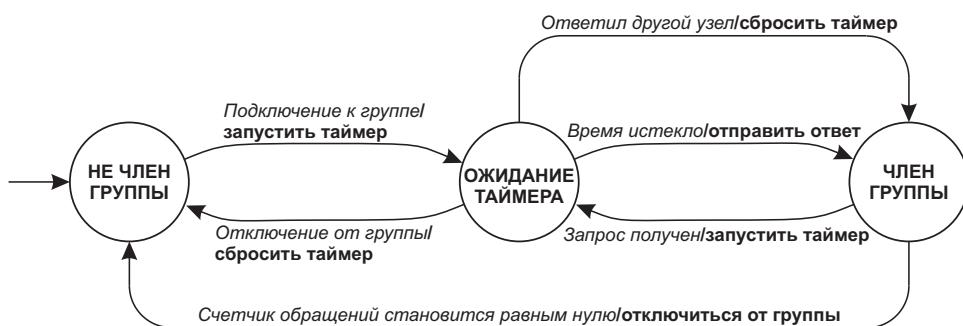


Рис. 17.2. Три возможных состояния элемента таблицы многоадресатных групп узла сети и переходы между ними. Каждый переход обозначен с указанием события и действия. На переходах из одного состояния в другое не показаны сообщения, посылаемые в момент присоединения к группе или отключения от нее

Для каждой группы, членом которой в текущий момент является узел, в таблице создается отдельный элемент. Как показано на рис. 17.2, когда узел сети первым присоединяется к группе или когда поступает запрос от многоадресатного маршрутизатора, элемент таблицы переводится в состояние *ожидания таймера* и выбирается случайное время задержки. Если другой узел этой группы ответит на запрос маршрутизатора прежде, чем истечет время таймера, узел сети сбрасывает значение своего таймера и переходит в состояние *члена группы*. Если же время таймера истекло, узел сети отсылает ответное сообщение и переходит в состояние *члена группы*. Поскольку маршрутизатор посылает запрос только каждые 125 секунд, предполагается, что большую часть времени сетевой узел продолжает оставаться в состоянии *члена группы*.

В изображенной на рис. 17.2 диаграмме опущено несколько деталей. Например, если запрос поступает в то время, когда узел сети находится в состоянии *ожидания таймера*, то, согласно протоколу, он должен переустановить значение таймера. Более того, для обеспечения обратной совместимости с протоколом IGMPv1 в версии 2 протокола предусмотрена также обработка сообщений версии 1. Это позволяет одновременно использовать как протокол IGMPv1, так и протокол IGMPv2 в одной и той же сети.

## 17.16. Формат IGMP-сообщений

Как показано на рис. 17.3, IGMP-сообщения, которыми обмениваются узлы сети, имеют простой формат.

Каждое IGMP-сообщение состоит из восьми октетов. Тип сообщения определяется в первом октете, значения которого приведены в табл. 17.3. Как уже упоминалось, для обеспечения обратной совместимости поддерживаются также



Рис. 17.3. Формат 8-октетного IGMP-сообщения, используемого для обмена информацией между узлами сети и маршрутизаторами

IGMP-сообщения версии 1. Когда маршрутизатор посылает запрос на определение членов группы, в поле под названием *время ответа* указывается максимальный интервал времени в десятых долях секунды, в течение которого он будет ожидать ответ. Каждый член группы случайным образом вычисляет собственное время задержки в диапазоне от нуля до указанного значения этого поля. Ответ на запрос посыпается только после истечения выбранного времени задержки. Как уже было сказано, стандартное время ожидания составляет 10 секунд. Это означает, что все узлы сети, входящие в группу, будут выбирать случайным образом значение задержки в диапазоне от 0 до 10 секунд. В протоколе IGMP маршрутизаторам разрешено устанавливать максимальное время ответа на запрос. Это сделано для того, чтобы можно было управлять трафиком протокола IGMP. Если к сети подключено много узлов, то повышение максимального времени задержки увеличивает время ответа, а значит, снижается вероятность того, что на запрос ответит сразу несколько узлов сети. В поле *контрольной суммы* содержится значение контрольной суммы сообщения. В протоколе IGMP контрольная сумма вычисляется только для сообщения. При этом используется такой же алгоритм, как и в протоколах TCP и IP. Поле *адрес группы* используется либо для указания определенной группы, либо содержит значение, равное нулю, если запрос относится ко всем группам сразу. Отсылая запрос определенной группе, маршрутизатор заполняет поле адреса группы, а узлы сети заполняют это поле при отправлении отчетов о членах группы.

Таблица 17.3. Типы сообщений протокола IGMP версии 2

Тип	Адрес группы	Описание
0x11	Отсутствует (ноль)	Запрос о членах всех групп
0x11	Используется	Запрос о членах определенной группы
0x16	Используется	Отчет о членах группы
0x17	Используется	Отключение от группы
0x12	Используется	Отчет о членах группы (версия 1)

Обратите внимание, что в протоколе IGMP не предусмотрен механизм, который позволяет узлу сети определить IP-адрес группы. Поэтому, прежде чем отправить IGMP-сообщение для присоединения к группе, прикладная программа должна каким-то образом узнать адрес группы. Эта проблема решается несколькими способами. В некоторых приложениях используются постоянно назначенные адреса, в других — адрес группы задается вручную сетевым администратором при установке программного обеспечения, в третьих — адрес группы определяется динамически (например, после отправки специального запроса серверу). Как бы то ни было, в протоколе IGMP нет средств для поиска адресов групп.

## 17.17. Пересылка многоадресатных пакетов и многоадресатная маршрутизация

Протокол IGMP и описанная выше структура адресации групп определяют, как узлы сети взаимодействуют с локальным маршрутизатором и каким образом многоадресатные дейтаграммы передаются по одной сети. Однако они не уточняют, каким образом маршрутизаторы должны обмениваться информацией о групповом членстве и обеспечивать доставку копии каждой из дейтаграмм всем членам группы. Обратите внимание, что, хотя разработчики предложили несколько протоколов, ни один из стандартов не был специально разработан для распространения многоадресатной маршрутной информации. Несмотря на все усилия, разработчики так и не пришли к согласию касательно общего проекта такого стандарта. Поэтому существующие протоколы отличаются выполняемыми функциями и подходом к многоадресатной передаче.

Почему же многоадресатная маршрутизация представляет собой такую трудную задачу? Почему бы не расширить обычную систему маршрутизации, чтобы она могла работать в режиме многоадресатной передачи? Проблема в том, что многоадресатная маршрутизация существенно отличается от обычной маршрутизации, поскольку многоадресатная пересылка пакетов отличается от обычной. Чтобы разобраться в некоторых отличиях между ними, рассмотрим многоадресатную пересылку пакетов по сети, структурная схема которой показана на рис. 17.4.

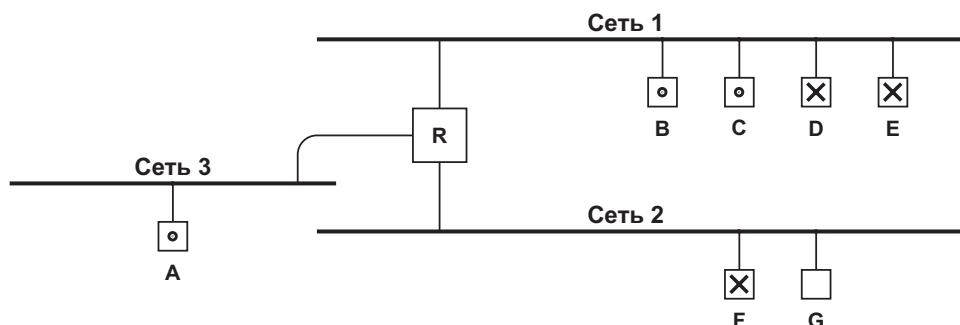


Рис. 17.4. Объединенная сеть, состоящая из трех локальных сетей, соединенных одним маршрутизатором, на примере которой проиллюстрирован процесс многоадресатной пересылки пакетов. Узлы сети, обозначенные на рисунке точками, входят в одну многоадресатную группу, а обозначенные крестиками — в другую

### 17.17.1. Необходимость в динамической маршрутизации

Даже для сети простой топологии, изображенной на рис. 17.4, процесс многоадресатной пересылки пакетов существенно отличается от одноадресатной. Например, на рис. 17.4 показаны две многоадресатные группы. Группа, обозначенная точками, состоит из членов A, B и C, а группа, обозначенная крестиками, состоит из членов D, E и F. Группа, обозначенная точками, не имеет членов в сети 2. Чтобы не создавать лишнего трафика, маршрутизатор не должен пересыпать пакеты, предназначенные для обозначенной точками группы, в сеть 2. Однако произвольный узел сети в любой момент может присоединиться к одной из групп. Если некоторый узел первым из своей сети присоединяется к определенной группе, в процесс многоадресатной маршрутизации необходимо внести изменения и включить в нее эту сеть. Таким образом, можно сделать вывод о важном

отличии между стандартной маршрутизацией и многоадресатной маршрутизацией, который состоит в следующем.

*В отличие от одноадресатной маршрутизации, когда маршруты изменяются только при изменении топологии сети или при отказе оборудования, многоадресатные маршруты могут изменяться всего лишь вследствие присоединения узла сети к многоадресатной группе или выхода из нее.*

### 17.17.2. Недостаточность информации об адресе получателя

На примере сети, изображенной на рис. 17.4, показан другой аспект многоадресатной маршрутизации. Если оба сетевых узла  $F$  и  $E$  отсылают дейтаграммы для обозначенной крестиками группы, маршрутизатор  $R$  получит и перешлет эти дейтаграммы в соответствующие сети. Поскольку обе дейтаграммы направлены одной группе, они имеют одинаковый адрес получателя. Однако для корректной пересылки каждой из дейтаграмм маршрутизатору нужно выполнить два разных действия. Дейтаграмму, посланную от узла  $E$ , необходимо передать в сеть 2, а от узла  $F$  — в сеть 1. Интересно, что, получив отправленную узлом  $A$  дейтаграмму, предназначенную для обозначенной крестиками группы, маршрутизатор должен выполнить третье действие — переслать две копии дейтаграммы, одна из которых предназначена для сети 1, а другая — для сети 2. Таким образом, становится очевидным второе существенное отличие между пересылкой обычных и многоадресатных пакетов.

*Для многоадресатной пересылки пакетов маршрутизатору недостаточно проанализировать только адрес получателя.*

### 17.17.3. Произвольные отправители

Еще одна (последняя) особенность многоадресатной маршрутизации, изображенная на рис. 17.4, заключается в том, что в протоколе IP узлу сети, который может не быть членом группы, разрешено отправлять дейтаграммы этой группе. Например, узел  $G$  (см. рис. 17.4) может послать дейтаграмму обозначенной точками группе несмотря на то, что этот узел не является членом ни одной из групп и к его сети не подключен ни один из членов обозначенной точками группы. Обратите внимание, что при передаче по объединенной сети дейтаграмма может пройти через сети, к которым не подключен ни один из членов данной группы. Таким образом, можно сделать следующий вывод.

*Многоадресатная дейтаграмма может быть послана с компьютера, не являющегося членом многоадресатной группы. При этом она может передаваться по сетям, к которым не подключен ни один из членов данной группы.*

## 17.18. Основные принципы многоадресатной маршрутизации

Из приведенного выше примера становится ясно, что для пересылки многоадресатных дейтаграмм маршрутизаторам недостаточно адреса получателя. Так какую же информацию использует многоадресатный маршрутизатор, принимая решение о пересылке дейтаграммы? Чтобы ответить на этот вопрос следует учитывать один важный момент: поскольку многоадресатный получатель представляет собой совокупность компьютеров, система оптимальной пересылки дейтаграмм должна доставить дейтаграмму каждому из них, не отсылая ее по одной и

той же сети дважды. В случае объединенной сети, содержащей только один многоадресатный маршрутизатор (см. рис. 17.4), эта проблема решается довольно просто: маршрутизатор не должен отсылать дейтаграмму обратно по тому интерфейсу, через который она была получена. Однако этот принцип работает далеко не всегда, например в случае, когда несколько маршрутизаторов закольцована между собой. Поэтому, чтобы предотвратить образование маршрутных петель, многоадресатные маршрутизаторы анализируют адрес отправителя дейтаграммы.

Одна из первых идей, связанных с многоадресатной рассылкой пакетов, заключалась в том, что ее следует рассматривать с точки зрения широковещательной передачи, описанной в главе 10, “Бесклассовая адресация и подсети (CIDR)”. Метод называли *пересылкой по обратному маршруту* (*Reverse Path Forwarding*, или *RFT*)<sup>5</sup>. Чтобы предотвратить повторную передачу дейтаграмм и образование маршрутных петель, при их пересылке маршрутизатор анализирует адрес отправителя дейтаграммы. Для использования метода RFT многоадресатный маршрутизатор должен иметь обычную таблицу маршрутизации, в которой указаны кратчайшие маршруты ко всем получателям. Получив дейтаграмму, маршрутизатор извлекает из нее адрес отправителя. Затем по этому адресу путем поиска в локальной таблице маршрутизации определяется интерфейс *I*, по которому дейтаграммы должны были бы передаваться отправителю (т.е. кратчайший путь к отправителю). И если окажется, что дейтаграмма поступила от отправителя через интерфейс *I*, маршрутизатор пересыпает по копии этой дейтаграммы всем остальным интерфейсам. В противном случае копии не рассылаются.

Поскольку базовый метод RFT обеспечивает пересылку копии каждой многоадресатной дейтаграммы во все сети, составляющие объединенную сеть, он также гарантирует, что каждый член многоадресатной группы получит по копии каждой отправленной этой группе дейтаграммы. Однако в чистом виде метод RFT не используется для многоадресатной маршрутизации, поскольку при этом создается дополнительная, часто ненужная, нагрузка на сеть. Все дело в том, что в методе RFT дейтаграммы пересыпаются также и в те сети, к которым не подключены члены группы и которые к этим членам не ведут.

Чтобы предотвратить распространение многоадресатных дейтаграмм там, где в них нет необходимости, была создана модифицированная форма алгоритма RFT. Этот метод получил название *усеченной пересылки по обратному маршруту* (*Truncated Reverse Path Forwarding*, или *TRPF*), или *усеченного широковещания по обратному маршруту* (*Truncated Reverse Path Broadcasting*, или *TRPB*). Суть его состоит в том, что в базовом алгоритме RFT ограничивается распространение многоадресатных дейтаграмм по тем маршрутам, которые не ведут к членам группы. Для использования алгоритма TRPF многоадресатному маршрутизатору необходимо располагать двумя источниками информации: обычной таблицей маршрутизации и списком многоадресатных групп, доступ к которым можно получить через каждый интерфейс маршрутизатора. После получения многоадресатной дейтаграммы, маршрутизатор сначала применяет алгоритм RFT. Если оказывается, что дейтаграмму никуда отсылать не нужно, маршрутизатор ее аннулирует. Однако если окажется, что дейтаграмму нужно передать по определенному интерфейсу, маршрутизатор выполняет дополнительную проверку по адресу получателя дейтаграммы. Он должен убедиться, что через этот интерфейс проложен маршрут хотя бы к одному члену группы. Если ни одного из членов группы нельзя достичь через данный интерфейс, маршрутизатор не посыпает через него дейтаграмму и переходит к проверке следующего

<sup>5</sup> Пересылку дейтаграмм по обратному маршруту иногда называют *широковещанием по обратному маршруту* (*Reverse Path Broadcasting*, или *RPB*).

интерфейса. Теперь становится понятным происхождение слова *усеченный* в названии алгоритма. Маршрутизатор завершает процесс пересылки дейтаграмм, когда ни один из маршрутов, проложенных через оставшиеся интерфейсы, не ведет к членам группы. Таким образом, можно сделать следующий вывод.

*Принимая решение о пересылке дейтаграмм по конкретному интерфейсу, многоадресатный маршрутизатор анализирует как адрес отправителя, так и получателя. Основной метод пересылки дейтаграмм называется алгоритмом усеченной пересылки по обратному маршруту (TRPF).*

### 17.19. Последствия использования TRPF

Хотя метод TRPF гарантирует, что все члены многоадресатной группы получат копию каждой отосланной этой группе дейтаграммы, его использование имеет два неожиданных последствия. Во-первых, поскольку в алгоритме TRPF применяется такой же способ предотвращения образования маршрутных петель, как и в RPF, в некоторые сети будут пересланы “лишние” копии дейтаграмм. Процесс возникновения дубликатов проиллюстрирован на рис. 17.5.

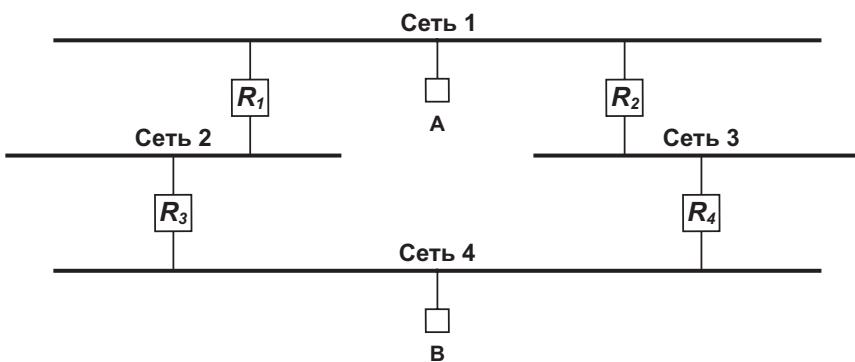


Рис. 17.5. Топологическая схема объединенной сети, приводящая к тому, что при использовании метода RPF некоторым получателям будет послано по несколько копий дейтаграммы

Как следует из рис. 17.5, маршрутизаторы R<sub>1</sub> и R<sub>2</sub> получат по копии дейтаграммы, отправленной узлом A. Поскольку дейтаграмма поступит на маршрутизаторы по интерфейсу, через который проходит кратчайший маршрут к узлу A, маршрутизатор R<sub>1</sub> перешлет копию дейтаграммы в сеть 2, а маршрутизатор R<sub>2</sub> — в сеть 3. Получив копию дейтаграммы через интерфейс, подключенный к сети 2 (он также расположен на кратчайшем маршруте к узлу A), маршрутизатор R<sub>3</sub> перешлет эту копию в сеть 4. К сожалению, маршрутизатор R<sub>4</sub> также перешлет копию дейтаграммы в сеть 4. Таким образом, несмотря на то, что метод RPF позволяет маршрутизаторам R<sub>3</sub> и R<sub>4</sub> предотвратить образование маршрутной петли, аннулировав поступающую по сети 4 копию дейтаграммы, узел B получит две копии этой дейтаграммы.

Во-вторых, еще одно неожиданное последствие использования метода TRPF возникает в результате того, что при пересылке дейтаграмм учитываются адреса отправителя и получателя. При этом способ доставки зависит от адреса отправителя дейтаграммы. Например, на рис. 17.6 показано, каким образом многоадресатные маршрутизаторы пересыпают дейтаграммы, отправленные двумя разными отправителями по объединенной сети с фиксированной топологией.

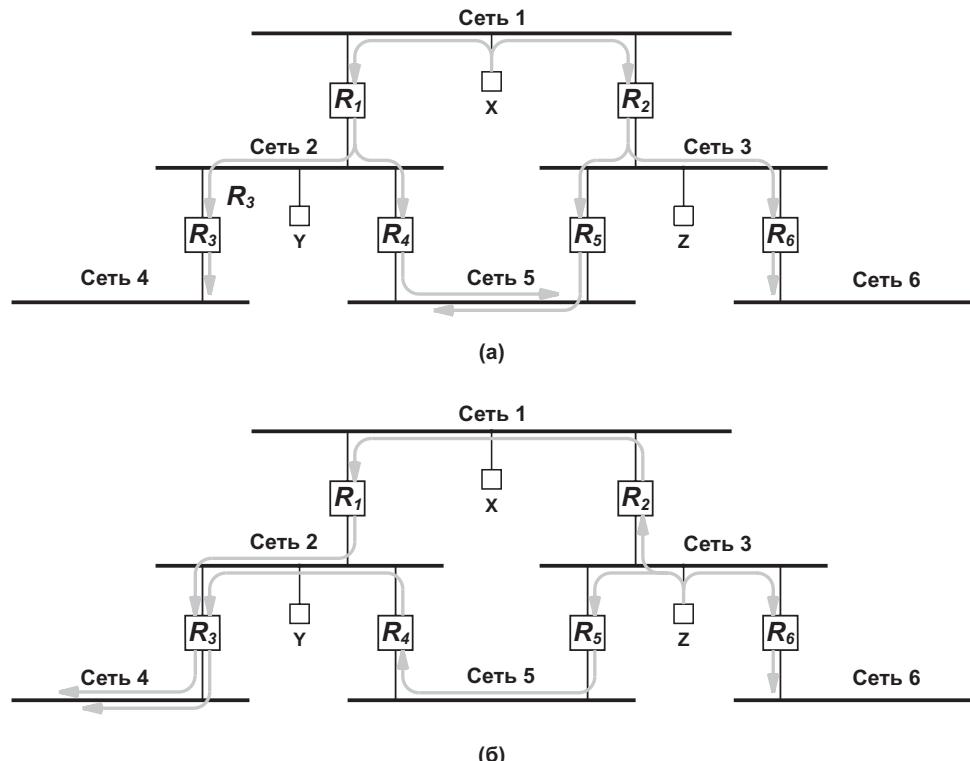


Рис. 17.6. Маршруты, по которым следует многоадресатнаядейтаграмма при пересыпке по методу TRPF. Предполагается, что отправителями являются узлы X (а) и Z (б), а в каждой из сетей расположены члены группы. При этом количество полученных корпий зависит от отправителя дейтаграммы.

Как следует из рис. 17.6, положение отправителя в объединенной сети оказывает влияние как на маршрут, по которому следует дейтаграмма, чтобы достичь конкретной сети, так и на особенности доставки. Например, на рис. 17.6(а) передача дейтаграммы узлом  $X$  приводит к тому, что в результате применения алгоритма TRPF в сеть 5 будет доставлено две копии дейтаграммы. На рис. 17.6(б) только одна копия отправленной узлом  $Z$  дейтаграммы достигает сети 5, а две копии попадают в сети 2 и 4.

## 17.20. Дерево многоадресатной передачи

Для описания совокупности маршрутов, ведущих от данного отправителя ко всем членам многоадресатной группы, используется терминология, взятая из теории графов. Принято говорить, что совокупность маршрутов определяет дерево<sup>6</sup>, которое иногда называют деревом пересылки (*forwarding tree*) или деревом доставки (*delivery tree*). Каждый многоадресатный маршрутизатор соответствует узлу графа (дерева), а сеть, соединяющая два маршрутизатора, соответствует ребру графа (дерева). Отправитель дейтаграммы называется корнем (*root*) или корневым узлом (*root node*) дерева. И, наконец, последний маршрутизатор,

<sup>6</sup> Граф является деревом, если он не содержит петель (т.е. на одном пути может быть расположено только один маршрутизатор).

расположенный на каждом из ведущих от отправителя дейтаграммы маршрутов, называется *краевым* (*leaf*) маршрутизатором. Иногда эта терминология применяется также к сетям. Так, исследователи называют сеть, подключенную к краевому маршрутизатору, *краевой сетью*.

Рассмотрим пример использования приведенной выше терминологии. На рис. 17.6(а) изображено дерево с корневым узлом  $X$  и краевыми маршрутизаторами  $R_3$ ,  $R_4$ ,  $R_5$  и  $R_6$ . Формально нельзя считать рис. 17.6(б) деревом, поскольку маршрутизатор  $R_3$  расположен сразу на двух маршрутах. Зачастую, не соблюдая формальностей, исследовали не обращают внимание на детали и воспринимают такие графы в качестве деревьев. Используя терминологию теории графов, можно сформулировать следующий важный принцип.

*Дерево многоадресатной пересылки пакетов определяется как совокупность маршрутов, ведущих через многоадресатные маршрутизаторы от отправителя ко всем членам многоадресатной группы. Для одной многоадресатной группы каждый возможный отправитель дейтаграмм может определить разное дерево пересылки.*

Из этого принципа сразу вытекает следствие, касающееся размера таблиц, которые используются при пересылке многоадресатных пакетов. В отличие от стандартных таблиц маршрутизации, каждый элемент таблицы многоадресатного маршрутизатора содержит следующую пару адресов:

(адрес группы, адрес отправителя)

Теоретически под *отправителем* подразумевается узел сети, который может отсылать дейтаграммы группе (т.е. любой узел в объединенной сети). На практике нежелательно создавать в таблице элемент для каждого узла сети, поскольку деревья пересылки, определяемые для всех узлов одной сети, идентичны. Следовательно, для экономии места в протоколе маршрутизации в качестве *отправителя* используется префикс IP-адреса сети. Это означает, что в таблице пересылки многоадресатного маршрутизатора нужно определить один элемент для всех узлов, расположенных в одной физической сети.

Группировка элементов таблицы по префиксу IP-адреса сети, а не по IP-адресу узла, позволяет существенно уменьшить размер таблицы. Дело в том, что размер таблицы многоадресатной маршрутизации может значительно превышать размер обычной таблицы маршрутизации. Размер обычной таблицы маршрутизации пропорционален общему количеству сетей, составляющих объединенную сеть. Размер же таблицы многоадресатной маршрутизации пропорционален произведению количества сетей в объединенной сети и количества многоадресатных групп.

## 17.21. Противоречия в системе многоадресатной маршрутизации

Вдумчивый читатель мог заметить противоречие между возможностями многоадресатной передачи протокола IP и алгоритмом TRPF. Уже упоминалось, что алгоритм TRPF используется вместо обычного алгоритма RPF для уменьшения ненужного трафика. В алгоритме TRPF дейтаграмма не пересыпается в сеть, если по этой сети не проходит маршрут, ведущий по меньшей мере к одному члену группы. Следовательно, многоадресатный маршрутизатор должен располагать информацией о членах группы. Также было сказано, что протокол IP позволяет любому узлу сети в любой момент присоединиться к многоадресатной группе или отключиться от нее, а это приводит к частым изменениям состава группы. И, что более важно, члены группы не обязательно

должны находиться в непосредственной близости друг от друга. Узел, который присоединяется к группе, может располагаться на большом удалении от маршрутизатора, пересылающего дейтаграммы этой группе. Поэтому информация о членах группы должна распространяться по объединенной сети.

Понятие членства группы является ключевым в процессе маршрутизации. Все системы многоадресатной маршрутизации должны обеспечивать механизм для распространения информации о членах группы, а также позволять использовать эту информацию при пересылке дейтаграмм. Поскольку состав группы может быстро изменяться, доступную на данном маршрутизаторе информацию нельзя считать полной. Поэтому при маршрутизации сделанные изменения могут быть не учтены. Следовательно, структура системы многоадресатной пересылки должна представлять собой компромиссное решение, учитывающее чрезмерный объем трафика и неэффективную передачу данных. С одной стороны, если информация о членах группы не будет быстро распространяться по сети, многоадресатные маршрутизаторы не смогут принять оптимального решения (т.е. они либо перешлют дейтаграмму в некоторые сети без необходимости, либо им не удастся отослать дейтаграмму всем членам группы). С другой стороны, система многоадресатной маршрутизации, в которой о каждом изменении в составе группы немедленно сообщается всем маршрутизаторам, обречена на провал, поскольку получаемый в результате этого трафик может вызвать перегрузку в объединенной сети. Таким образом, структура любой системы многоадресатной маршрутизации представляет собой компромиссное решение между двумя экстремальными альтернативами.

## 17.22. Многоадресатная передача по обратному маршруту

В основу работы одной из первых форм многоадресатной маршрутизации был положен алгоритм TRPF. Ее назвали *многоадресатной передачей по обратному маршруту* (Reverse Path Multicast, или RPM). Для придания алгоритму TRPF большей динамичности он был немного расширен. При создании этой системы маршрутизации было сделано три предпосылки. Во-первых, намного важнее обеспечить доставку многоадресатной дейтаграммы каждому члену группы, для которой она предназначена, чем предотвратить ненужную передачу дейтаграмм. Во-вторых, каждый из многоадресатных маршрутизаторов содержит стандартную таблицу маршрутизации, в которой находится правильная маршрутная информация. В-третьих, многоадресатная маршрутизация должна повысить эффективность доставки многоадресатных пакетов там, где это возможно (т.е. предотвратить ненужную передачу).

Маршрутизация по методу RPM происходит в два этапа. В начале работы алгоритм RPM рассыпает копии многоадресатных дейтаграмм по всем сетям объединенной сети, используя широковещательный режим передачи и RPF. Таким образом гарантируется, что все члены группы получат по копии дейтаграмм. Одновременно алгоритм RPM заставляет многоадресатные маршрутизаторы сообщить друг другу о маршрутах, которые не ведут к членам этой группы. Как только маршрутизатор получает информацию о том, что на данном маршруте нет членов группы, он прекращает пересылку дейтаграмм по этому маршруту.

Каким образом маршрутизаторы узнают о местонахождении членов группы? Как и в большинстве систем многоадресатной маршрутизации, в алгоритме RPM информация о групповом членстве распространяется снизу вверх. Распространение информации начинается с узлов сети, которые намереваются присоединиться к группам или отключиться от них. Узлы сети сообщают информацию о членах группы своему локальному маршрутизатору с помощью протокола IGMP. Поэтому, хотя многоадресатному маршрутизатору не известно об удаленных членах

группы, он все же располагает информацией о ее локальных членах (т.е. членах, расположенных во всех непосредственно подключенных к нему сетях). Следовательно, маршрутизаторы, подключенные к краевым сетям, могут сами решать, пересылать ли им дейтаграммы по краевой сети. Если в краевой сети нет членов группы, маршрутизатор, через который подключена эта сеть к объединенной сети, не пересыпает по ней пакеты. Кроме выполнения своих обычных локальных функций, краевой маршрутизатор передает групповую информацию следующему маршрутизатору, расположенному по маршруту, ведущему обратно к отправителю. Как только следующий маршрутизатор узнает, что за пределами данного сетевого интерфейса нет членов группы, он прекращает пересылку через него предназначенных для этой группы дейтаграмм. Если маршрутизатор обнаруживает, что в подключенных к нему сетях нет членов группы, он сообщает об этом следующему маршрутизатору, расположенному по маршруту, ведущему к корню.

Используя терминологию из теории графов, можно сказать, что когда маршрутизатор узнает о том, что по данному маршруту не расположены члены группы, и прекращает пересылку дейтаграмм, он *отсекает* (т.е. удаляет) этот маршрут из дерева пересылки. По сути в алгоритме RPM используется стратегия *широковещания и отсечения* (*broadcast and prune*), поскольку маршрутизатор широковещательно рассыпает дейтаграммы по данному маршруту (используя алгоритм RPF) до тех пор, пока он не получит информацию, которая позволит ему отсечь этот маршрут. Для обозначения алгоритма RPM исследователи также используют другой термин — система, управляемая данными (*data-driven system*). Все дело в том, что маршрутизатор не отсылает информацию о членах группы другим маршрутизаторам до тех пор, пока не получит предназначенные для этой группы дейтаграммы.

В системе, управляемой данными, маршрутизатор также должен обрабатывать ситуацию, когда узел сети “захочет” присоединиться к определенной группе после того, как маршрутизатор отсек ведущий к ней маршрут. В алгоритме RPM подобные ситуации обрабатываются по принципу снизу вверх. Когда узел сообщает локальному маршрутизатору, что он присоединился к группе, маршрутизатор обращается к своим данным об этой группе, и находит в них адрес маршрутизатора, к которому он ранее отоспал запрос на отсечение. Затем маршрутизатор посыпает новое сообщение, которое аннулирует результаты предыдущего отсечения, и передача дейтаграмм возобновляется. Такие сообщения называют *запросами на подключение ветви* (*graft requests*). Говорят, что алгоритм подсоединяет отсеченную ветвь обратно к дереву.

### 17.23. Дистанционно-векторный протокол многоадресатной маршрутизации

В глобальной сети Internet все еще используется один из первых протоколов многоадресатной маршрутизации. Этот протокол, называемый *дистанционно-векторным протоколом многоадресатной маршрутизации* (*DVMRP*), позволяет многоадресатным маршрутизаторам обмениваться информацией о членах группы и маршрутной информацией. Протокол DVMRP напоминает протокол RIP, описанный в главе 16, “Маршрутизация внутри автономной системы (RIP, OSPF, HELLO)”. Он является расширенной версией протокола RIP, обеспечивающей режим многоадресатной передачи. По сути этот протокол распространяет информацию о текущих членах группы, а также о стоимости передачи дейтаграмм между маршрутизаторами. Для каждой возможной пары (*группа, отправитель*) маршрутизатор строит дерево пересылок на основе дерева физических подключений. Получив предназначенную для многоадресатной IP-группы дейтаграмму,

он отсылает копию этой дейтаграммы по сетевым соединениям, которые соответствуют ветвям дерева пересылки<sup>7</sup>.

Интересно, что в спецификации протокола DVMRP определена расширенная форма протокола IGMP, которая используется для обмена информацией между двумя многоадресатными маршрутизаторами. В ней указаны дополнительные типы IGMP-сообщений, которые позволяют маршрутизаторам объявить о вступлении в многоадресатную группу, отключении от нее, а также опросить другие маршрутизаторы. Расширения протокола IGMP также позволяют создавать сообщения, в которых пересылается маршрутная информация, включая метрику стоимости маршрутов.

## 17.24. Программа mrouted

Программа `mrouted` является популярным демоном, реализующим протокол DVMRP для систем UNIX. Подобно `routed`<sup>8</sup>, программа `mrouted` тесно взаимодействует с ядром операционной системы для сбора многоадресатной маршрутной информации. Однако в отличие от `routed`, программа `mrouted` не использует обычную таблицу маршрутизации. Поэтому ее можно использовать только со специальной версией ядра системы UNIX, которое называется *многоадресатным (multicast kernel)*, поддерживающим многоадресатную пересылку дейтаграмм. В этом ядре системы UNIX содержится специальная таблица многоадресатной маршрутизации, а также программный код, который необходим для пересылки многоадресатных дейтаграмм. Программа `mrouted` выполняет следующие функции.

- *Распространение маршрутов.* В программе `mrouted` для распространения многоадресатной маршрутной информации от одного маршрутизатора к другому используется протокол DVMRP. Компьютер, на котором запущена программа `mrouted`, интерпретирует многоадресатную маршрутную информацию и создает таблицу многоадресатной маршрутизации. Как и следовало ожидать, в каждом элементе этой таблицы указана пара адресов (*группа, отправитель*), а также набор интерфейсов, по которым будут пересылаться соответствующие этому элементу дейтаграммы. Программа `mrouted` не заменяет собой другие стандартные программы поддержки протоколов маршрутизации. Как правило, эта программа запускается на компьютере в качестве дополнения к программному обеспечению поддержки стандартного протокола маршрутизации.
- *Многоадресатное туннелирование.* Одна из главных проблем, связанных с режимом многоадресатной передачи в объединенной сети, возникает из-за того, что не все маршрутизаторы могут пересылать многоадресатные дейтаграммы. Поэтому в программе `mrouted` предусмотрена возможность пересылки многоадресатных дейтаграмм по туннелю между маршрутизаторами, проходящему через промежуточные маршрутизаторы, которые не участвуют в многоадресатной маршрутизации.

Хотя программа `mrouted` может одна справиться с описанными выше задачами, на конкретном компьютере обе эти функции могут и не понадобиться. Поэтому, чтобы администратор мог точно определить функции программы, в программе `mrouted` предусмотрено использование файла конфигурации. В этом файле содержатся данные, которые определяют, какие именно многоадресатные

<sup>7</sup> Версии 2 и 3 протокола DVMRP существенно отличаются. В последнюю из них включен описанный выше алгоритм RPM.

<sup>8</sup> Напомним, что `routed` является программой для UNIX, которая реализует протокол RIP.

группы программа `mrouted` имеет право анонсировать через каждый интерфейс, а также способ пересылки дейтаграмм. Кроме того, в файле конфигурации для каждого из маршрутизаторов назначается определенная метрика и пороговое значение. Метрика позволяет администратору присвоить каждому маршруту определенную стоимость (например, у маршрута, проходящего по локальной сети, должна быть меньшая стоимость, чем у маршрута, проложенного по низкоскоростной линии последовательной передачи). Пороговое значение представляет собой минимальное значение *времени жизни (TTL)* IP-дейтаграммы, которое необходимо ей для прохождения маршрута. Если время жизни дейтаграммы меньше указанного порогового значения, многоадресатное ядро не пересыпает дейтаграмму получателю, аннулируя ее. В результате пропускная способность сети не растративается понапрасну.

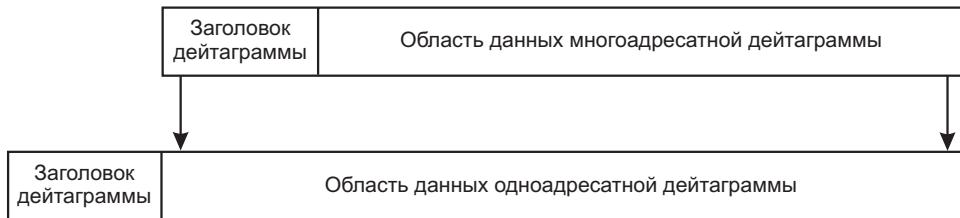
Многоадресатное туннелирование, вероятно, — самое интересное свойство программы `mrouted`. Необходимость в туннеле возникает, когда несколько узлов сети хотят принять участие в многоадресатной передаче, но один или несколько маршрутизаторов, расположенных по лежащему между ними маршруту, не поддерживают многоадресатной маршрутизации. Этот случай проиллюстрирован на рис. 17.7.



Рис. 17.7. Пример конфигурации объединенной сети, в которой необходимо выполнять туннелирование для того, чтобы компьютеры, подключенные к сетям 1 и 2, могли участвовать в обмене многоадресатными дейтаграммами. Маршрутизаторы объединенной сети, которая расположена между сетями 1 и 2, не распространяют многоадресатные маршруты и не могут пересыпать дейтаграммы, отосланые по адресу группы

Чтобы подключенные к сетям 1 и 2 узлы могли обмениваться многоадресатными дейтаграммами, администраторы маршрутизаторов  $R_1$  и  $R_2$  должны сконфигурировать туннель между двумя программами `mrouted`. Такой туннель возникает только тогда, когда программы `mrouted`, запущенные на обоих маршрутизаторах, “договорятся” о взаимном обмене дейтаграммами. Каждый маршрутизатор “прослушивает” свою локальную сеть с целью обнаружения дейтаграмм, посланных по адресу группы, для которой и был сконфигурирован туннель. Когда поступает многоадресатная дейтаграмма, адрес получателя которой соответствует одному из сконфигурированных туннелей, программа `mrouted` инкапсулирует эту дейтаграмму в обычную одноадресатную дейтаграмму и отсылает ее по объединенной сети маршрутизатору, расположенному на другом конце туннеля. Получив одноадресатную дейтаграмму по одному из своих туннелей, программа `mrouted` извлекает из нее многоадресатную дейтаграмму и пересыпает ее в соответствии со своей таблицей многоадресатной маршрутизации.

Метод инкапсуляции, используемый программой `mrouted` для отсылки дейтаграмм по туннелю, называется *IP-в-IP (IP-in-IP)*. Он проиллюстрирован на рис. 17.8.



*Рис. 17.8. Инкапсуляции IP-в-IP: одна дейтаграмма помещается в область данных другой дейтаграммы. Пара многоадресатных маршрутизаторов использует этот тип инкапсуляции для обмена информацией в том случае, если промежуточные маршрутизаторы не поддерживают многоадресатную передачу данных*

Как показано на рис. 17.8, при инкапсуляции типа IP-в-IP исходная многоадресатная дейтаграмма вместе с ее заголовком помещается в область данных обычной одноадресной дейтаграммы. На принимающей дейтаграмму машине многоадресатное ядро извлекает и обрабатывает многоадресатную дейтаграмму так, как будто она поступила через один из локальных интерфейсов. В частности, после извлечения многоадресатной дейтаграммы принимающая машина сначала должна уменьшить на единицу значение поля времени жизни, находящееся в ее заголовке, и только затем пересыпать дейтаграмму. Следовательно, после создания туннеля программа `mrouted` воспринимает объединенную сеть, соединяющую два многоадресатных маршрутизатора, как единую физическую сеть. Обратите внимание, что исходная одноадресная дейтаграмма имеет собственный счетчик времени жизни, значение которого никак не связано со счетчиком времени жизни, расположенным в заголовке многоадресатной дейтаграммы. Поэтому можно ограничить количество физических переходов одноадресатной дейтаграммы по данному туннелю. Причем это число не зависит от количества логических переходов через маршрутизаторы, которые должна посетить многоадресатная дейтаграмма на своем пути от отправителя до конечного получателя.

Из многоадресатных туннелей сформирована основа магистрали многоадресатной передачи (*Multicast Backbone* или *MBONE*) глобальной сети Internet. В работе магистрали MBONE принимают участие многие сетевые центры. Она позволяет узлам, размещенным в задействованных сетевых центрах, отсылать и получать многоадресатные дейтаграммы, которые затем пересыпаются всем остальным задействованным центрами сети Internet. Магистраль MBONE часто используется для распространения аудио- и видеофайлов (например, при проведении телеконференций).

Чтобы принять участие в работе магистрали MBONE, сетевой центр должен иметь по меньшей мере одну локальную сеть, подключенную к Internet через многоадресатный маршрутизатор. Кроме того, необходимо получить разрешение от администрации любого другого сетевого центра на пересылку многоадресатного трафика по туннелю и сконфигурировать, собственно, сам туннель между двумя маршрутизаторами разных центров. Когда произвольный узел, расположенный в одном из сетевых центров, отсылает многоадресатную дейтаграмму, ее копия доставляется локальному маршрутизатору этого сетевого центра. Маршрутизатор сверяет адреса отправителя и получателя со своей таблицей многоадресатной маршрутизации и направляет дейтаграмму по туннелю, используя метод IP-в-IP. Получив дейтаграмму по многоадресатному туннелю, многоадресатный маршрутизатор, расположенный в другом сетевом центре, извлекает из нее внутреннюю дейтаграмму и пересыпает ее согласно своей таблице локальной многоадресатной маршрутизации.

Проще всего понять строение магистрали MBONE, представив ее в виде виртуальной сети, созданной на основе глобальной сети Internet (которая, в свою

очередь, также является виртуальной сетью). Магистраль MBONE состоит из набора многоадресатных маршрутизаторов, связанных с помощью двухточечных соединений. Некоторые из этих абстрактных двухточечных соединений являются физическими соединениями (выделенными каналами связи), а другие создаются с помощью многоадресатного туннелирования. Детали этой структуры скрыты от программного обеспечения, выполняющего многоадресатную маршрутизацию. Поэтому, когда программа `mrouted` строит дерево многоадресатной пересылки для пары (*группа, отправитель*), она рассматривает туннель как обычный двухточечных канал связи, соединяющий два маршрутизатора.

Использование многоадресатного туннелирования имеет два последствия. Во-первых, поскольку стоимость прокладки некоторых туннелей существенно выше, чем других, нельзя считать все туннели равноценными. В программе `mrouted` эта проблема решена за счет того, что администратор может присвоить каждому туннелю свою стоимость, которая будет учитываться при выборе маршрутов. Как правило, администратор присваивает стоимость туннелю на основе количества переходов, которые совершают дейтаграмма, проходя по объединенной сети. Стоимость можно также присваивать, отталкиваясь от административных границ (например, очевидно, что стоимость туннеля, соединяющего два сетевых центра одной компании, должна быть намного ниже стоимости туннеля, ведущего в сетевой центр другой компании). Во-вторых, поскольку пересылка дейтаграмм в протоколе DVMRP зависит от наличия информации о кратчайшем маршруте к каждому отправителю и поскольку информация о многоадресатных туннелях не обрабатывается в стандартных протоколах маршрутизации, программа протокола DVMRP должна самостоятельно определить вариант пересылки одноадресатных дейтаграмм с учетом туннелирования.

## 17.25. Альтернативные протоколы

Хотя протокол DVMRP уже много лет используется в магистрали MBONE, по мере роста глобальной сети Internet инженеры группы IETF стали осознавать все его недостатки. Подобно RIP, в протоколе DVMRP для обозначения бесконечной длины маршрута используется небольшое целое число. Обратите внимание, что количество информации, которая нужна программе протокола DVMRP для выполнения маршрутизации, — довольно большое. Кроме данных для каждой активной пары (*группа, отправитель*), в таблице многоадресатной маршрутизации приходится также хранить информацию о ранее активных группах. Она нужна для того, чтобы маршрутизатор “знал”, куда отправлять запрос на подсоединение ветви, если узел сети “решит” присоединиться к отсеченной группе. И наконец, в протоколе DVMRP используется метод широковещания и отсечения, который создает дополнительный трафик во всех сетях до тех пор, пока не будет полностью распространена информация о членах группы. По иронии судьбы в протоколе DVMRP также используется дистанционно-векторный алгоритм распространения информации о членах группы, что явно не способствует быстрому распространению информации.

Если свести воедино все недостатки протокола DVMRP, можно прийти к выводу, что он плохо приспособлен к росту сети и не может справиться с большим количеством маршрутизаторов, многоадресатных групп или частыми изменениями состава группы. Таким образом, протокол DVMRP не может использоваться в глобальной сети Internet в качестве универсального протокола многоадресатной маршрутизации.

Чтобы устранить недостатки протокола DVMRP, инженеры группы IETF провели исследования других протоколов многоадресатной передачи. Их усилия не пропали даром. Было предложено несколько альтернативных методов

многоадресатной маршрутизации, среди которых можно выделить следующие: *наращиваемые от центра дерева* (*Core Based Trees*, или *CBT*), *независящая от протокола маршрутизации многоадресатная передача* (*Protocol Independent Multicast* или *PIM*) и *многоадресатные расширения протокола OSPF* (*MOSPF*). Каждый из указанных протоколов предназначен для решения проблем, связанных с ростом объединенной сети, однако способы их решения неизначительно отличаются друг от друга. Несмотря на то что все перечисленные протоколы уже реализованы в виде программного обеспечения, а протоколы PIM и MOSPF используются в отдельных частях магистрали MBONE, ни один из них пока не стал обязательным стандартом.

## 17.26. Наращиваемые от центра дерева (СВТ)

В протоколе СВТ разработчики постарались избежать широковещательного режима передачи дейтаграмм и позволили всем отправителям (там где это возможно) совместно использовать одно дерево пересылки дейтаграмм. Чтобы избежать широковещательного режима передачи, в протоколе СВТ многоадресатные пакеты не пересыпаются по маршруту до тех пор, пока хотя бы один узел сети, расположенный по этому маршруту, не присоединится к многоадресатной группе. Следовательно, в протоколе СВТ применен обратный, по сравнению с протоколом DVMRP, метод решения задачи многоадресатной пересылки. Как уже говорилось, в протоколе DVMRP дейтаграммы пересыпаются до тех пор, пока маршрутизатору не поступит отрицательная информация (т.е. что на данном маршруте больше нет членов группы). В протоколе СВТ дейтаграммы по заданному маршруту не пересыпаются до тех пор, пока маршрутизатор не получит положительную информацию (т.е. что на данном маршруте появились члены группы). Говорят, что вместо использования принципа *управления данными* в протоколе СВТ применен метод *управления по запросу*.

Принцип управления по запросу, применяемый в протоколе СВТ, означает, что при присоединении узла сети к определенной многоадресатной группе локальный маршрутизатор до начала пересылки дейтаграмм должен известить об этом другие маршрутизаторы с помощью IGMP-сообщений. Остается выяснить, какие именно “другие” маршрутизаторы должен оповестить локальный маршрутизатор? От ответа на этот вопрос зависит, как будет реализована структура многоадресатной маршрутизации во всех управляемых по запросу системах. Напомним, что в системах, управляемых данными, для определения получателей маршрутных сообщений маршрутизатор должен проанализировать входящий трафик, поскольку эти сообщения отправляются в направлении, обратном прибывающим пакетам. В системах, управляемых по запросу, до объявления информации о членах группы, маршрутизатор не может принять трафик для данной группы.

Для создания дерева многоадресатной пересылки в протоколе СВТ используется комбинация статических и динамических алгоритмов. Чтобы сделать структуру дерева расширяемой, в протоколе СВТ объединенная сеть разбивается на зоны. Размер зоны определяется сетевыми администраторами. В пределах зоны один из маршрутизаторов назначается *центральным маршрутизатором*. Остальные маршрутизаторы должны быть сконфигурированы так, чтобы они легко находили центральный маршрутизатор своей зоны. Адрес центрального маршрутизатора может быть указан либо статически, либо для его поиска задействуется специальный механизм обнаружения. В любом случае поиск центрального маршрутизатора выполняется только при начальной загрузке вспомогательного маршрутизатора.

Информация о центральном маршрутизаторе зоны является важной, поскольку она позволяет многоадресатным маршрутизаторам этой зоны построить общее дерево (*shared tree*) для этой зоны. Как только узел сети присоединяется к многоадресатной группе, локальный маршрутизатор  $L$ , получив запрос от этого сетевого узла, создает в рамках протокола СВТ запрос на присоединение, который он посыпает центральному маршрутизатору в виде обычной одноадресатной дейтаграммы. Каждый промежуточный маршрутизатор, расположенный по ведущему к центральному маршрутизатору маршруту, анализирует содержимое запроса. Как только запрос достигает маршрутизатора  $R$ , который является частью общего дерева протокола СВТ, маршрутизатор  $R$  отсылает в ответ сигнал подтверждения приема запроса, передает информацию о членах группы своему предку по дереву и начинает пересыпать предназначенный для группы трафик. Во время передачи краевому маршрутизатору сигнала, подтверждающего получение запроса, промежуточные маршрутизаторы также анализируют содержимое сообщения и настраивают свои таблицы многоадресатной маршрутизации так, чтобы можно было пересыпать предназначенные для группы дейтаграммы. В результате маршрутизатор  $L$  присоединяется к ветви дерева пересылки, исходящей от маршрутизатора  $R$ . Можно сделать следующий вывод.

*Поскольку в протоколе СВТ используется принцип системы, управляемой по запросу, объединенная сеть разделяется на зоны и для каждой зоны назначается центральный маршрутизатор. Остальные маршрутизаторы зоны строят динамическое дерево пересылки, отсылая центральному маршрутизатору запросы на присоединение.*

В протоколе СВТ используется специальный механизм для обслуживания ветвей дерева, который помогает обнаружить обрыв соединения между парой маршрутизаторов. Для проверки соединения каждый маршрутизатор периодически посыпает СВТ-запрос на отклик своему предку по дереву (т.е. следующему маршрутизатору расположенному по ведущему к центру маршруту). Если запрос не подтверждается, маршрутизатор в рамках протокола СВТ сообщает об этом всем зависящим от него маршрутизаторам, после чего повторно присоединяется к дереву в другой точке.

## 17.27. Независящая от протокола маршрутизации многоадресатная передача (PIM)

На самом деле в состав протокола PIM входит два независимых протокола, которые имеют мало общего друг с другом, кроме названия и форматов заголовка базовых сообщений: *PIM-DM* и *PIM-SM*. Название первого протокола расшифровывается как *Protocol Independent Multicast – Dense Mode*, т.е. уплотненный режим независящий от протокола многоадресатной передачи. Название второго протокола расшифровывается как *Protocol Independent Multicast – Sparse Mode*, т.е. разреженный режим независящий от протокола многоадресатной передачи. Эти два протокола возникли в результате того, что ни один из существующих протоколов не мог эффективно функционировать во всех возможных ситуациях. В частности, уплотненный режим протокола PIM предназначен для работы в локальной сети, когда члены всех существующих многоадресатных групп подключены ко всем или практически ко всем сетям. Разреженный режим протокола PIM предназначен для работы в глобальной сети, когда члены многоадресатной группы подключены только к небольшому подмножеству всех возможных сетей.

### **17.27.1. Уплотненный режим протокола PIM (PIM-DM)**

Поскольку уплотненный режим протокола PIM предназначен для работы в сетях с малым временем задержки передачи пакетов и достаточно большой пропускной способностью, протокол был оптимизирован для обеспечения гарантированной доставки сообщений, а не для уменьшения нагрузки на сеть. Поэтому в режиме PIM-DM используется метод широковещания и отсечения, как и в протоколе DVMRP. В начале работы для рассылки поступающих многоадресатных дейтаграмм маршрутизатор использует широковещательный режим и алгоритм RPF. Пересылка прекращается только после получения явного запроса на отсечение.

### **17.27.2. Независимость от протокола маршрутизации**

Ключевое отличие между протоколом DVMRP и уплотненным режимом протокола PIM заключается в том, что для работы протокола PIM-DM должна быть доступна правильная маршрутная информация. В частности, для использования RPF в уплотненном режиме протокола PIM необходима обычная одноадресатная маршрутная информация, т.е. маршрутизатору должны быть известны кратчайшие маршруты к каждому получателю. Однако, в отличие от протокола DVMRP, в протоколе PIM-DM не предусмотрено средств для распространения стандартных маршрутов. Вместо этого предполагается, что маршрутизатор, используя один из стандартных протоколов маршрутизации, вычисляет кратчайший маршрут к каждому получателю, заносит его в свою таблицу маршрутизации и периодически обновляет его. Фактически, в некоторой степени *независимость* режима PIM-DM от протокола маршрутизации заключается в его способности сосуществовать со стандартными протоколами маршрутизации. Следовательно, маршрутизатор может использовать любой из уже описанных протоколов (например, RIP или OSPF) для поддержки корректных одноадресатных маршрутов, а в уплотненном режиме протокола PIM могут использоваться маршруты, созданные любым из этих протоколов. Подведем итог сказанному выше.

*Хотя для работы уплотненного режима протокола PIM необходима таблица, содержащая правильные одноадресатные маршруты, в нем нет средств для их распространения по сети. Вместо этого предполагается, что каждый маршрутизатор должен поддерживать правильные одноадресатные маршруты с помощью одного из стандартных протоколов маршрутизации.*

### **17.27.3. Разреженный режим протокола PIM (PIM-SM)**

Разреженный режим протокола PIM можно рассматривать как расширение основных идей, реализованных в протоколе СВТ. Подобно протоколу СВТ, в PIM-SM используется метод управления по запросу. Как и в протоколе СВТ, для работы PIM-SM необходима информация о точке, в которую будут отсылаться сообщения о присоединении. Она называется *точкой сбора (Rendezvous Point, или RP)*. В точке сбора располагается маршрутизатор, который является функциональным эквивалентом центрального маршрутизатора протокола СВТ. Когда к многоадресатной группе присоединяется узел сети, локальный маршрутизатор отсылает маршрутизатору точки сбора одноадресатный запрос о *присоединении*. Маршрутизаторы, расположенные на маршруте, проходящем через точку сбора, анализируют сообщение, и если какой-либо из них уже является частью дерева, он перехватывает это сообщение и отвечает на него. Поэтому в протоколе PIM-SM,

как и в протоколе СВТ, создается общее дерево пересылки для каждой группы. Корень каждого из созданных деревьев находится в точке сбора<sup>9</sup>.

Основное отличие между протоколом СВТ и режимом PIM-SM заключается в способности последнего оптимизировать возможность подключения к точке сбора с помощью процедур реконфигурации. Например, вместо информации об одной точке сбора, в каждом из маршрутизаторов, поддерживающих протокол PIM-SM, могут быть указаны адреса маршрутизаторов, расположенных в потенциальных точках сбора. Ими можно воспользоваться в любой момент времени. Если текущий маршрутизатор, расположенный в точке сбора, становится по какой-либо причине недоступным (например, из-за обрыва соединения), маршрутизатор, поддерживающий режим PIM-SM, выбирает другой доступный маршрутизатор и начинает перестраивать структуру дерева пересылки для каждой многоадресатной группы. Вопросы реkonфигурации подробнее рассматриваются в следующем разделе.

#### 17.27.4. Переключение с общих деревьев на деревья кратчайшего пути

Помимо выбора альтернативного маршрутизатора точки сбора, в протоколе PIM-SM можно переключаться с общего дерева пересылки на *дерево кратчайшего пути* (*Shortest Path tree*, или *SP-дерево*). Чтобы понять причину такого перехода, рассмотрим схему сетевых соединений, изображенную на рис. 17.9.

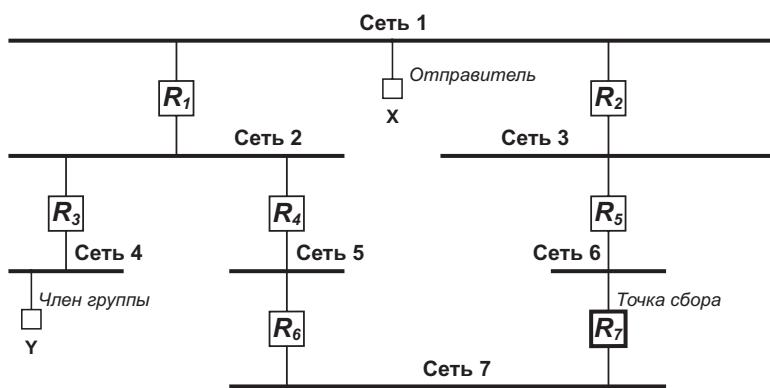


Рис. 17.9. Составность сетей с точкой сбора и многоадресатной группой, состоящей из двух членов. Использование метода управления по запросу для построения общего дерева с корнем в точке сбора приводит к неоптимальной маршрутизации

На рис. 17.9 в качестве точки сбора выбран маршрутизатор  $R_7$ . Таким образом, маршрутизаторы присоединяются к общему дереву, отсылая сообщения по ведущему к  $R_7$  маршруту. Например, допустим, что узлы  $X$  и  $Y$  присоединились к некоторой многоадресатной группе. Маршрут, ведущий к общему дереву от узла  $X$ , проходит через маршрутизаторы  $R_2$ ,  $R_3$  и  $R_7$ , а маршрут, ведущий от узла  $Y$  к общему дереву, проходит через маршрутизаторы  $R_3$ ,  $R_4$ ,  $R_6$  и  $R_7$ .

Хотя использование принципа общего дерева позволяет проложить кратчайшие маршруты от каждого узла сети к точке сбора, процесс маршрутизации не всегда бывает оптимальным. В частности, если члены группы находятся на

<sup>9</sup> Дейтаграмма, отсылаемая узлом многоадресатной группе, сначала передается по туннелю в точку сбора, а затем пересыпается в режиме многоадресатной передачи вниз по узлам общего дерева.

большом расстоянии от точки сбора, неэффективность маршрутизации может быть довольно ощутимой. Например, из рис. 17.9 видно, что, когда узел  $X$  отсылает дейтаграмму группе, сначала выполняется ее маршрутизация от узла  $X$  к маршрутизатору точки сбора  $R_7$ , а затем — к узлу  $Y$ . В результате дейтаграмма проходит через шесть маршрутизаторов, хотя оптимальный (т.е. кратчайший) путь от узла  $X$  к узлу  $Y$  проходит только через два маршрутизатора ( $R_1$  и  $R_3$ ).

В разреженном режиме протокола PIM предусмотрен механизм, позволяющий маршрутизатору переключаться между общим деревом и деревом кратчайшего маршрута к источнику. Последнее иногда называют *деревом источника* (*source tree*). Хотя принцип переключения с одного дерева на другое, очевидно, уже понятен, многие детали его структуры усложняют протокол. Например, в большинстве реализаций для осуществления перехода используется метод измерения интенсивности получаемого трафика. Если интенсивность трафика, поступающего от некоторого отправителя, превышает предварительно заданную пороговую величину, маршрутизатор начинает искать кратчайший маршрут<sup>10</sup>. К сожалению, интенсивность трафика может быстро меняться. Поэтому, чтобы предотвратить колебания, маршрутизаторам приходится применять гистерезис. Кроме того, изменение интенсивности трафика вызывает необходимость взаимодействия маршрутизаторов, расположенных по кратчайшему пути. Они должны пересыпать предназначенные для группы дейтаграммы. Интересно, что, поскольку изменение интенсивности трафика влияет только на одного отправителя, маршрутизатор должен поддерживать связь с общим деревом, чтобы иметь возможность получать дейтаграммы от других отправителей. Маршрутизатор должен располагать достаточным количеством маршрутной информации, чтобы избежать пересылки нескольких копий каждой дейтаграммы, поступающей от пары (*группа, отправитель*), для которой создано дерево кратчайшего пути.

## 17.28. Многоадресатные расширения протокола OSPF (MOSPF)

Выше уже говорилось о том, что для построения дерева доставки в протоколе многоадресатной маршрутизации наподобие PIM может использоваться информация, содержащаяся в обычной (одноадресатной) таблице маршрутизации. Исследователи изучили вопрос более широко: как использовать дополнительную информацию, собранную стандартными протоколами маршрутизации, для выполнения многоадресатной маршрутизации? В частности, при использовании протокола маршрутизации, отслеживающего состояние соединения, такого как OSPF, каждому маршрутизатору доставляется копия топологической схемы объединенной сети. Если быть точным, то в протоколе OSPF каждому маршрутизатору доставляется копия топологической схемы только той зоны, к которой принадлежит маршрутизатор. Поэтому, если такая информация доступна, протоколы многоадресатной передачи могут ее использовать для вычисления дерева пересылки.

Эта идея была положена в основу работы протокола, который называется *многоадресатным расширением протокола OSPF (MOSPF)*. В нем используется топологическая база данных протокола OSPF для создания дерева пересылок для каждого отправителя. Преимущество протокола MOSPF заключается в том, что он относится к классу систем, управляемых по запросу. Это означает, что предназначенный для определенной группы трафик не распространяется до тех пор, пока в нем не возникнет необходимость (например, в случае присоединения

<sup>10</sup> Существует как минимум одна реализация протокола PIM-SM, в которой маршрутизатор начинает искать кратчайший маршрут сразу (т.е. пороговая величина интенсивности трафика устанавливается равной нулю).

узла сети к группе или отключения от нее). Недостатком систем, управляемых по запросу, является высокая стоимость распространения маршрутной информации, поскольку все маршрутизаторы зоны должны хранить информацию о членах каждой группы. Кроме того, эту информацию периодически необходимо синхронизировать, чтобы обеспечить все маршрутизаторы одинаковой базой данных. В результате, протокол MOSPF генерирует меньший поток данных, но посыпает больше маршрутной информации, чем управляемые данными протоколы.

Хотя примененный в протоколе MOSPF подход, заключающийся в рассылке всем маршрутизаторам информации обо всех группах, действует в пределах зоны, его нельзя распространить на объединенную сеть произвольного размера. Таким образом, в протоколе MOSPF многоадресатная маршрутизация между зонами выполняется несколько иначе. Как известно, в протоколе OSPF один или несколько маршрутизаторов зоны назначаются в качестве *пограничного маршрутизатора* (*Area Border Router*, или *ABR*), в функции которого входит распространение маршрутной информации другим зонам. В протоколе MOSPF эта идея с пограничными маршрутизаторами несколько расширена. Теперь каждый пограничный маршрутизатор является *многоадресатным пограничным маршрутизатором зоны* (*Multicast Area Border Router*, или *MABR*). Он должен распространять другим зонам информацию о членах группы. В маршрутизаторах MABR используется не симметричная передача информации, а принцип ядра. При этом информация о членах группы своей зоны распространяется только в ядро сети (магистральную сеть) и не распространяется из ядра вниз по иерархии.

Маршрутизатор MABR может распространять многоадресатную информацию другой зоне, не являясь при этом активным получателем трафика. В каждой зоне назначается специальный маршрутизатор для получения многоадресатных пакетов от имени этой зоны. Когда из внешней зоны поступает многоадресатный трафик, предназначенный для всех групп этой зоны, он отсылается специально-му получателю, который иногда называют *многоадресатным групповым получателем* (*multicast wildcard receiver*).

## 17.29. Надежная многоадресатная передача и перегрузка уведомлениями

Под термином система с *надежной многоадресатной передачей* будем понимать систему, в которой используется принцип многоадресатной доставки пакетов и гарантируется, что все члены группы будут получать данные упорядоченно, без потерь, дублирования или искажений. Теоретически в такой системе должны использоваться преимущества метода пересылки пакетов, который является более эффективным, чем обычное широковещание, а также обеспечивающее сохранность доставляемых получателю данных. Таким образом, системы с надежной многоадресатной передачей имеют большое потенциальное преимущество и применимость (например, такая система может использоваться на фондовой бирже для сообщения большому количеству получателей цен на акции).

На практике, реализовать систему с надежной многоадресатной передачей не так-то просто и понятно, как это может показаться на первый взгляд. Во-первых, если в многоадресатной группе имеется несколько отправителей, понятие упорядоченной доставки дейтаграмм теряет смысл. Во-вторых, мы убедились, что в широко используемых системах многоадресатной пересылки (например, на основе протокола RPF) могут возникнуть дубли дейтаграмм даже в небольших объединенных сетях. В-третьих, помимо того, что надежная система многоадресатной пересылки должна гарантировать, что все данные в конечном счете достигнут своих получателей, она также должна обеспечивать для

прикладных программ, пересылающих большие потоки данных (например, аудио или видео), небольшое время задержки и устойчивую синхронизацию. В четвертых, поскольку для обеспечения надежной передачи получатель должен периодически посыпать отправителю сигналы подтверждения приема, а многоадресатная группа может состоять из произвольного количества членов, при использовании традиционных надежных протоколов отправитель должен обрабатывать неограниченно большое количество уведомлений. К сожалению, ни один компьютер пока не может справиться с этой задачей, поскольку его вычислительные мощности ограничены. Эта проблема, называемая *перегрузкой уведомлениями* (*ACK implosion*), была предметом многочисленных исследований.

Для решения проблемы перегрузки уведомлениями в надежных протоколах многоадресатной передачи применяют иерархический подход, при котором многоадресатная передача ограничивается одним отправителем<sup>11</sup>. Прежде чем будут отосланы данные, от отправителя ко всем членам группы необходимо создать дерево пересылки и определить *точки уведомления* (*acknowledgement points*).

Точку уведомления называют также *сборным пунктом уведомлений* (*acknowledgement aggregator*), или *отмеченным маршрутизатором* (*designated router, DR*). В ней находится маршрутизатор, обслуживающий дерево пересылки, который кэширует пересылаемые данные и обрабатывает уведомления, поступающие от узлов и маршрутизаторов, расположенных внизу дерева. Если возникает необходимость повторной передачи, точка уведомления посылает копию утерянных данных из своего кэша.

В большинстве систем надежной многоадресатной передачи используются отрицательные, а не положительные сигналы подтверждения приема. Другими словами, узел сети не посыпает уведомлений до тех пор, пока дейтаграмма не будет утеряна. Чтобы узел смог обнаружить потерю дейтаграммы, ей присваивается уникальный порядковый номер. Обнаружив потерю, узел сети отсылает отрицательное уведомление (сигнал NACK), в ответ на которое выполняется повторная передача данных. Отрицательное уведомление распространяется вверх по дереву пересылки в направлении отправителя до тех пор, пока не будет достигнута точка уведомления. Сигнал NACK обрабатывается маршрутизатором в точке уведомления, после чего он повторно отправляет копию утерянной дейтаграммы вниз по дереву пересылки.

Каким же образом маршрутизатор, находящийся в точке уведомления, может удостоверится, что у него имеются копии всех дейтаграмм данной последовательности? Он руководствуется тем же принципом, что и узел сети. Когда в точку уведомления поступает дейтаграмма, маршрутизатор проверяет ее порядковый номер, помещает копию дейтаграммы в свою память и затем распространяет дейтаграмму вниз по дереву пересылки. Если в точке уведомления обнаруживается, что дейтаграмма утеряна, маршрутизатор отсылает сигнал NACK в направлении отправителя вверх по дереву. При этом сигнал NACK либо достигает другой точки уведомления, в которой имеется копия дейтаграммы (в этом случае повторно посыпается копия дейтаграммы, находящаяся в точке уведомления), либо самого отправителя дейтаграммы, который повторно высылает утерянную дейтаграмму.

Использование ветвящейся топологической схемы и точек уведомления является важнейшей предпосылкой успешного функционирования надежной системы многоадресатной передачи. Если точек уведомления недостаточно, утерянная

<sup>11</sup>Обратите внимание, что это ограничение не снижает функциональных возможностей системы, поскольку один отправитель может переслать в многоадресатном режиме любое сообщение, которое он получил в режиме одноадресной передачи. Таким образом, узел сети может послать пакет этому отправителю, который затем перешлет его группе в режиме многоадресатной передачи.

дейтаграмма может перегрузить отправителя сигналами АСК. В частности, если маршрутизатор имеет много потомков, то вследствие утери дейтаграммы он может быть перегружен запросами на повторную передачу. К сожалению, создать алгоритм автоматического выбора точек уведомления непросто. Следовательно, для работы программ поддержки многих надежных протоколов многоадресатной передачи, сетевой администратор должен их сконфигурировать вручную. Поэтому режим надежной многоадресатной передачи предназначен для использования в системах, предоставляющих услуги на протяжении длительного времени, работающих в редко изменяющихся топологиях, а также в том случае, когда промежуточные маршрутизаторы могут выполнять функции точек уведомления.

Существует ли альтернативный подход к реализации надежного протокола многоадресатной передачи? Конечно, да, и он связан с решением проблемы уменьшения количества повторно пересылаемых дейтаграмм. В идеальном случае таких дейтаграмм вообще не должно быть. Согласно одной из методик, отправитель должен отправлять избыточное количество копий дейтаграммы. Например, вместо отправки одной копии каждой дейтаграммы, отправитель отправляет  $N$  таких копий (как правило, 2 или 3). Прием пересылки избыточного количества дейтаграмм особенно хорошо срабатывает в случае, когда в маршрутизаторах реализован принцип произвольного раннего обнаружения (*RED*), поскольку вероятность того, что будет отвергнуто более одной копии дейтаграммы, очень мала.

Еще один способ передачи избыточной информации — добавление к дейтаграммам кодов коррекции ошибок. Этот метод широко используется в радиосвязи, а также при записи компакт-дисков. Суть его заключается в том, что к каждой отправляемой дейтаграмме добавляется специальный избыточный код, по которому можно будет на приемном конце в случае утери или повреждения дейтаграммы полностью восстановить ее содержимое без отсылки запроса на ее повторную передачу.

## 17.30. Резюме

Многоадресатная передача протокола IP является обобщением режима аппаратной многоадресатной передачи. Она обеспечивает доставку дейтаграммы нескольким получателям. В протоколе IP для определения многоадресатной доставки дейтаграмм используется адрес класса  $D$ . При выполнении многоадресатной рассылки обычно задействуются соответствующие возможности сетевого оборудования (если, конечно они доступны).

Многоадресатные группы протокола IP являются динамическими, т.е. узел сети может в любой момент присоединиться к группе или отключиться от нее. Для выполнения многоадресатной передачи по локальной сети узлам сети необходимо всего лишь “уметь” отсыпать и получать многоадресатные дейтаграммы. Однако режим многоадресатной передачи протокола IP не ограничивается только одной физической сетью. С помощью многоадресатных маршрутизаторов информация о членах группы распространяется в другие сети. При этом процесс маршрутизации строится так, чтобы каждый член многоадресатной группы получал копии всех отосланных этой группе дейтаграмм.

Узлы сети сообщают информацию о членах своей группы многоадресатным маршрутизаторам, используя протокол IGMP. Протокол IGMP был специально разработан для обеспечения эффективной маршрутизации и экономного использования сетевых ресурсов. В большинстве случаев трафик, создаваемый протоколом IGMP, невелик. Он состоит из периодических сообщений, посыпаемых группам многоадресатным маршрутизатором, а также из ответных сообщений

(по одному для каждой многоадресатной группы) узлов, принадлежащих к группе в этой сети.

Для распространения многоадресатной маршрутной информации по объединенной сети было разработано несколько протоколов. Существует два основных принципа многоадресатной передачи: управляемая данными и запросами. В любом случае количество информации, содержащейся в таблице многоадресатной пересылки, намного больше, чем в стандартной таблице одноадресатной маршрутизации. Дело в том, что для многоадресатной передачи для каждой пары (*группа, отправитель*) в таблице пересылки должны содержаться дополнительные сведения.

Не все маршрутизаторы глобальной сети Internet могут распространять многоадресатную маршрутную информацию и пересыпать многоадресатный трафик. Для передачи многоадресатного трафика между группами, расположенными в нескольких сетевых центрах, разделенных друг от друга объединенной сетью, которая не поддерживает режим многоадресатной маршрутизации, используется туннель на уровне протокола IP. При этом программа помещает многоадресатную дейтаграмму в обычную одноадресатную дейтаграмму и отправляет ее на другой конец туннеля по объединенной сети. Получатель должен извлечь и обработать многоадресатную дейтаграмму.

Под надежной многоадресатной передачей понимают системы пересылки многоадресатных пакетов, в которых гарантируется их доставка получателю. Чтобы устранить проблему перегрузки отправителя сигналами подтверждения приема, в надежных системах многоадресатной пересылки используются точки уведомления, расположенные в виде иерархической структуры, либо отсылается избыточная информация.

## Материал для дальнейшего изучения

Стандарт для рассматриваемого в этой главе режима многоадресатной передачи протокола IP, включая версию 2 протокола IGMP, описан Дириングом (Deering) в [RFC 2236]. Протокол DVMRP описан Вайцманом (Waitzman), Партриджем (Partridge) и Дириингом (Deering) в [RFC 1075]. Эстрин (Estrin) и др. в [RFC 2362] описывают разреженный режим протокола PIM, а Балларди (Ballardie) в [RFC 2189] и [RFC 2201] рассматривает протокол СВТ. Протокол MOSPF описан Мойем (Moy) в [RFC 1585].

Структура многоадресатной сетевой магистрали описана Эрикссоном (Eriksson) в статье [48]. Каснер (Casner) и Дириинг (Deering) в статье [15] описывают первую систему многоадресатной передачи, предназначенную для проведения заседаний проблемной группы IETF.

## Упражнения

- 17.1. Для формирования многоадресатного аппаратного адреса в стандарте семейства протоколов TCP/IP предусматривается использование младших 23 бит многоадресатного IP-адреса. Какое количество многоадресатных IP-адресов при этом может соответствовать одному многоадресатному аппаратному адресу?
- 17.2. Объясните, почему в многоадресатных IP-адресах используется только 23 из 28 возможных битов. (Подсказка. Учтите практические ограничения, налагаемые на количество групп, к которым может принадлежать узел сети, а также на количество узлов, подключаемых к одной сети.)

- 17.3. Согласно протоколу IP, узел сети должен проверять адреса получателей во всех входящих многоадресатных дейтаграммах и аннулировать те, что адресованы другой группе (т.е. той группе, к которой этот узел сети не подключен). Объясните, каким образом узел сети может получать многоадресатные дейтаграммы, предназначенные для группы, к которой он не подключен.
- 17.4. Многоадресатные маршрутизаторы должны располагать информацией о том, есть ли в данной сети члены определенной группы. Нужен ли им полный список узлов сети, которые принадлежат этой многоадресатной группе?
- 17.5. Назовите три известные вам прикладные программы, для которых желательна поддержка режима многоадресатной пересылки пакетов.
- 17.6. Согласно стандарту, программное обеспечение протокола IP должно обеспечить доставку копии каждой исходящей многоадресатной дейтаграммы прикладным программам, запущенным на узле, который принадлежит определенной многоадресатной группе. Поясните, насколько это требование усложняет или упрощает процесс программирования.
- 17.7. Если применяемое сетевое оборудование не поддерживает режим многоадресатной передачи, для рассылки многоадресатных пакетов в протоколе IP используется аппаратный широковещательный режим передачи. Какие проблемы могут при этом возникнуть? Есть ли какие-либо преимущества от использования режима многоадресатной передачи протокола IP в таких сетях?
- 17.8. Протокол DVMRP создан на основе протокола RIP. Ознакомьтесь с описанием протокола DVMRP в [RFC 1075] и сравните эти два протокола. Насколько протокол DVMRP сложнее протокола RIP?
- 17.9. В протоколе IGMP не предусмотрено подтверждение приема пакетов или их повторная передача, хотя он может использоваться в сетях с негарантированной доставкой пакетов. Что может произойти, если IGMP-запрос будет утерян? А если потерянется ответное сообщение?
- 17.10. Объясните, почему узел, подключенный к нескольким физическим сетям, должен подключаться к многоадресатной группе, относящейся только к одной из сетей. (*Подсказка*. Рассмотрите случай проведения аудиоконференции.)
- 17.11. Определите размер таблицы многоадресатной пересылки дейтаграмм, которая необходима для обработки многоадресатных аудиодейтаграмм, поступающих из 100 радиостанций, при условии, что каждая станция имеет около десяти миллионов слушателей в разных точках земного шара.
- 17.12. Объясните, почему в сети Internet могут использоваться только два типа многоадресатной передачи. Первый связан со статически сконфигурированными коммерческими службами, которые рассылают многоадресатные дейтаграммы большому количеству своих подписчиков. Второй — с динамически конфигурируемыми службами, рассылающими дейтаграммы небольшому числу подписчиков (например, при проведении конференц-связи между сотрудниками одной организации).

- 17.13.** Рассмотрите систему надежной многоадресатной передачи, в которой передается избыточное количество информации. Если вероятность возникновения ошибок при передаче данных в канале связи высока, то какой из методов обеспечения надежности является более эффективным: пересылка избыточного количества копий дейтаграмм или пересылка копии дейтаграммы, содержащей коды коррекции ошибок. Обоснуйте свой ответ.
- 17.14.** Управляемая данными многоадресатная маршрутизация лучше всего работает в локальных сетях, которые имеют малую задержку и большую пропускную способность, в то время как маршрутизация управляемая по запросу наиболее эффективна в глобальных сетях, которые имеют ограниченную пропускную способность и более высокую задержку. Стоит ли создать один общий протокол, который объединил бы в себе оба принципа? Обоснуйте свой ответ. (*Подсказка.* Внимательно изучите спецификацию протокола MOSPF.)
- 17.15.** Придумайте количественный критерий, который можно было бы использовать для определения момента перехода от общего дерева на дерево кратчайшего пути в протоколе PIM-SM.
- 17.16.** Ознакомьтесь со спецификацией протокола PIM и найдите в нем определение “разреженности”, используемое в режиме PIM-SM. Приведите пример объединенной сети, в которой совокупность всех членов группы является разреженной, но для которой протокол DVMRP лучше всего подходит в качестве протокола многоадресатной маршрутизации.



# 18

## Протокол TCP/IP в сетях ATM

### 18.1. Введение

В предыдущих главах были рассмотрены основные составляющие семейства протоколов TCP/IP, а также продемонстрировано, каким образом его компоненты функционируют в локальных и глобальных сетях, созданных на основе стандартных технологий. В этой главе исследуется вопрос применения семейства протоколов TCP/IP, которое изначально разрабатывалось для использования в сетях, не требующих установки соединения с получателем, в сетях, построенных на основе технологий, требующих установки такого соединения<sup>1</sup>. В этой главе будет показано, что структура семейства протоколов TCP/IP является чрезвычайно гибкой. Для использования этих протоколов в сетевом окружении, требующем установки соединения с получателем, необходимо изменить лишь некоторые детали механизма привязки адресов. При этом остальные протоколы семейства остаются без изменений.

Трудности возникают при использовании семейства протоколов TCP/IP в сетях с множественным доступом, не поддерживающих режим широковещания (*Non-Broadcast Multiple-Access Networks*, или NBMA). Речь идет о сетях, ориентированных на установку соединения между машинами, которые позволяют подключаться нескольким компьютерам, но не поддерживают режим широковещательной передачи пакетов от одного компьютера ко всем остальным машинам. Далее будет показано, что при использовании сетей NBMA модификации подлежат те из протоколов семейства TCP/IP, для работы которых требуется широковещательная рассылка пакетов. В качестве примера можно привести протокол ARP.

Для того чтобы конкретизировать обсуждение и связать его с доступным аппаратным обеспечением, все приведенные в этой главе примеры ориентированы на использование в сетях с асинхронным режимом передачи (ATM). В этой главе приводится подробное описание режима ATM, обсуждение которого было начато в главе 2, “Обзор основных сетевых технологий”. В следующих разделах описывается физическая топология сети ATM, обеспечивающая установку логических соединений между получателями, принцип подключения к сети ATM, а также протокол адаптации ATM, используемый для передачи данных. В конце главы рассматривается связь между режимом ATM и семейством протоколов TCP/IP. Будет описана методика адресации, используемая в сетях ATM, а также показана связь между ATM-адресом узла сети и его IP-адресом. Также будет описана модифицированная форма протокола преобразования адресов (ARP), который

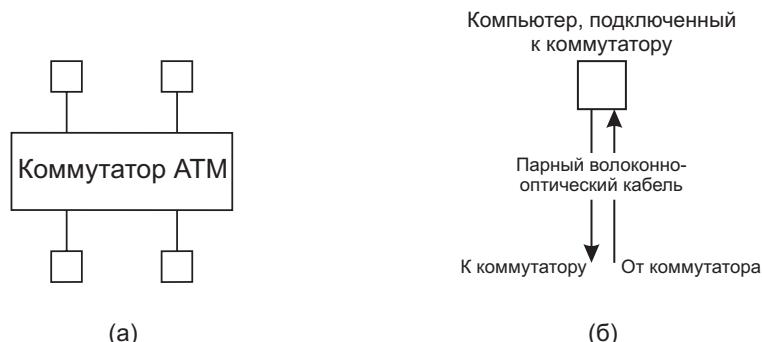
---

<sup>1</sup> В некоторых документах для обозначения технологий, не требующих установки соединения с получателем используется аббревиатура CL (*connectionless*), а для технологий, требующих установки соединения с получателем — CO (*connection-oriented*).

используется для преобразования IP-адреса в сетях, ориентированных на установку соединения между получателями, а также модифицированная форма протокола обратного преобразования адресов RARP, который может использоваться сервером для получения адресов и управления. И самое важное: в этой главе будет продемонстрировано, каким образом IP-дейтаграммы могут передаваться по сетям ATM без фрагментации.

## 18.2. Оборудование для сетей ATM

Подобно большинству технологий, требующих установки соединения с получателем, сети ATM строятся на основе стандартных блоков, роль которых выполняют специализированные электронные коммутаторы. Как правило, к одному такому коммутатору можно подключить от 16 до 32 компьютеров<sup>2</sup>. Хотя узел сети можно подключить к коммутатору ATM посредством медного кабеля, в большинстве устройств для повышения скорости передачи данных используют волоконно-оптическую линию связи. На рис. 18.1 изображена структурная схема сети ATM с коммутатором, к которому подключены компьютеры пользователей, и необходимыми пояснениями.



*Рис. 18.1. Схематическое изображение коммутатора ATM, к которому подключено 4 компьютера (а); схема подключения компьютера к коммутатору (б). Для передачи данных к коммутатору и от него используется парный волоконно-оптический кабель*

Плата сетевого интерфейса подключается к шине компьютера. Конструктивно она состоит из приемника и передатчика оптических сигналов, а также схемы преобразования электрических сигналов, посылаемых компьютером, в оптические импульсы, проходящие по световодам, и наоборот. Поскольку по каждому оптоволоконному кабелю световой сигнал передается только в одном направлении, для создания соединения, которое позволило бы компьютеру как посылать, так и получать данные, необходимо два оптоволоконных кабеля.

## 18.3. Большие сети ATM

Выше уже говорилось о том, что к одному коммутатору ATM можно подключить ограниченное количество компьютеров. Поэтому для создания крупной сети приходится соединять между собой несколько коммутаторов. В частности, чтобы

<sup>2</sup> В крупных сетях используются коммутаторы, к которым можно подключить и большее количество компьютеров. Однако, основной момент заключается в том, что к одному коммутатору можно подключить ограниченное количество компьютеров.

подключить к одной сети компьютеры, расположенные в двух сетевых центрах, в каждом из центров нужно установить по коммутатору, а затем соединить их между собой. Принцип соединения коммутаторов между собой незначительно отличается от подключения узла сети к коммутатору. Например, данные между коммутаторами, как правило, передаются на более высокой скорости, чем между узлом сети и коммутатором; кроме того, при этом используются немногим другие протоколы. На рис. 18.2 показана описываемая топологическая схема, а также приведены ключевые отличия между интерфейсами “сеть–сеть” (*Network to Network Interface, NNI*) и “пользователь – сеть” (*User to Network Interface, UNI*).



Рис. 18.2. Три коммутатора ATM, объединенные в одну большую сеть. Хотя для подключения двух коммутаторов предназначен интерфейс NNI, в небольших частных сетях для этой цели вполне может использоваться интерфейс UNI

#### 18.4. Логическое представление сети ATM

При разработке сети ATM ставилась задача создать систему сквозной передачи данных от отправителя до конечного получателя. Для компьютера, подключенного к сети ATM, совокупность коммутаторов<sup>3</sup> представляет собой однородную сеть. Как и в обычной телефонной сети, передающей голосовые данные, а также в соединенной мостами сети Ethernet или в объединенной сети на основе протокола TCP/IP, в сети ATM детали используемого физического оборудования скрыты от пользователя. Поэтому для него создается видимость единой физической сети, к которой подключено большое количество компьютеров. В качестве примера на рис. 18.3 показана логическая схема сети ATM, к которой подключено восемь компьютеров.



Рис. 18.3. Логическая схема сети ATM, изображенная на рис. 18.2. В сети ATM создается видимость единой унифицированной сети, в которой два любых компьютера могут обмениваться информацией

Таким образом, технология ATM является общепринятой абстракцией для однородного сетевого оборудования, так же, как семейство протоколов TCP/IP для гетерогенных систем.

<sup>3</sup> Их еще называют коммутационным центром (*switching fabric*).

*Несмотря на то что физически коммутационный центр может состоять из нескольких коммутаторов, применение технологии ATM позволяет создать для пользователя видимость единой физической сети, к которой подключено множество компьютеров. При этом любой подключенный к сети ATM компьютер может непосредственно осуществлять обмен информацией с любым другим компьютером. Все компьютеры сети не располагают данными о структуре самой физической сети.*

## 18.5. Два типа подключения к сети ATM

Сеть ATM является системой, ориентированной на установку соединения между получателями. Это значит, что прежде чем послать данные удаленному получателю, узел должен установить с ним соединение. Соединение — это абстрактное понятие, аналогичное телефонному звонку. Хотя существует только один вид базового соединения, в сети ATM предусмотрено два способа создания такого соединения. Первый называется *постоянным виртуальным каналом* (*Permanent Virtual Circuit*, или *PVC*), а второй — *коммутируемым виртуальным каналом* (*Switched Virtual Circuit*, или *SVC*)<sup>4</sup>.

### 18.5.1. Постоянные виртуальные каналы

На телефонном жаргоне канал PVC называют *технически обеспечиваемой услугой* (*provisioned service*). Техническое обеспечение подразумевает под собой всего лишь ручную настройку каждого коммутатора, расположенного по маршруту от отправителя до конечного получателя. Ручная настройка выполняется администратором, например, посредством ввода команд с консоли устройства в процессе коммутации. Хотя термины *канал PVC* и *технически обеспечиваемая услуга* могут показаться весьма загадочными, рассматриваемая нами идея очень проста. Более того, практически в любом сетевом оборудовании, ориентированном на установку соединения, предусмотрена возможность создания постоянного виртуального канала.

С одной стороны, ручная конфигурация имеет один очевидный недостаток: в нее нельзя быстро и легко внести изменения. Следовательно, канал PVC используется только для соединений, которые существуют без изменений продолжительное время (несколько недель или лет). С другой стороны, ручная конфигурация имеет свои преимущества: при создании канала PVC от всех коммутаторов не требуется поддержка одной стандартной сигнальной системы. Поэтому при использовании каналов PVC может применяться оборудование от разных производителей, которое может и не работать при создании каналов SVC. Во-вторых, каналы PVC зачастую необходимы для управления сетями, технического обслуживания, поиска и устранения неполадок.

### 18.5.2. Коммутируемые виртуальные каналы

В отличие от канала PVC, коммутируемый виртуальный канал (SVC) автоматически создается программным обеспечением и разрывается, как только в нем исчезает необходимость. Программное обеспечение узла сети инициирует создание канала SVC путем посылки запроса локальному коммутатору. В запросе указывается полный адрес удаленного компьютера, с которым необходимо установить канал SVC, а также параметры, определяющие требуемое качество

<sup>4</sup> В стандарте ATM для обозначения виртуального канала используется также англоязычный термин *virtual channel*.

соединения (например, пропускная способность и величина задержки). Затем узел сети ожидает, пока коммутатор ATM создаст канал и пришлет ответ на запрос. Маршрут от исходного узла по сети ATM (возможно, через несколько коммутаторов) к удаленному компьютеру устанавливается с помощью *сигнальной системы*<sup>5</sup> (*signaling system*), или системы вызовов.

Во время вызова каждый расположенный по маршруту коммутатор ATM и удаленный компьютер должны согласовать друг с другом процесс создания виртуального канала. После согласования коммутатор записывает в свои таблицы информацию о канале, резервирует необходимые для его создания ресурсы и посыпает запрос следующему коммутатору, расположенному по маршруту. Как только все коммутаторы и удаленный компьютер ответили на запрос, процесс вызова завершается, и коммутаторы, расположенные на обоих концах соединения, сообщают узлам сети, что виртуальный канал успешно установлен.

Как любое абстрактное понятие, соединение необходимо как-то идентифицировать. Для идентификации каждого виртуального канала в интерфейсе UNI используется 24-битовое целое число. При создании канала PVC идентификатор ему присваивает системный администратор.

Когда программное обеспечение узла сети пытается создать новый канал SVC, локальный коммутатор ATM присваивает ему идентификатор и сообщает об этом узлу сети. В отличие от технологий, не требующих установки соединений между получателями, в системе, ориентированной на установку такого соединения, не нужно указывать в каждом пакете адрес отправителя или получателя. Вместо этого, в каждый отправляемый пакет узел сети помещает идентификатор канала, а коммутатор помещает идентификатор канала в каждый доставляемый им пакет.

## 18.6. Маршруты, каналы и идентификаторы

Выше уже упоминалось, что при использовании технологии, ориентированной на установку соединения между получателями, каждому каналу присваивается уникальный целочисленный идентификатор, который указывается узлом сети при выполнении операций ввода-вывода, а также при закрытии канала. Однако в системах, ориентированных на установку соединения, уникальность такого идентификатора не является глобальной. Другими словами идентификатор канала является аналогом дескриптора ввода-вывода, который возвращается программой операционной системы при открытии файла. Подобно дескриптору ввода-вывода, идентификатор канала является подобием стенографического знака, применяемым программой для идентификации полного набора параметров, которые были использованы для создания канала. Как и дескриптор ввода-вывода, идентификатор канала является действительным только при открытом канале. Кроме того, значением идентификатора канала можно воспользоваться только в пределах одного сегмента соединения между коммутаторами. Дело в том, что идентификаторы одного и того же канала, полученные узлами сети на обоих концах сегмента виртуального канала, как правило, отличаются друг от друга. Например, отправитель может использовать для идентификации канала число 17, в то время как получатель — число 49. Поэтому при передаче пакета от одного узла сети до другого каждый коммутатор, расположенный по пути прохождения пакета, заменяет в нем номер идентификатора канала на текущий.

Формально, используемый в интерфейсе UNI идентификатор канала состоит из 24-битового целого числа, разделенного на два поля<sup>6</sup>: 8-битового

<sup>5</sup> Этот термин взят из телефонии.

<sup>6</sup> Идентификатор канала, используемый в интерфейсе NNI, имеет несколько другой формат и длину

идентификатора виртуального маршрута (Virtual Path Identifier, или VPI) и 16-битового идентификатора виртуального канала (Virtual Circuit Identifier, или VCI) (рис. 18.4). Зачастую весь идентификатор в целом называют парой VPI/VCI.



Рис. 18.4. Формат 24-битового идентификатора соединения в интерфейсе UNI. Идентификатор разделен на две части, которые идентифицируют виртуальный маршрут и виртуальный канал

Причина разделения идентификатора соединения на два поля VPI и VCI аналогична той, которая вызвала разделение IP-адреса на поле адреса сети и поле адреса узла. Если несколько виртуальных каналов проложены по одному и тому же маршруту, администратор может сделать так, чтобы у всех этих каналов был один и тот же идентификатор VPI. Идентификатор VPI может использоваться сетевым оборудованием ATM для выполнения более эффективной маршрутизации трафика. Коммерческие поставщики услуг связи могут также использовать идентификатор VPI в целях учета. Так, поставщик может взять с клиента плату за виртуальный маршрут, предоставив ему самому решать, сколько виртуальных каналов будет проложено по этому маршруту.

## 18.7. Механизм передачи ячеек в сети ATM

На самом низком уровне в сети ATM для передачи данных используются фреймы фиксированного размера, называемые ячейками (cells). В режиме ATM предполагается, что все ячейки должны быть одинакового размера, поскольку это позволяет создать быстродействующее коммутационное оборудование, а также эффективно обрабатывать голосовые и сетевые данные. Каждая ячейка ATM имеет длину 53 октета и состоит из заголовка размером 5 октетов, после которого следуют 48 октетов полезной нагрузки (т.е. данных). Формат заголовка ячейки UNI показан на рис. 18.5.

	0	1	2	3	4	5	6	7
0	Биты управления пересылкой				VPI (Первые 4 бита)			
+1	VPI (Последние 4 бита)				VCI (Первые 4 бита)			
+2			VCI (Средние 8 битов)					
+3	VCI (Последние 4 бита)			Тип полезной нагрузки				Приоритет
+4			Код избыточной циклической проверки					

Рис. 18.5. Формат заголовка ячейки UNI длиной 5 октетов, используемый при передаче пакетов между узлом сети и коммутатором. На рисунке каждый октет показан в отдельной строке; за заголовком следуют восемь октетов данных

## 18.8. Уровень адаптации ATM

Хотя на самом низком уровне для передачи данных в сети ATM используются ячейки небольшого размера, прикладные программы, которые передают данные по сети ATM, не работают с ними напрямую (т.е. не считывают и не записывают информацию в ячейки). Прикладные программы компьютера взаимодействуют с сетью ATM через так называемый *уровень адаптации ATM* (*ATM Adaptation Layer*), описание которого является частью стандарта ATM. Уровень адаптации выполняет несколько функций, включая обнаружение и исправление ошибок (например, потерю ячейки или разрушение данных в ней). Как правило, программно-аппаратные средства, реализующие уровень адаптации ATM, находятся прямо на плате сетевого интерфейса, там же, где и средства, обеспечивающие передачу и прием ячеек. На рис. 18.6 изображена структурная схема типичного интерфейса ATM, а также показаны потоки данных, поступающие от операционной системы компьютера через интерфейсную плату в сеть ATM и обратно.



Рис. 18.6. Упрощенная схема платы сетевого интерфейса ATM и потоков данных, протекающих через него. Программное обеспечение узла сети взаимодействует с протоколом уровня адаптации для пересылки и получения данных. Уровень адаптации предназначен для преобразования данных, помещаемых в ячейки и получаемых из них

При установке соединения узел сети должен указать используемый тип протокола уровня адаптации. Причем узлы, расположенные на обоих концах соединения, должны согласовать используемые ими типы протоколов. А это значит, что после установки соединения тип протокола уровня адаптации уже нельзя изменить. Подведем итого всему сказанному выше.

Для передачи данных в сети ATM на уровне сетевого оборудования используются небольшие ячейки фиксированного размера. Однако прикладные программы не работают напрямую с этими ячейками. Для этого используется специальный протокол более высокого уровня (адаптации), который обеспечивает прикладные программы средствами для передачи и приема данных по сети ATM. При создании виртуального канала компьютеры, расположенные на обоих концах канала должны совместно решить, какой именно протокол уровня адаптации будет использоваться для передачи данных.

## 18.9. Протокол адаптации ATM уровня 5

Для пересылки данных по сети ATM прикладные программы используют службы, которые предоставляет протокол адаптации ATM уровня 5 (*Adaptation Layer 5*, или *AAL5*). Интересно то, что хотя на самом низком уровне в режиме ATM предусмотрено использование небольших ячеек фиксированного размера, протокол AAL5 предоставляет прикладным программам интерфейс, позволяющий пересылать большие по размеру пакеты переменной длины. Таким образом, при использовании этого интерфейса для прикладных программ создается видимость, что сеть ATM относится к разряду технологий, не требующих установки соединения между получателями. В частности, в протоколе AAL5 предусмотрено, что в каждом пакете может содержаться от 1 до 65 535 октетов данных. Формат пакета, используемый протоколом AAL5, показан на рис. 18.7.

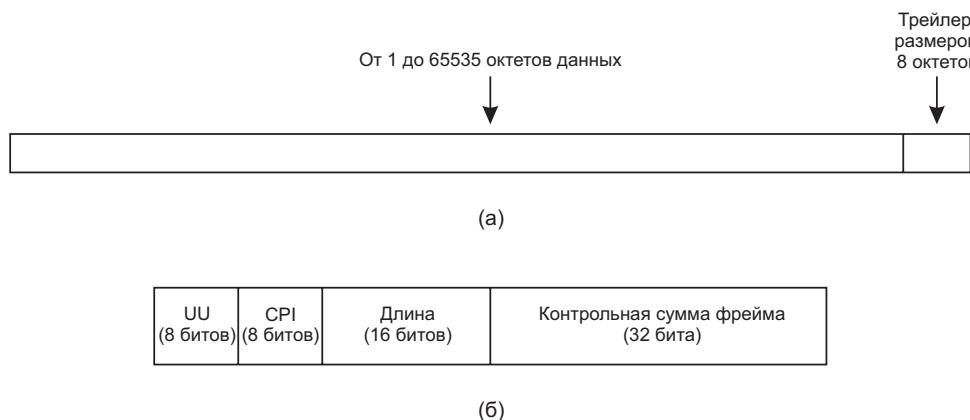


Рис. 18.7. Базовый формат пакета, используемый в протоколе AAL5 (а); формат полей, находящихся в трейлере пакета, длиной 8 октетов, который располагается в конце блока данных

В отличие от большинства сетевых фреймов, в которых управляющая информация хранится в заголовке, в протоколе AAL5 эта информация хранится в трейлере пакета, размером 8 октетов. Согласно стандарту, трейлер пакета состоит из поля длины, размер которого составляет 16 бит, 32-битового поля кода циклической избыточной проверки (CRC), который используется в качестве контрольной суммы фрейма, и двух 8-битовых полей, обозначенных на рис. 18.7 как *UU* и *CPI*. В настоящее время эти два поля пока не используются<sup>7</sup>.

<sup>7</sup> В поле *UU* может находиться любое значение, а поле *CPI* необходимо обнулить.

Перед передачей по сети ATM каждый пакет протокола AAL5 необходимо разделить на ячейки. После приема всех ячеек получатель должен объединить их обратно в пакет и только затем отправить его прикладной программе. Если длина пакета с учетом трейлера размером 8 октетов кратна 48 октетам, в результате разделения будут получаться полностью заполненные ячейки. Если же это условие не выполняется, последняя ячейка будет заполнена данными не до конца. Для обеспечения передачи пакетов переменной длины в протоколе AAL5 предусмотрено, что в последней ячейке может содержаться от 0 до 40 октетов данных, после которых следуют октеты, содержащие нули, а затем трейлер размером 8 октетов. Другими словами, в протоколе AAL5 трейлер пакета помещается в последние 8 октетов завершающей ячейки. Сделано это для того, чтобы его можно было легко найти и извлечь, не зная точной длины пакета.

## 18.10. Сходимость, сегментация и сборка пакетов в протоколе AAL5

При передаче данных через ATM-соединение с помощью протокола AAL5, прикладная программа отправляет блок данных через интерфейс AAL5. При этом программа протокола AAL5 создает трейлер пакета, разделяет пакет на части размером 48 октетов и передает каждую часть по сети ATM в одной ячейке. На принимающем конце соединения программа протокола AAL5 собирает поступившие ячейки в пакет, проверяет код циклической избыточной проверки CRC, чтобы убедиться в том, что все части пакета поступили неповрежденными, и передает полученный блок данных прикладной программе. Процесс разделения блока данных на ячейки и их последующая группировка на приемном конце называется *сегментацией и последующей сборкой ячеек ATM (Segmentation And Reassembly, или SAR)*<sup>8</sup>.

Протокол AAL5 полностью соответствует принципу разбиения на уровни, поскольку в нем функции сегментации и последующей сборки ячеек отделены от механизма передачи ячеек. Поэтому уровень передачи ячеек ATM можно отнести к классу “от машины к машине”, поскольку передача ячеек выполняется от одной машины до следующей, расположенной по маршруту к конечно-му получателю, например, от узла сети до коммутатора, или между двумя коммутаторами. В свою очередь уровень AAL5 относится к классу сквозных уровней, поскольку пакеты данных передаются непосредственно от отправителя к конечному получателю. Другими словами, программа протокола AAL5, запущенная на машине получателя, передает прикладной программе получателя точную копию блока, который был получен программой протокола AAL5, запущенной на машине отправителя.

Каким образом программа протокола AAL5, работающая на машине получателя, узнает количество ячеек в пакете? Для указания последней ячейки пакета в протоколе AAL5 используется значение младшего бита поля *типа полезной нагрузки*, расположенного в заголовке ячейки ATM. Этот бит устанавливается отправителем при передаче последней ячейки и называется *битом окончания пакета*. Таким образом, программа протокола AAL5 на машине получателя собирает все входящие ячейки до тех пор, пока не обнаружит ячейку с установленным битом окончания пакета. Для описания механизмов, распознавающих окончание пакета, в стандарте ATM используется специальный термин — *сходимость (convergence)*. Хотя в протоколе AAL5 для определения сходимости

<sup>8</sup> Процесс последующей сборки ячеек ATM в протоколе AAL5 во многом напоминает процесс объединения сегментов в IP-дейтаграмму. В обоих случаях большой блок данных разбивается на компоненты меньшего размера для их последующей передачи по сети.

проверяется значение только одного бита в заголовке ячейки, в остальных протоколах уровня адаптации ATM могут использоваться и другие механизмы сходимости. Подведем итог всему сказанному выше.

*Для передачи большого блока данных по виртуальному каналу ATM прикладная программа использует протокол адаптации ATM уровня 5. На машине отправителя модуль протокола AAL5 создает концевик пакета, разбивает блок данных на ячейки и передает их по виртуальному каналу. На машине конечного получателя модуль протокола AAL5 выполняет сборку ячеек и восстанавливает исходный блок данных, удаляет трейлер пакета и передает результирующий блок данных прикладной программе получателя. Для обозначения последней ячейки блока данных в ее заголовке устанавливается специальный бит.*

## 18.11. Инкапсуляция дейтаграмм и размер MTU в протоколе IP

Выше уже упоминалось, что для передачи дейтаграмм по сети ATM в протоколе IP используется протокол AAL5. Перед отсылкой данных отправитель должен установить с получателем виртуальный канал (PVC или SVC) и согласовать используемый тип протокола уровня адаптации ATM, например AAL5. При передаче дейтаграммы модулю протокола AAL5, отправитель указывает также пару идентификаторов VPI/VCI, которые определяют сам канал. Модуль протокола AAL5 создает трейлер дейтаграммы, разбивает ее на ячейки и передает их по сети. На приемном конце виртуального канала модуль протокола AAL5 выполняет сборку ячеек в пакет данных, вычисляет код CRC и проверяет целостность пакета, извлекает дейтаграмму из пакета и пересыпает ее модулю протокола IP.

На практике в протоколе AAL5 используется поле длины пакета, размер которого составляет 16 бит. Это позволяет отсылать в одном пакете до 65 535 (64К) октетов. Несмотря на это в семействе протоколов TCP/IP наложены ограничения на размер дейтаграмм, которые можно посыпать по сети ATM. В стандарте максимальный размер IP-дейтаграммы в сетях ATM устанавливается равным 9180 октетов<sup>9</sup>. Как и в любой сетевой технологии, если размер отправляемой дейтаграммы превышает установленный в сети размер максимальной единицы передачи данных (MTU), модуль протокола IP разбивает дейтаграмму на фрагменты и пересыпает каждый фрагмент модулю протокола AAL5. Таким образом, по умолчанию модуль протокола AAL5 принимает, передает и доставляет дейтаграммы, размер которых не превышает 9180 октетов. Подведем итог сказанному выше.

*При пересылке данных по сети ATM модуль протокола IP передает дейтаграмму модулю протокола адаптации ATM уровня 5. Несмотря на то что этот модуль может принимать и передавать пакеты длиной до 65535 октетов, в стандартах семейства протоколов TCP/IP для сетей ATM размер MTU ограничен на уровне 9180 октетов. Поэтому, если размер дейтаграммы превышает 9180 октетов, модуль протокола IP должен разбить ее на фрагменты и только после этого передать модулю протокола AAL5.*

<sup>9</sup> Этот размер был выбран для обеспечения совместимости технологии ATM с ранее появившейся технологией высокоскоростной коммутации данных SMDS (*Switched Multimegabit Data Service*), предлагавшейся телефонными компаниями США. Однако по согласованию сторон в сети ATM может использоваться другой максимальный размер дейтаграмм.

## 18.12. Тип пакета и мультиплексирование

Наблюдательный читатель, очевидно, заметил, что в формате трейлера протокола AAL5 не предусмотрено поле *типа* пакета. Следовательно, фрейм протокола ATM5 не является автоматически опознаваемым. Поэтому описанная выше упрощенная форма инкапсуляции не может использоваться в сети ATM, если расположенные по обоим концам канала узлы будут посыпать по одному виртуальному каналу несколько типов данных (например, пакеты, не принадлежащие протоколу IP). Поэтому в данном случае существует два варианта решения проблемы.

- Компьютеры на концах виртуального канала договариваются *заранее* о передаче трафика только определенного протокола (например, о пересылке только IP-дейтаграмм).
- Компьютеры на концах виртуального канала договариваются *заранее* о том, что некоторые октеты области данных будут использоваться в качестве поля типа.

Первый вариант, когда компьютеры договариваются об использовании в канале только одного из протоколов высшего уровня, имеет преимущество, поскольку в пакет не нужно помещать дополнительную информацию. Например, если компьютеры собираются передавать только IP-дейтаграммы, отправитель может пересыпать каждую дейтаграмму непосредственно модулю протокола AAL5 для дальнейшей ее передачи по сети ATM. В этом случае получателю необходимо переслать только саму дейтаграмму и созданный протоколом AAL5 трейлер. Главный недостаток подобной системы заключается в увеличении числа виртуальных каналов, поскольку компьютер должен создать отдельный виртуальный канал для каждого из используемых протоколов высшего уровня. Поскольку большинство операторов связи взимают плату за каждый виртуальный канал, потребители их услуг стараются избегать использования нескольких каналов, так как это увеличивает затраты.

Преимущество второго варианта, когда два компьютера используют один виртуальный канал для нескольких протоколов, заключается в том, что весь трафик проходит по одному и тому же каналу. Однако и в этом случае есть свои недостатки. Первый заключается в том, что каждый пакет должен содержать октеты, определяющие тип протокола. Второй недостаток состоит в том, что поступившие от всех протоколов пакеты передаются по сети с одинаковой задержкой и приоритетом.

В стандартах семейства протоколов TCP/IP предусмотрено, что все компьютеры могут выбирать по своему усмотрению один из двух методов использования протокола AAL5. И отправитель, и получатель должны заранее договориться о способе использования канала. Причем, иногда такая договоренность может также предполагать выполнение ручной настройки. Кроме того, в стандартах предусмотрено, что при включении в пакет информации о его типе компьютеры должны поместить в начало пакета заголовок *управления логическим соединением* (*Logical Link Control*, или *LLC*), используемый в стандарте IEEE 802.2, а следом за ним — заголовок, определяющий *точку подключения к подсети* (*SubNetwork Attachment Point*, или *SNAP*). На рис. 18.8 представлен заголовок LLC/SNAP, который присоединяется к дейтаграмме перед ее пересылкой по виртуальному каналу ATM.

Как показано на рис. 18.8, поле заголовка *LLC* состоит из трех октетов, которые содержат шестнадцатеричные значения AA-AA-03<sup>10</sup>. Заголовок *SNAP* состоит из пяти октетов: трех октетов, содержащих *的独特标识符*

<sup>10</sup> В такой форме записи шестнадцатеричные значения октетов отделены друг от друга точками.

организации (*Organizationally Unique Identifier, OUI*), и двух октетов, предназначенных для указания типа пакета<sup>11</sup>. Значение поля *OUI* определяет организацию, которая осуществляет контроль над значениями, определяющими *тип пакета*. При передаче IP-дейтаграмм в поле *OUI* помещается последовательность 00.00.00, которая определяет организацию, ответственную за стандарты Ethernet, а в поле *типа* — последовательность 08.00 (т.е. значение, которое используется для идентификации IP-дейтаграмм, помещенных во фрейм сети Ethernet). Таким образом, программное обеспечение, работающее на отсылающем дейтаграмму узле сети, должно присоединить заголовок LLC/SNAP к каждому пакету, а затем отослать его модулю протокола AAL5. Программные средства, запущенные на принимающем дейтаграмму узле сети, должны сначала проанализировать заголовок, чтобы определить, каким образом следует обрабатывать весь пакет.

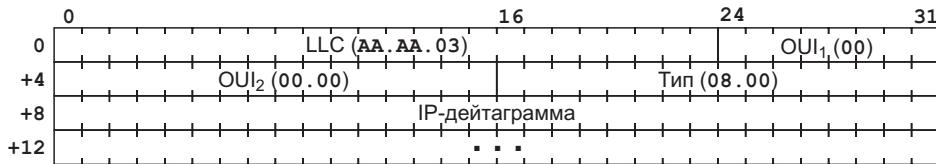


Рис. 18.8. Формат пакета, используемый для пересылки дейтаграммы по протоколу AAL5, при мультиплексировании нескольких пакетов по одному виртуальному каналу. Тип содержимого пакета (в данном случае это IP-дейтаграмма) определяет заголовок LLC/SNAP, длина которого составляет 8 октетов

### 18.13. Механизм привязки IP-адресов в сети ATM

Выше уже было продемонстрировано, что инкапсуляция дейтаграммы для передачи по сети ATM не представляет особой трудности. Однако привязка адресов в сетевом окружении с множественным доступом, не поддерживающим режим широковещания (*Non-Broadcast Multiple-Access*, или *NBMA*), может оказаться непростым делом. Подобно другим сетевым технологиям, в сети ATM каждому компьютеру присваивается физический адрес, который необходимо использовать при создании виртуального канала. С одной стороны, поскольку физический адрес узла ATM по размеру больше, чем IP-адрес, его нельзя закодировать в самом IP-адресе. Следовательно, в протоколе IP для сетей ATM нельзя использовать статическую привязку адресов. С другой стороны, сетевое оборудование ATM не поддерживает режим широковещания, поэтому для привязки адресов в сетях ATM нельзя использовать принятый в протоколе IP механизм ARP.

Постоянные виртуальные каналы сети ATM еще больше усложняют механизм привязки адресов. Поскольку администратор зачастую конфигурирует вручную каждый постоянный виртуальный канал, узлу сети известна лишь пара идентификаторов VPI/VCI. Более того, программное обеспечение узла сети может и не располагать информацией ни об IP-адресе, ни о физическом адресе ATM удаленного получателя. Следовательно, механизм привязки IP-адресов должен обеспечить идентификацию удаленного компьютера, подключенного через канал PVC, также возможность динамического создания каналов SVC, ведущих к известным получателям.

<sup>11</sup> Чтобы предотвратить ненужную фрагментацию, при вычислении размера MTU не учитываются восемь октетов заголовка LLC/SNAP (т.е. реальное значение MTU для ATM-соединений, в которых используется заголовок LLC/SNAP, составляет 9188 октетов).

Технологии коммутации, требующие установки соединения между получателями, еще больше усложняют механизм привязки адресов, поскольку при их использовании необходимо выполнить два уровня привязки. Во-первых, при создании виртуального канала для пересылки дейтаграмм IP-адрес получателя необходимо преобразовать в адрес конечной точки сети ATM. Адрес конечной точки используется для создания виртуального канала. Во-вторых, при отправке дейтаграммы удаленному компьютеру по уже существующему виртуальному каналу IP-адрес получателя необходимо преобразовать в пару идентификаторов VPI/VCI для этого канала. Второй уровень привязки используется каждый раз, когда дейтаграмма посыпается по сети ATM, в то время как первый уровень привязки необходим только при создании узлом сети канала SVC.

### 18.14. Понятие логической IP-подсети

Для общего решения проблемы привязки адресов в сетях NBMA (таких как ATM) не подходит ни один из существующих протоколов, а другие протоколы так и не были предложены. Однако все же был разработан один протокол, который можно использовать для выполнения ограниченной формы привязки. Речь идет о случае, когда группа компьютеров использует сеть ATM в качестве одной (зачастую локальной) физической сети. При этом говорят, что из группы компьютеров сформирована *логическая IP-подсеть (LIS)*. Интересно то, что из совокупности компьютеров, подключенных к одной сети ATM, можно сформировать несколько логических IP-подсетей. Например, на рис. 18.9 изображено восемь компьютеров, которые подключены к сети ATM, разделенной на две логических подсети.

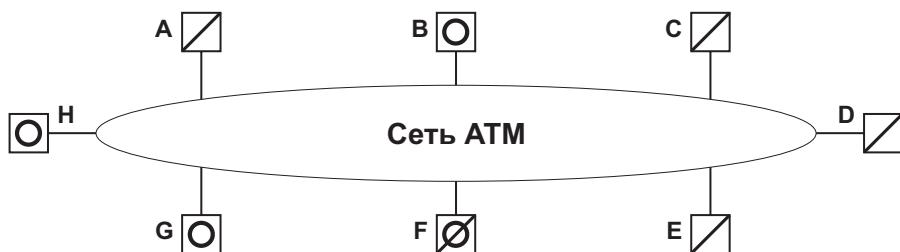


Рис. 18.9. Восемь компьютеров, подключенных к сети ATM, из которых сформировано две логических IP-подсети. Компьютеры, отмеченные косой чертой, относятся к одной логической подсети, а обозначенные кружочком — к другой

Как показано на рис. 18.9, все компьютеры подключены к одной физической сети ATM. Компьютеры A, C, D, E и F относятся к одной логической подсети, а компьютеры B, F, G и H — к другой. Каждая логическая IP-подсеть функционирует как отдельная локальная сеть. Компьютеры, относящиеся к одной подсети, устанавливают между собой виртуальные каналы для обмена дейтаграммами<sup>12</sup>. Поскольку теоретически каждая подсеть образует отдельную сеть, в ней можно использовать стандартные способы адресации протокола IP, такие же, как и в любой другой физической сети. Например, для всех компьютеров, относящихся к одной логической подсети назначается общий сетевой префикс IP-адреса. Причем значение этого префикса отличается от префиксов, используемых в других подсетях. Более того, хотя на разных компьютерах логической

<sup>12</sup> В стандарте указано, что в рамках IP-подсети должны использоваться заголовки LLC/SNAP.

подсети может использоваться разный (нестандартный) размер MTU, при установке виртуальных каналов между машинами логической подсети значение MTU должно быть одинаковым у всех. И наконец, узлу, принадлежащему к одной логической подсети, запрещается напрямую устанавливать соединение с узлом, расположенным в другой логической подсети, хотя сетевое оборудование ATM позволяет это сделать. Обмен информацией между двумя логическими подсетями должен происходить через маршрутизатор. Например, на рис. 18.9 машина F представляет собой IP-маршрутизатор, поскольку она относится к обеим логическим подсетям. Подведем итого всему сказанному выше.

*В семействе протоколов TCP/IP предусмотрено, что совокупность компьютеров, подключенных к одной сети ATM, может функционировать как независимая локальная сеть. Такая группа компьютеров называется логической IP-подсетью (LIS). Компьютерам, относящимся к одной логической подсети, назначается один сетевой префикс IP-адреса. Произвольный компьютер логической подсети может напрямую обмениваться информацией с любым другим компьютером, относящимся к этой же логической подсети. Однако, для обмена информацией с компьютером, относящимся к другой логической подсети должен использоваться маршрутизатор, хотя оборудование ATM позволяет это сделать напрямую.*

## 18.15. Управление соединениями

Узлы сети должны экономно использовать ресурсы, предоставляемые оборудованием ATM. Все дело в том, что для создания канала необходимо определенное время, и если речь идет о коммерческих службах ATM, это может повлечь за собой дополнительные денежные затраты. Следовательно упрощенный подход к проблеме создания виртуального канала, который предполагает закрытие канала сразу после того, как по нему была отправлена одна дейтаграмма, является дорогостоящим. Поэтому узел сети должен вести таблицу открытых каналов, которые можно было бы повторно использовать.

Виртуальные каналы управляются программами, выполняющими на уровне сетевого интерфейса (он расположен ниже уровня протокола IP). Если узлу сети нужно послать дейтаграмму, он использует средства обычной IP-маршрутизации для определения IP-адреса ( $N$ ) ближайшей точки перехода, а затем передает этот адрес вместе с самой дейтаграммой драйверу сетевого интерфейса. Драйвер сетевого интерфейса анализирует таблицу открытых виртуальных каналов и, если существует открытый канал, ведущий к узлу сети с адресом  $N$ , пересыпает дейтаграмму для отправки модулю протокола AAL5. В противном случае прежде чем отправить дейтаграмму, узел сети должен определить местонахождение компьютера с IP-адресом  $N$ , создать виртуальный канал и внести данные об этом канале в свою таблицу.

Использование логических IP-подсетей налагает определенные ограничения на процесс IP-маршрутизации. Напомним, что в правильно сконфигурированной таблице маршрутизации для каждого получателя должен быть указан IP-адрес ближайшей точки перехода, который принадлежит той же логической подсети, что и отправитель. Чтобы понять, в чем заключается суть ограничения, вспомним, что каждая логическая подсеть является полным аналогом одной локальной физической сети. Как известно, в таблице маршрутизации узла, подключенного к локальной сети, должны указываться адреса ближайших точек перехода, относящихся к этой же локальной сети. Другими словами, в таблице маршрутизации узла локальной сети указывается адрес подключенного к этой сети маршрутизатора.

Одна из причин разделения компьютеров на логические подсети заключается в ограничениях, налагаемых аппаратным и программным обеспечением. Узел сети не может одновременно поддерживать неограничено большое количество открытых виртуальных каналов, поскольку создание каждого канала требует выделения дополнительных ресурсов как сетевого оборудования ATM, так и операционной системы компьютера. Разделение компьютеров на логические подсети ограничивает максимальное количество одновременно открытых каналов и сводит его к количеству компьютеров в данной подсети LIS.

## 18.16. Привязка адресов в логической подсети

Создавая виртуальный канал к компьютеру своей логической подсети, узел сети должен указать физический адрес получателя в сети ATM. Как узел сети преобразует адрес ближайшей точки перехода в соответствующий физический адрес ATM? Узел сети не может широковещательно отослать запрос всем компьютерам логической подсети, поскольку режим ATM не поддерживает широковещание на уровне сетевого оборудования. Для преобразования адресов узел сети связывается со специальным сервером. Для обмена информацией между узлом сети и сервером используется протокол *ATMARP*, который представляет собой разновидность описанного в главе 5, “Преобразование IP-адресов в физические адреса (ARP)”, протокола ARP.

Как и при использовании стандартного протокола ARP, отправитель создает запрос, в который помещает свой IP-адрес и физический адрес ATM, а также IP-адрес получателя, для которого необходимо узнать физический адрес ATM. Затем отправитель передает запрос *серверу ATMARP* своей логической подсети. Если серверу известен физический адрес ATM, он отсылает ответное *ATMARP-сообщение*. В противном случае, сервер отсылает *отрицательный ATMARP-ответ*.

## 18.17. Формат пакета протокола ATMARP

На рис. 18.10 представлен формат ATMARP-пакета. Как следует из рисунка, он незначительно отличается от формата пакета протокола ARP. Основное отличие заключается в дополнительных полях, содержащих длину физических адресов ATM. Таким образом, в ATMARP-пакет можно поместить физический адрес ATM произвольной длины. Чтобы понять суть этих изменений, необходимо иметь в виду, что для использования в сети ATM было предложено нескольких форм физических адресов, однако пока ни одну из них не утвердили в качестве действующего стандарта. Телефонные компании, предоставляющие услуги общего доступа к сети ATM, в качестве физического адреса используют телефонные номера абонентов ISDN, длина которых составляет 8 октетов. Формат телефонных номеров абонентов ISDN описан в документе *E.164*, изданного международным телекоммуникационным союзом (ITU). Для сравнения: ATM Forum<sup>13</sup> разрешает присваивать каждому подключенному к частной сети ATM компьютеру физические адреса, состоящие из 20 октетов, которые являются адресами *точек доступа к сетевой службе* (*Network Service Access Point*, или *NSAP*). Поэтому на концах ATM-соединения могут использоваться физические адреса, заданные в разных форматах. Например, для узла, подключенного к локальному коммутатору, может использоваться физический адрес NSAP, тогда как для удаленного узла — физический адрес, формат которого определен в документе E.164.

---

<sup>13</sup> ATM Forum — это консорциум производителей, рекомендующих стандарты для частных сетей ATM.

	0	8	16	24	31
	Тип сетевого оборудования (0x0013)		Тип протокола (0x0800)		
+4	HLEN отпр. (20)	HLEN2 отпр. (0)		Тип операции	
+8	PLEN отпр. (4)	HLEN цели (20)	HLEN2 цели (0)	PLEN цели (4)	
+12			Физический ATM-адрес отправителя (октеты 0-3)		
+16			Физический ATM-адрес отправителя (октеты 4-7)		
+20			Физический ATM-адрес отправителя (октеты 8-11)		
+24			Физический ATM-адрес отправителя (октеты 12-15)		
+28			Физический ATM-адрес отправителя (октеты 16-19)		
+32			Протокольный адрес отправителя		
+36			Физический ATM-адрес целевого компьютера (октеты 0-3)		
+40			Физический ATM-адрес целевого компьютера (октеты 4-7)		
+44			Физический ATM-адрес целевого компьютера (октеты 8-11)		
+48			Физический ATM-адрес целевого компьютера (октеты 12-15)		
+52			Физический ATM-адрес целевого компьютера (октеты 16-19)		
+56			Протокольный адрес целевого компьютера		

Рис. 18.10. Формат ATMARP-пакета при использовании рекомендованных ATM Forum физических адресов, длина которых составляет 20 октетов

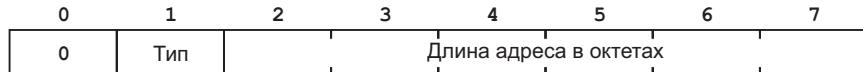
Для возможности использования разных форматов физических адресов на обоих концах ATM-соединения в ATMARP-пакете предусмотрено по два поля длины для каждого ATM-адреса, а также по одному полю длины для каждого протокольного адреса. Как показано на рис. 18.10, ATMARP-пакет начинается с полей фиксированного размера, определяющих формат и длину последующих полей. Формат двух первых полей совпадает с форматом, принятым в обычном протоколе ARP. Для сети ATM в поле *типа сетевого оборудования* содержится шестнадцатеричное значение 0x0013, а в поле *типа протокола* — значение 0x0800, что соответствует протоколу IP.

Поскольку форматы физических адресов отправителя и целевого компьютера могут отличаться, в ATMARP-пакете должна быть указана длина этих адресов. Поле *HLEN отпр.* определяет длину физического ATM-адреса отправителя, а поле *HLEN2 отпр.* — длину его подадреса. Поля *HLEN цели* и *HLEN2 цели* определяют длину физического ATM-адреса целевого компьютера и длину его подадреса. И наконец, поля *PLEN отпр.* и *PLEN цели* определяют длины протокольных адресов отправителя и целевого компьютера.

После описанных выше полей длины физических адресов в ATMARP-пакете следуют шесть адресов. Первые три адреса соответствуют физическому ATM-адресу отправителя, его подадресу и протокольному адресу. В последних трех адресах указывается физический ATM-адрес целевого компьютера, его подадрес и протокольный адрес. В показанном на рис. 18.10 формате ATMARP-пакета поля длины подадреса отправителя и целевого компьютера содержат нулевые значения, поэтому эти подадреса отсутствуют в пакете.

### 18.17.1. Формат полей длины в адресе ATM

Поскольку протокол ATMARP предназначен для использования с физическими ATM-адресами двух типов (Е.164 или NSAP; длина последнего составляет 20 октетов), в поля, содержащие длину ATM-адреса, входит бит, определяющий формат адреса. На рис. 18.11 изображено, каким образом в протоколе ATMARP в 8-битовом поле кодируется тип адреса и его длина.



*Рис. 18.11. Кодирование типа физического ATM-адреса и его длины в 8-битовом поле. Два типа ATM-адресов отличаются между собой значением первого бита*

Тип ATM-адреса кодируется значением только одного бита, поскольку в данном случае возможны только два варианта. Если первый бит содержит нулевое значение, адрес представлен в рекомендованном ATM Forum формате NSAP. Если этот бит содержит единицу, адрес представлен в рекомендованном союзом ITU формате E.164. Поскольку в каждый ATMARP-пакет входит два поля длины физического адреса, формат которого показан на рис. 18.11, в одном пакете могут находиться физические ATM-адреса, представленные в разных форматах.

### 18.17.2. Типы операций протокола ATMARP

Изображенный на рис. 18.10 формат пакета используется для создания запроса на привязку адресов, ответного сообщения на запрос или запроса на обратную привязку адресов. Отсылая ATMARP-пакет, компьютер должен поместить в поле *типа операции* специальный код, чтобы определить тип привязки. В табл. 18.1 приведены значения, которые могут использоваться в поле *типа операции*, и описание каждого из них. Их значения были выбраны в соответствии с кодами операций, используемыми в стандартном протоколе ARP. В оставшихся разделах этой главы объясняется, каким образом функционирует протокол ATMARP.

**Таблица 18.1. Коды типа операции протокола ATMARP**

<i>Код</i>	<i>Описание</i>
1	Запрос протокола ATMARP
2	Ответ протокола ATMARP
8	Запрос инверсного протокола ATMARP
9	Ответ инверсного протокола ATMARP
10	Сигнал отрицательного уведомления протокола ATMARP

## 18.18. Использование ATMARP-пакетов для определения физических адресов

Принцип выполнения привязки адресов в сетевом оборудовании, требующем установки соединения между получателями, несколько сложнее, по сравнению с сетевым оборудованием, не требующем установки такого соединения. Поскольку сетевое оборудование ATM поддерживает два вида виртуальных каналов, возможны два случая. Сначала мы рассмотрим случай использования постоянных виртуальных каналов, а затем — коммутируемых виртуальных каналов.

### 18.18.1. Постоянные виртуальные каналы

Чтобы понять проблемы, возникающие при использовании каналов PVC, вспомним, как функционирует сетевое оборудование сети ATM. Сетевой администратор должен сконфигурировать каждый канал PVC. При этом сами узлы сети не участвуют в настройке параметров каналов PVC. В частности, узел сети начинает

передавать данные только после того, как канал PVC уже установлен. При этом он не получает никакой информации от сетевого оборудования об адресе удаленного абонента. Таким образом, если адресная информация не была задана при настройке программного обеспечения узла сети (например, сохранена в файле на жестком диске), узлу сети не известен ни IP-адрес, ни физический ATM-адрес компьютера, к которому ведет канал PVC.

Инверсный протокол *ATMARP* (*InATMARP*) решает проблему поиска ATM-адресов при использовании каналов PVC. Для работы этого протокола компьютер должен располагать информацией о всех сконфигурированных постоянных виртуальных каналах. Чтобы определить IP-адрес и ATM-адрес удаленного абонента, компьютер отсылает запрос инверсного протокола ATMARP, где в поле типа *операции* задано значение 8. Получив по каналу PVC такой запрос, удаленный компьютер создает ответное сообщение инверсного протокола ATMARP, где в поле типа *операции* установлено значение 9. И в запросе, и в ответе указываются IP-адрес и ATM-адрес отправителя. В результате компьютер на одном конце соединения получает информацию об адресной привязке компьютера другом конце. Подведем итог.

*Два компьютера, которые обмениваются информацией по постоянному виртуальному каналу, используют инверсный протокол ATMARP для определения IP-адреса и ATM-адреса друг друга. Один из компьютеров посыпает запрос инверсного протокола ATMARP, на который присыпает ответ другой компьютер.*

### 18.18.2. Коммутируемые виртуальные каналы

В пределах одной логической подсети компьютеры создают коммутируемые виртуальные каналы при возникновении потребности. Если компьютеру *A* необходимо послать дейтаграмму компьютеру *B*, но в этот момент не существует канала к компьютеру *B*, компьютер *A*, используя сигнальную систему, создает его. Следовательно, компьютеру *A* должен быть заранее известен IP-адрес компьютера *B*, который он преобразовывает в эквивалентный ему ATM-адрес. Выше уже упоминалось, что в каждой логической подсети должен быть запущен специальный ATMARP-сервер. При этом информация о способе достижимости этого сервера должна быть заранее известна всем компьютерам логической подсети (обычно она задается в процессе начального конфигурирования узла сети). Например, компьютер может иметь ведущий к серверу канал PVC или адрес сервера ATMARP может быть сохранен у него на диске. Сервер не устанавливает соединений с другими компьютерами. Он только ожидает, пока компьютеры логической подсети установят с ним связь. Чтобы преобразовать IP-адрес компьютера *B* в физический адрес сети ATM, компьютер *A* должен открыть виртуальный канал для связи с ATMARP-сервером своей логической подсети. После этого компьютер *A* создает ATMARP-запрос и отсылает его по каналу связи серверу. В поле *типа операции* в отсылаемом пакете помещается значение 1, а в поле протокольного адреса целевого компьютера — IP-адрес компьютера *B*.

На ATMARP-сервере должна иметься база данных соответствия IP-адресов и физических ATM-адресов. Если серверу известен ATM-адрес компьютера *B*, он функционирует подобно программе-агенту протокола ARP (Proxy ARP). Сервер создает ответное сообщение протокола ATMARP, помещает в поле *типа операции* значение 2 и заполняет поле физического ATM-адреса, который соответствует IP-адресу целевого компьютера. Как и в обычном протоколе ARP, сервер меняет местами адреса отправителя и целевого компьютера, прежде чем ответить пославшему запрос компьютеру.

Если серверу не известен ATM-адрес, который соответствует IP-адресу целевого компьютера, действия, предпринимаемые программой протокола ATMARP, отличаются от действий обычного протокола ARP. Сервер отсылает в ответ отрицательное уведомление (пакет протокола ATMARP, в котором в поле *типа операции* находится значение 10). Напомним, что в подобном случае в протоколе ARP поступивший запрос попросту игнорируется. Факт получения узлом сети отрицательного уведомления, означает, что ATMARP-сервер исправно функционирует, но пока не располагает данными о запрашиваемой адресной привязке (т.е. указанный целевой компьютер в настоящее время отключен от логической подсети). Таким образом, с помощью запроса к ATMARP-серверу узел сети может однозначно определить три вещи:

- физический ATM-адрес целевого компьютера;
- факт отключения целевого компьютера от логической подсети;
- факт неработоспособности ATMARP-сервера.

## 18.19. Сбор информации для базы данных ATMARP-сервера

Сбор и обработка информации об адресных привязках выполняется ATMARP-сервером автоматически. Для этого используется инверсный протокол ATMARP. Каждый раз, когда узел сети или маршрутизатор открывает виртуальный канал к ATMARP-серверу, последний сразу же отсылает ему запрос инверсного протокола ATMARP<sup>14</sup>. Узел сети или маршрутизатор отвечает на запрос отправкой пакета инверсного протокола ATMARP (см. табл. 18.1). Получив ответный пакет, сервер извлекает из него IP- и ATM-адрес отправителя и сохраняет эту адресную привязку в своей базе данных. Таким образом, сразу после подключения к логической сети ATM, каждый компьютер должен установить соединение с ATMARP-сервером, даже если ему не нужна информация об адресной привязке. Подведем итоги.

*Каждый узел или маршрутизатор логической подсети должен зарегистрировать свой IP-адрес и соответствующий ему физический ATM-адрес на ATMARP-сервере, обслуживающем эту подсеть. Регистрация происходит автоматически при создании виртуального канала с ATMARP-сервером. При этом сервер отсылает узлу сети или маршрутизатору запрос инверсного протокола ATMARP, на которой он должен получить ответ.*

## 18.20. Достоверность информации, хранящейся на ATMARP-сервере

Как и в случае обычного протокола ARP, адресные привязки, хранящиеся на ATMARP-сервере, могут через определенное время устареть и должны быть удалены с сервера. Как долго информация об адресной привязке должна находиться на сервере? По умолчанию данные на ATMARP-сервере хранятся максимум 20 минут после регистрации их узлом сети. По истечении 20 минут сервер анализирует содержимое записи и состояние канала связи. Если к этому моменту открытый канал связи с компьютером, приславшим адресную привязку, уже не

---

<sup>14</sup> В канале должен использоваться протокол AAL5 и заголовки идентификации типа LLC/SNAP.

существует, она удаляется с ATMARP-сервера<sup>15</sup>. Если отославший данные компьютер по прежнему имеет открытый канал к серверу, он пытается подтвердить достоверность данных. Сервер снова отсылает запрос инверсного протокола ATMARP и ожидает ответа. Если ответное сообщение подтверждает информацию, содержащуюся в базе данных, сервер снова устанавливает таймер и выжидает следующие 20 мин. Если данные, содержащиеся в ответном сообщении инверсного протокола ATMARP, не совпадают с информацией на сервере, он закрывает канал и удаляет элемент из базы данных.

Для уменьшения объема передаваемого трафика в стандарте протокола ATMARP предусмотрена определенная оптимизация. Узлу сети разрешено использовать один виртуальный канал для выполнения обмена любыми данными с ATMARP-сервером. В отсылаемом узлом сети запросе ATMARP-серверу находится адресная привязка для отправителя. Сервер может извлечь ее и проверить достоверность информации, которая у него храниться. Таким образом, если узел сети посыпает ATMARP-серверу запросы чаще, чем раз в 20 мин, то серверу не нужно посыпать узлу сети запросы инверсного протокола ATMARP.

## 18.21. Достоверность информации, хранящейся на узле и маршрутизаторе

Узлы сети и маршрутизаторы также проверяют по тайм-ауту достоверность информации, полученной от ATMARP-сервера. В частности, в стандарте определено, что компьютер может хранить адресную привязку, полученную от ATMARP-сервера, максимум 15 минут. По истечении 15 минут данные о привязке необходимо проверить и, если они недействительны, удалить. Если после истечения тайм-аута узел сети не имеет открытого виртуального канала с получателем, то адресная привязка удаляется из кэш-памяти протокола ATMARP. Если узел сети имеет открытый виртуальный канал к получателю, то по истечении тайм-аута он должен проверить достоверность адресной привязки. Истечение срока действия адресной привязки может вызвать задержку при передаче сетевого трафика. Причина в том, что

*узел сети или маршрутизатор должен прекратить на время отсылку данных к тем получателям, для которых истек срок действия адресной привязки. Возобновление передачи данных возможно только после проверки достоверности адресной привязки.*

Метод, применяемый узлом сети для проверки достоверности адресной привязки, зависит от типа используемого виртуального канала. Если с получателем установлен постоянный виртуальный канал (PVC), отправитель отсылает по нему запрос инверсного протокола ATMARP и ожидает на него ответа. Если к получателю ведет коммутируемый виртуальный канал, то запрос протокола ATMARP посыпается ATMARP-серверу.

## 18.22. Технологии коммутации IP-пакетов

В предыдущих разделах этой главы на примере протокола ATM была описана сетевая технология, требующая установки соединения между получателями, которая может использоваться для передачи IP-дейтаграмм. Однако инженеры провели фундаментальное исследование возможности объединения двух технологий. Суть

---

<sup>15</sup> На практике сервер не удаляет автоматически данные сразу при закрытии канала, а выжидает некоторое время, заданное тайм-аутом.

проблемы заключалась в следующем: можно ли повысить скорость пересылки IP-трафика за счет использования быстродействующего коммутационного сетевого оборудования? Было высказано предположение, что сетевое оборудование может коммутировать больше пакетов в единицу времени, чем маршрутизировать их. И если это предположение правильно, то имело прямой смысл исследовать упомянутую выше проблему, поскольку производители маршрутизаторов постоянно пытались найти способы повышения производительности и расширяемости своих устройств.

Корпорация Ipsilon была одной из первых компаний, которая начала выпускать сетевое оборудование для коммутации IP-трафика. В этих устройствах использовался режим ATM, новая технология была названа *IP-коммутацией* (*IP-switching*), а сами устройства — *IP-коммутаторами*. Кроме корпорации Ipsilon, исследования в этой области проводили и другие компании. В результате были предложены новые технологии и новая терминология, среди которых можно выделить *коммутацию тэгов* (*tag switching*), *коммутацию третьего уровня* (*layer 3 switching*) и *коммутацию меток* (*label switching*). Возникшие идеи были объединены в один стандарт, разработанный при поддержке проблемной группы IETF и названный *многопротокольной коммутацией меток* (*Multi-Protocol Label Switching*, или *MPLS*)<sup>16</sup>. Все, участвовавшие в разработке открытого стандарта, надеются, что он позволит взаимодействовать продуктам различных производителей.

## 18.23. Принцип работы коммутаторов

Как же работают устройства коммутации IP-пакетов? На этот вопрос существует два общих ответа. Все созданные ранее технологии коммутации пакетов были ориентированы на работу в стандартной сети NBMA (как правило, сети ATM). Цель использования таких технологий заключалась в оптимизации структуры IP-маршрутизации путем пересылки дейтаграмм по магистральным сетям ATM там, где это возможно. Кроме возможных путей оптимизации маршрутов, в последующих разработках предполагалась модификация коммутационного сетевого оборудования для его оптимального использования при передаче IP-трафика. В частности, было предложено два способа оптимизации. Во-первых, если коммутационное сетевое оборудование удастся перепроектировать таким образом, чтобы в нем можно было использовать либо большие по размеру ячейки, либо фреймы переменной длины, то потери при передаче заголовков уменьшатся<sup>17</sup>. Во-вторых, если сетевое оборудование построить таким образом, чтобы оно могло анализировать заголовки IP-пакетов и извлекать из них нужные поля, то входящую дейтаграмму можно будет быстрее пересылать по сети.

В основе технологии коммутации меток лежит принцип пересылки пакетов (*packet forwarding*). При этом необходимо учитывать следующие три момента. Во-первых, на уровне протокола IP устройство пересылки пакетов должно функционировать как обычный IP-маршрутизатор, пересылающий дейтаграммы между локальной сетью и коммутационным центром (*switching fabric*). Таким образом, устройство пересылки пакетов должно располагать информацией об удаленных получателях и преобразовывать IP-адрес получателя в адрес ближайшей точки перехода. Во-вторых, на уровне сетевого интерфейса устройство пересылки

<sup>16</sup> Хотя в названии стандарта фигурирует слово “многопротокольный”, технология MPLS направлена исключительно на поиск путей передачи IP-трафика по сетям с множественным доступом, не поддерживающим режим широковещания (NBMA).

<sup>17</sup> В сетевой индустрии потери при передаче заголовков ячеек в сетях ATM называют *издержками* (*cell tax*).

пакетов должно создавать виртуальные соединения и управлять ими посредством коммутационного центра (т.е. оно должно преобразовывать IP-адреса в аппаратные адреса и создавать при необходимости каналы SVC). В-третьих, устройство пересылки пакетов должно оптимизировать маршруты, проходящие по коммуникационному центру.

## 18.24. Оптимизация пересылки IP-пакетов

Для оптимизации пересылки пакетов выполняется высокоскоростная классификация и определяются *укороченные маршруты* (*shortcut paths*). Чтобы понять, что собой представляют укороченные маршруты, представим себе три коммутатора  $S_1$ ,  $S_2$  и  $S_3$  и предположим, что для достижения некоторого получателя в таблицах IP-маршрутизации этих устройств определена следующая схема пересылки пакетов: от коммутатора  $S_1$  пакет пересыпается коммутатору  $S_2$ , пересылающему его коммутатору  $S_3$ , который и доставляет пакет получателю. Предположим также, что все три коммутатора подключены к одному коммуникационному центру. Если коммутатор  $S_1$  обнаруживает, что некоторому получателю отсылается слишком большое количество дейтаграмм, он может оптимизировать маршрут, обойдя коммутатор  $S_2$  и установив укороченный маршрут (т.е. виртуальный канал) непосредственно к коммутатору  $S_3$ . Естественно, при этом нужно учесть множество деталей. Например, хотя в приведенном выше примере используется только три устройства, на самом деле в реально действующей сети их может быть гораздо больше. Определив маршрут, по которому дейтаграмма будет следовать к получателю, коммутатор  $S_1$  должен узнать адрес последней точки перехода, расположенной на этом маршруте, доступ к которой можно получить через коммутируемую сеть. Затем коммутатор должен преобразовать IP-адрес этой точки перехода в аппаратный адрес и создать с ней виртуальное соединение. Установить, подключена ли последняя точка перехода к тому же коммуникационному центру, и преобразовать адреса — задача не из легких. Для ее решения нужны усложненные протоколы, с помощью которых будет передаваться необходимая информация. Чтобы на уровне протокола IP создать видимость прохождения дейтаграммами заранее определенных маршрутов, один из коммутаторов, либо  $S_1$ , либо  $S_3$ , должен учитывать обойденный маршрутизатор при уменьшении значения поля времени жизни в заголовке дейтаграммы. Кроме того, коммутатор  $S_1$  должен продолжать получать сообщения об обновлении маршрутов от коммутатора  $S_2$ , чтобы можно было в случае изменения маршрутной информации вернуться к ранее используемому маршруту.

## 18.25. Классификация, информационные потоки и коммутация на верхнем уровне

*Механизм классификации* анализирует содержимое каждой входящей дейтаграммы и выбирает соединение, по которому она должна следовать. Реализация этого механизма на аппаратном уровне позволяет расширить возможности технологии и существенно повысить скорость работы коммутатора. В большинстве предложенных реализаций используется двухуровневый иерархический подход. Сначала в процессе классификации коммутатор относит дейтаграмму к одному из существующих *потоков* (*flows*), после чего выбранный поток отображается на одно из открытых соединений. Математически такое преобразование можно представить в виде двух функций:

$$f = c_1 \text{ (дейтаграмма)}, \text{ и} \\ vc = c_2 (f)$$

В этих функциях  $f$  обозначает определенный информационный поток, а  $vc$  — соединение. Далее будет показано, что использование двух отдельных функций обеспечивает определенную гибкость при выполнении возможных отображений.

На практике в функции  $c_1$  анализируется содержимое не всей дейтаграммы, а только полей ее заголовка. В строгой классификации на уровне 3 при вычислении функции  $c_1$  используются такие поля заголовка IP-дейтаграммы, как IP-адреса отправителя и получателя, а также тип обслуживания. Большинство производителей реализуют в своих продуктах классификацию на уровне 4<sup>18</sup>, а некоторые — на уровне 5. Помимо анализа полей заголовка IP-дейтаграммы, в системах классификации на уровне 4 также анализируются номера портов протоколов, указанных в заголовке TCP- или UDP-дейтаграммы. В системах классификации на уровне 5 выполняется углубленный анализ содержимого дейтаграммы с учетом создавшего ее приложения.

Понятие информационного потока играет важную роль в процессе коммутации IP-трафика, поскольку оно позволяет устройству коммутации отслеживать интенсивность передачи трафика в заданном направлении. В качестве примера представим, что в процессе обработки дейтаграмм коммутатором составляется список пар (отправитель, получатель) и для каждой пары ведется отдельный счетчик ее использования. Очевидно, что для коммутатора нет смысла оптимизировать все маршруты, поскольку по некоторым информационным потокам передается всего несколько пакетов (например, как при проверке доступности удаленного компьютера с помощью программы ping). На основании подсчета количества пакетов в потоке можно сформировать критерий оптимизации: если значение счетчика достигнет пороговой величины, коммутатор должен начать поиск оптимизированного маршрута. Классификация на уровне 4 дает возможность оптимизировать информационные потоки, поскольку коммутатор располагает информацией о приблизительной продолжительности существования соединения, а также о том, вызван ли трафик одним или несколькими TCP-соединениями.

Информационные потоки также являются важным средством, обеспечивающим эффективную работу устройств коммутации при передаче трафика протокола TCP. Переход коммутатора в процессе передачи TCP-трафика на использование укороченного маршрута приведет к изменению полного времени доставки пакета. В результате порядок доставки некоторых сегментов может быть нарушен. Это приведет к изменению значения величины тайм-аута повторной передачи протокола TCP. Таким образом, коммутатор, использующий классификацию на уровне 4, может преобразовывать каждый сеанс связи протокола TCP в отдельный поток, а затем решить, передавать ли его по исходному или укороченному маршруту. В большинстве коммутационных технологий применяется принцип запаздывания, который заключается в том, что трафик существующих TCP-соединений передается по исходному маршруту, а для новых соединений используется укороченный маршрут. При этом переход на укороченный маршрут для существующих соединений происходит только после истечения фиксированного интервала времени или если данное соединение не используется.

## 18.26. Распространение технологии коммутации

Хотя устройства коммутации IP-трафика выпускаются многими производителями, используемая в них технология не получила такого широкого распространения, как ожидалось. Это объясняется некоторыми причинами. Во-первых, в большинстве случаев коммутация оказывается существенно дороже обычной

---

<sup>18</sup> При описании продуктов, в которых реализована классификация на уровне 4, производители часто используют термин *коммутация на уровне 4*.

маршрутизации, а обеспечиваемый при этом прирост производительности не соответствует затратам. Причем эта разница особенно заметна в локальном сетевом окружении, где существуют недорогие технологии типа Ethernet, обладающие достаточной пропускной способностью и использующие недорогое устройство маршрутизации. Не стоит забывать, что процесс маршрутизации IP-дейтаграмм также не стоит на месте. Инженеры продолжают искать пути улучшения систем пересылки пакетов. Это означает, что производительность обычных устройств маршрутизации может быть существенно повышена без увеличения скорости работы сетевого оборудования. Во-вторых, при создании корпоративных сетей передачи данных недорогие высокоскоростные технологии локальных сетей, такие как Gigabit Ethernet, вытеснили более дорогостоящие технологии, требующие установки соединения между получателями,. В-третьих, хотя процесс коммутации IP-трафика на первый взгляд кажется достаточно простым, его реализацию усложняют различные детали. Следовательно, протоколы коммутации намного сложнее протокола IP, что резко усложняет процесс создания программного обеспечения, а также его последующую установку, настройку и сопровождение. Можно сделать вывод, что, несмотря на очевидные преимущества технологии коммутации IP-трафика, она не заменит традиционные методы маршрутизации.

## 18.27. Резюме

Протокол IP может использоваться совместно с сетевыми технологиями, требующими установки соединения между получателями. В качестве примера такой технологии была рассмотрена ATM, которая представляет собой высокоскоростную сетевую технологию, построенную на основе одного или нескольких коммутаторов, соединенных между собой и образующих коммутационный центр. Полученную в результате систему можно охарактеризовать как сеть с множественным доступом (поскольку она функционирует как единая большая сеть), которая обеспечивает обмен информацией между двумя подключенными к ней компьютерами, но не позволяет выполнять широковещательную рассылку одного пакета всем подключенными к ней компьютерам.

Поскольку в основе сети ATM лежит технология, требующая установки соединения между получателями, два компьютера должны установить через коммутационный центр виртуальный канал, прежде чем смогут обменяться данными. Существует два типа виртуальных каналов — постоянный и коммутируемый. Коммутируемые каналы создаются автоматически сетевым оборудованием по мере необходимости, тогда как для создания постоянных каналов обычно требуется ручная конфигурация коммутаторов. В обоих случаях в сети ATM каждому открытому каналу присваивается целочисленный идентификатор. В каждом отсылаемом узлом сети фрейме, как и в доставляемых по сети ATM фреймах, содержится идентификатор виртуального канала. Обратите внимание, что в заголовке фрейма не указываются адреса отправителя и получателя.

Хотя на самом низком уровне в сети ATM для передачи информации используются ячейки размером 53 октета, модуль протокола IP всегда взаимодействует с модулем протокола адаптации ATM уровня 5 (AAL5). Модуль протокола AAL5 принимает от отправителей и доставляет получателям блоки данных переменного размера, причем размер каждого блока не может превышать 65535 октетов. Чтобы отослать IP-дейтаграмму по сети ATM, отправитель должен создать виртуальный канал связи с получателем, указать, что в этом канале в качестве адаптационного используется протокол AAL5, и пересыпать дейтаграмму модулю этого протокола в виде одного блока данных. В модуле протокола AAL5 к каждой дейтаграмме добавляется трейлер, после чего она разбивается на ячейки фиксированной длины (53 октета) для передачи по сети. На машине получателя

модуль протокола AAL5 из поступивших ячеек собирает дейтаграмму и передает ее модулю протокола IP операционной системы. В протоколе IP для сетей ATM используется стандартный размер MTU, равный 9180 октетов. Это позволяет модулю протокола AAL5 разделить дейтаграмму на ячейки, избежав ненужной фрагментации.

Совокупность компьютеров, использующих сеть ATM в качестве локальной сети, образуют логическую IP-подсеть. Компьютеры, входящие в одну логическую подсеть, создают между собой виртуальные каналы и обмениваются по ним дейтаграммами. Поскольку в сети ATM не поддерживается режим широковещательной передачи данных, для преобразования IP-адресов в физические адреса используется модифицированная форма протокола ARP, которая называется ATMAPR. Суть ее состоит в том, что все адресные привязки в пределах одной логической подсети, хранятся на специальном ATMAPR-сервере. Перед началом работы каждый компьютер логической подсети должен зарегистрироваться на сервере, сообщив ему свой IP-адрес и физический адрес ATM. Как и в обычном протоколе ARP, информация об адресных привязках, хранящаяся на ATMAPR-сервере со временем устаревает. Поэтому по истечении времени жизни адресную привязку необходимо обновить или удалить. Кроме протокола ATMAPR, существует также инверсный протокол ATMAPR, который используется для нахождения физического адреса ATM и IP-адреса удаленного компьютера, подключенного с помощью постоянного виртуального канала.

Для коммутации IP-трафика были разработаны специальные устройства, которые называются IP-коммутаторами. Они выполняют роль маршрутизаторов, а также классифицируют IP-дейтаграммы и, если возможно, отсылают их по коммутируемой сети. В случае классификации на уровне 3 используется только информация в заголовке дейтаграммы. При классификации на уровне 4 выполняется также анализ заголовка TCP- или UDP-пакета. Для коммутации IP-трафика был разработан новый стандарт многопротокольной коммутации меток (MPLS), благодаря которому обеспечивается взаимодействие между системами, созданными разными производителями.

## Материал для дальнейшего изучения

Принципы IP-коммутации описаны в статье Ньюмана (Newman) и др. [94]. Лойбах (Laubach) и Холперн (Halpern) в [RFC 2225] вводят понятие логической IP-подсети, описывают протокол ATMAPR и обсуждают стандартный размер MTU. Гроссман (Grossman) и Хейнанен (Heinanen) в [RFC 2684] описывают использование заголовков LLC/SNAP при инкапсуляции IP-дейтаграмм в дейтаграммы протокола AAL5.

Партидж (Partridge) в работе [98] в общих чертах описывает организацию гигабитовых сетей передачи данных и, в частности, рассказывает о важности процесса коммутации ячеек. Де Прудкер (De Prycker) в книге [42] рассматривает теоретические предпосылки функционирования сети ATM и обсуждает ее связь с телефонными сетями.

## Упражнения

- 18.1. Если в вашей организации есть коммутатор ATM либо вы имеете доступ к службе сети ATM, изучите их технико-экономические данные, а затем сравните стоимость эксплуатации сети ATM и другой сетевой технологии, например Ethernet.
- 18.2. Как правило, скорость соединения между узлом сети и локальным коммутатором ATM составляет 155 Мбит/с. Определите пропускную

способность шины вашего компьютера. На сколько процентов должна быть загружена шина, чтобы интерфейс ATM работал на полную мощность?

- 18.3. Во многих операционных системах размер буфера протокола TCP выбирается кратным 8К октетам (1К октетов равен 1024 октетам). Если в модуле протокола IP выполняется фрагментация дейтаграмм согласно размеру MTU, равного 9180 октетам, фрагменты какого размера получатся в результате разделения дейтаграммы, в которой передается TCP-сегмент размером 16К октетов? 24К октетов?
- 18.4. Ознакомьтесь с описанием протокола IP версии 6 (IPv6), приведенным в главе 33, “Будущее протокола TCP/IP (IPv6)”. Какой из новых механизмов непосредственно относится к сети ATM?
- 18.5. Сеть ATM не является системой, гарантирующей доставку дейтаграмм. Это связано с тем, что сетевое оборудование может отвергать ячейки в случае перегрузки сети. Какова вероятность потери дейтаграммы, если вероятность потери одной ячейки составляет  $1/P$ , а дейтаграмма имеет длину 576 октетов? 1500 октетов? 4500 октетов? 9180 октетов?
- 18.6. Во время типичного сеанса связи с удаленным терминалом по протоколу TCP, узлы сети обмениваются дейтаграммами размером 41 октет. Из них 20 октетов приходится на заголовок IP-дейтаграммы, 20 — на заголовок TCP-сегмента и 1 октет — на данные. Сколько ячеек ATM необходимо для отсылки такой дейтаграммы при использовании стандартного типа инкапсуляции на уровне протокола AAL5?
- 18.7. Какое количество ячеек, октетов и битов может одновременно находиться в оптоволоконном кабеле, подключенном к коммутатору ATM, если кабель имеет длину 3 метра? 100 метров? 3000 метров? Чтобы найти ответ, рассмотрите коммутатор ATM, передающий данные со скоростью 155 Мбит/с. Каждый бит передается со скоростью света в виде светового импульса, продолжительностью  $1/(155 \times 10^6)$  с. Вычислите длину импульса и сравните его с длиной кабеля.
- 18.8. Узел сети при отсылке запроса на выделение канала SVC может указывать двухуровневый физический адрес ATM. В каких сетевых топологиях ATM может использоваться двухуровневая структура адресации? Опишите ситуации, в которых могут оказаться полезными дополнительные уровни иерархии.
- 18.9. В сети ATM выполняется упорядоченная доставка ячеек, однако в случае перегрузки часть из них может быть утеряна. Можно ли изменить протокол TCP так, чтобы уменьшить объем служебного трафика? Обоснуйте свой ответ.
- 18.10. Изучите стандарты LANE и МРОА, которые позволяют в сети ATM эмулировать работу локальной сети, такой как Ethernet или любой другой. В чем заключается главное преимущество использования сети ATM для эмулирования работы локальной сети? В чем состоит основной недостаток?
- 18.11. Администратор сети ATM большой организации собирается разделить ее компьютеры на логические IP-подсети. При этом существует два крайних подхода. Во-первых, можно отнести все узлы к одной большой подсети. Во-вторых, можно создать много подсетей

(например, отнести каждую пару узлов к одной логической подсети). Объясните, почему ни один из этих вариантов нежелателен.

- 18.12. Сколько ячеек ATM необходимо для передачи одного пакета протокола ATMARP при условии, что длина каждого физического адреса ATM вместе с подадресом составляет 20 октетов, а длина каждого протокольного адреса составляет 4 октета?
- 18.13. Сеть ATM позволяет узлу сети устанавливать несколько виртуальных каналов к одному получателю. В чем преимущество такого подхода?
- 18.14. Измерьте пропускную способность сети ATM и величину задержки передачи пакета протокола TCP через коммутатор ATM. Если позволяет ваша операционная система, повторите эксперимент, устанавливая различную длину буфера передачи протокола TCP. Если в вашей системе используется интерфейс сокетов, обратитесь к справочному руководству и уточните, как задается размер буфера. Удивляют ли вас полученные результаты?
- 18.15. В протоколе IP не предусмотрен механизм привязки дейтаграмм, проходящих по сети ATM, к конкретному виртуальному каналу. При каких условиях такой механизм оказался бы полезным?
- 18.16. Сервер ATMARP не удаляет данные со своего кэша сразу после того, как узел сети, который их прислал, закрывает с сервером канал связи. В чем преимущество и недостаток такого подхода?
- 18.17. Стоит ли использовать режим IP-коммутации для прикладных программ, которые используются на вашем компьютере? Чтобы найти ответ, пронаблюдайте за трафиком, исходящим от вашего компьютера, и вычислите среднюю продолжительность TCP-соединения, количество одновременно открытых соединений, а также количество IP-получателей, с которыми ваш компьютер устанавливает связь на протяжении недели.
- 18.18. Ознакомьтесь со спецификацией стандарта MPLS. Должна ли в нем поддерживаться пересылка дейтаграмм на втором уровне (т.е. выполняться функции моста), а также оптимизированная IP-пересылка? Обоснуйте свой ответ.



# Протокол мобильной связи с IP-сетями

## 19.1. Введение

В предыдущих главах были описаны первоначальные системы IP-адресации и маршрутизации, используемые в стационарных компьютерах. В этой главе рассматривается одно из новых расширений протокола IP, позволяющее портативным компьютерам легко перемещаться из одной сети в другую.

## 19.2. Мобильность, маршрутизация и адресация

В широком смысле слова термин *мобильная вычислительная система (mobile computing)* обозначает систему, в которой компьютеры могут свободно перемещаться с одного места на другое. Понятие мобильности зачастую ассоциируется с беспроводными технологиями, позволяющими абонентам перемещаться на дальние расстояния и на высокой скорости. Однако скорость не является основным препятствием при внедрении протокола IP в мобильных вычислительных системах. Трудность возникает только в том случае, когда узел сети изменяет свое физическое положение и переходит из одной сети в другую. Например, ноутбук, подключенный к беспроводной локальной сети, может перемещаться с высокой скоростью в зоне действия передатчика и при этом корректно работать без изменения IP-адреса. Однако чтобы просто отключить настольный компьютер и подключить его к другой сети, необходимо изменить параметры настройки протокола IP этого компьютера.

Система IP-адресации, которая была разработана и оптимизирована для использования в стационарном сетевом окружении, усложняет процесс внедрения мобильных коммуникаций. В частности, поскольку в префиксе IP-адреса узла содержится информация о номере сети, при перемещении узла в новую сеть нужно сделать одно из двух:

- изменить IP-адрес узла;
- маршрутизаторы должны анонсировать по объединенной сети маршруты к этому узлу.

Ни один из предложенных вариантов не является оптимальным. С одной стороны, на изменение IP-адреса требуется некоторое время. Как правило, это влечет за собой перезагрузку компьютера и разрыв всех существующих соединений на транспортном уровне. Кроме того, если узел сети связывается с сервером, использующим IP-адреса для аутентификации, может возникнуть необходимость

внести еще одно изменение в DNS-сервер. С другой стороны, подход, когда маршрутизация выполняется индивидуально для каждого узла сети, нельзя считать удачным с точки зрения расширяемости сети. Причина состоит в том, что для хранения информации об индивидуальных маршрутах необходимо зарезервировать место в таблицах маршрутизации, пропорциональное количеству узлов сети. Кроме того, передача маршрутной информации вызывает дополнительную нагрузку на сети.

### 19.3. Свойства мобильного протокола IP

Инженерная группа IETF нашла решение проблемы мобильности, с помощью которого удалось преодолеть некоторые из ограничений первоначальной системы IP-адресации. Официально эта технология называется *поддержкой мобильного протокола IP (IP mobility support)*, а в разговорном языке известна как *мобильный протокол IP*. Эта технологию характеризуют следующими свойствами.

- *Скрытие низкоуровневой структуры.* Мобильный протокол IP является “прозрачным” для приложений и протоколов транспортного уровня, а также для немобильных маршрутизаторов. В частности, находясь в неактивном состоянии, все открытые соединения протокола TCP сохраняют работоспособность после изменений в конфигурации сети и сразу же готовы для дальнейшего использования.
- *Взаимодействие с протоколом IP версии 4 (IPv4).* Узел сети, использующий мобильный протокол IP, может взаимодействовать со стационарными узлами сети, на которых запущено программное обеспечение стандартной версии протокола IPv4, а также с другими мобильными узлами. Более того, нет необходимости применять особую систему адресации, так как присвоенные мобильным узлам сети адреса ничем не отличаются от адресов, присвоенных стационарным узлам сети.
- *Расширяемость.* Мобильный протокол IP может использоваться и в больших объединенных сетях. В частности, он обеспечивает мобильность устройств в глобальной сети Internet.
- *Надежность.* В мобильном протоколе IP предусмотрены средства безопасности, которые используются для аутентификации всех сообщений (т.е. для того, чтобы помешать какому-либо компьютеру играть роль мобильного сетевого узла).
- *Макромобильность.* При создании мобильного протокола IP разработчики сконцентрировались на решении проблемы подключения мобильного узла к сети на длительное время, а не на обработке быстрых переходов из одной сети в другую, как при использовании беспроводной сотовой связи. Например, мобильный протокол IP может эффективно применяться пользователем, который решил взять с собой в командировку портативный компьютер и подключает его на неделю к новой сети.

### 19.4. Принципы работы мобильного протокола IP

Самое большое препятствие для реализации мобильной системы заключается в том, что при переходе из одной сети в другую узел сети должен сохранять свой адрес. При этом маршрутизаторы не должны устанавливать специфичные маршруты для каждого мобильного узла сети. В мобильном протоколе IP эта проблема решена за счет того, что одному компьютеру разрешено иметь одновременно два адреса. Первый адрес, который можно считать *основным адресом*

компьютера; он является фиксированным и никогда не меняется. Он используется приложениями и протоколами транспортного уровня. Второй адрес, который считается *дополнительным*, является временным. Он изменяется при подключении компьютера к другой сети и действителен только до тех пор, пока компьютер находится в пределах данной внешней сети.

При назначении основного адреса мобильному узлу учитывается его принадлежность к одной из исходных сетей (ее называют “*домашней*” сетью, или *home network*). Как только мобильный узел подключается к *внешней* сети (*foreign network*) и получает дополнительный адрес, он должен отослать новый адрес *агенту* (как правило, маршрутизатору), расположенному в домашней сети. Агент должен перехватывать все дейтаграммы, отосленные по основному адресу мобильного узла, и пересыпать их по дополнительному адресу, используя туннелирование и инкапсуляцию *IP-в-IP*<sup>1</sup>.

Если мобильный узел снова перемещается в другую сеть, ему назначается новый дополнительный адрес, который он должен сообщить агенту “*домашней*” сети. Когда мобильный узел возвращается обратно в “*домашнюю*” сеть, он должен обратиться к агенту “*домашней*” сети, чтобы *отменить регистрацию*. Это означает, что агент прекратит перехватывать предназначенные для мобильного узла дейтаграммы. Точно также, произвольный мобильный узел в любой момент может отменить регистрацию, например, при отключении от удаленной сети.

Уже было сказано, что мобильный протокол IP предназначен для обеспечения макромобильности, а не перемещений на высокой скорости. Причина, должно быть, понятна — непроизвольные издержки. В частности, переместившись, мобильный узел должен определить свое новое местоположение, сообщить эту информацию по внешней сети для получения дополнительного адреса, а затем обратиться по объединенной сети к своему агенту “*домашней*” сети, чтобы обеспечить пересылку дейтаграмм. Основной смысл сказанного выше заключается в следующем.

*Поскольку после каждого перемещения мобильного узла требуется выполнить ряд служебных действий, приводящих к значительным непроизвольным издержкам, мобильный протокол IP предназначен для использования в тех ситуациях, когда узел сети перемещается нечасто, находясь в определенном месте достаточно долго.*

## 19.5. Особенности адресации мобильного протокола IP

Основной или “*домашний*” адрес мобильного узла присваивается и контролируется администратором “*домашней*” сети. Этот адрес ничем не отличается от любого другого адреса стационарного компьютера, подключенного к “*домашней*” сети. В приложениях, запущенных на мобильном компьютере всегда используется основной адрес.

Каждый раз, когда мобильный узел подключается ко внешней сети, он должен получить временный адрес. Этот адрес не используется приложениями, выполняющимися на мобильном компьютере; он просто выполняет роль адреса *для передачи* (*care-of address*). Им оперирует только программное обеспечение протокола IP мобильного узла, а также агенты “*домашней*” или внешней сети. Управление адресом для передачи осуществляется точно так же, как и любым другим адресом, принадлежащим внешней сети. Информация о маршрутах, ведущих к узлу, которому присвоен временный адрес для передачи, распространяется с помощью одного из стандартных протоколов маршрутизации.

<sup>1</sup> Инкапсуляция *IP-в-IP* описана в главе 17, “Режим многоадресатной передачи в объединенной сети”.

На практике применяются временные адреса двух типов. Тип адреса, используемый, подключившимся к сети мобильным узлом, определяется администратором этой сети. Типы адресов отличаются способом назначения временного адреса узлу сети, а также объектом, ответственным за пересылку пакетов. Первый тип временного адреса называется *сопряженным адресом для передачи* (*co-located care-of address*). При этом мобильный компьютер должен сам пересыпать все пакеты. В сущности на мобильном компьютере, использующем сопряженный временный адрес, всегда будет присутствовать программное обеспечение, в котором одновременно используются два адреса. Как уже было сказано выше, основной адрес используется в приложениях, а временный адрес используется в программах нижнего уровня для получения дейтаграмм. Главное преимущество сопряженного адреса заключается в том, что его можно использовать в существующей инфраструктуре объединенной сети. Маршрутизаторам, находящимся во внешних сетях, не известно, является ли определенный компьютер мобильным или нет. Временные адреса назначаются мобильным компьютерам с помощью тех же механизмов, которые используются для присвоения адресов стационарным компьютерам (например, с помощью протокола DHCP, рассмотренного в главе 23, “Начальная загрузка и автоконфигурация (BOOTP, DHCP)”). Главный недостаток при использовании сопряженного временного адреса заключается в том, что на мобильном компьютере должно быть установлено дополнительное программное обеспечение, с помощью которого осуществляется получение этого адреса и взаимодействие с агентом “домашней” сети.

Второй тип временного адреса связан с использованием *агента во внешней сети*, который, по сути, является активным участником процесса передачи дейтаграмм от мобильного компьютера в “домашнюю” сеть, и наоборот. Активный объект, который одновременно является маршрутизатором, называют *внешним агентом* (*foreign agent*), чтобы его можно было отличить от “домашнего” агента (*home agent*), расположенного в домашней сети мобильного компьютера. При использовании описанной формы адресации мобильный компьютер должен вначале определить адрес внешнего агента, а затем обратиться к нему за получением временного адреса. В данном случае внешнему агенту не нужно присваивать мобильному компьютеру уникальный IP-адрес. Ниже будет показано, что вместо этого агент может выделить в качестве временного адреса один из своих IP-адресов, и организовать пересылку дейтаграмм. Хотя присвоение уникального адреса несколько облегчает взаимодействие с мобильным компьютером, использование существующих IP-адресов внешнего агента означает, что для подключения мобильных компьютеров временных посетителей не нужно выделять отдельных IP-адресов, что приводит к экономии адресного пространства.

## 19.6. Поиск внешнего агента

Для обнаружения внешнего агента используется механизм *поиска маршрутизатора* протокола ICMP. В главе 9, “Протокол IP: обработка ошибок и управляющие сообщения (ICMP)”, уже упоминалось, что каждый маршрутизатор периодически рассыпает ICMP-сообщение, извещающее узлы сети об адресах маршрутизаторов, а узел сети может отослать на адрес маршрутизатора ICMP-запрос, в ответ на который маршрутизатор пришлет нужную информацию<sup>2</sup>. При поиске агента в предназначенный маршрутизатору ICMP-запрос включается дополнительная информация, с помощью которой внешний агент может анонсировать свое присутствие, а мобильный компьютер — отослать запрос на анонсирование информации.

<sup>2</sup> Мобильный узел сети, которому не известен IP-адрес агента, может передавать информацию *группе всех агентов* в режиме многоадресатной передачи по IP-адресу 224.0.0.11.

Дополнительная информация, помещаемая в каждое сообщение, называется *расширением мобильного агента*<sup>3</sup>. Для идентификации расширений не используется отдельный тип ICMP-сообщения. Мобильный узел может определить, что расширение включено в дейтаграмму, если ее длина, определенная в IP-заголовке, больше, чем длина ICMP-запроса, посланного маршрутизатору. На рис. 19.1 показан формат расширения мобильного агента.



Рис. 19.1. Формат расширения мобильного агента, которое добавляется к ICMP-сообщению, анонсирующему адреса маршрутизаторов

Каждое сообщение начинается с поля *типа* размером один октет, после которого следует состоящее из одного октета поле *длины*. В поле *длины* указывается размер следующего за ним сообщения в октетах. Другими словами в этом поле указывается значение длины расширения мобильного агента минус 2 октета, поскольку размеры полей *типа* и *длины* не учитываются. В поле *времени жизни* определяется максимальный период времени в секундах, на протяжении которого агент готов принимать запросы на регистрацию. Если в поле указано значение, состоящее из всех единиц, то это соответствует бесконечному времени жизни. В поле *порядкового номера* указывается номер сообщения, что позволяет получателю обнаружить потерю сообщений. Единичное значение каждого из битов в поле *кода* определяет одно из свойств агента, перечисленных в табл. 19.1.

Таблица 19.1. Формат поля кода расширения мобильного агента

Номер бита	Описание
0	Необходима регистрация у агента; использование сопряженных временных адресов запрещено
1	Агент занят и не принимает запросы на регистрацию
2	Агент выполняет функции "домашнего" агента
3	Агент выполняет функции внешнего агента
4	Агент использует минимальную инкапсуляцию
5	Агент использует инкапсуляцию типа GRE <sup>4</sup>
6	Агент поддерживает сжатие заголовков при взаимодействии с мобильным узлом сети
7	Не используется (должно равняться нулю)

<sup>3</sup> Агент мобильного узла сети также присоединяет к сообщению *префиксное расширение*, в котором указывается используемый в сети префикс IP-адреса. Мобильный узел использует префиксное расширение, чтобы определить факт подключения к новой сети.

<sup>4</sup> Аббревиатура GRE (*Generic Routing Encapsulation*, или *обобщенная маршрутная инкапсуляция*), употребляется для обозначения обобщенной системы, позволяющей инкапсулировать дейтаграммы любого протокола. Инкапсуляция типа IP-в-IP представляет собой один из частных случаев употребления этой системы.

## 19.7. Регистрация агента

Прежде чем мобильный узел, подключенный ко внешней сети, сможет получать дейтаграммы из объединенной сети, он должен пройти регистрацию. Процесс *регистрации* позволяет узлу сети следующее.

- Зарегистрироваться у агента во внешней сети.
- Зарегистрироваться непосредственно у своего “домашнего” агента, чтобы он начал пересыпать пакеты.
- Обновить регистрацию, срок действительности которой истекает.
- Отменить регистрацию после возвращения в “домашнюю” сеть.

Если мобильному компьютеру присвоен сопряженный временный адрес, он может напрямую взаимодействовать с объединенной сетью и сам пройти процесс регистрации. Мобильный компьютер может использовать этот адрес для взаимодействия со своим “домашним” агентом. Однако если мобильный компьютер получил временный адрес от внешнего агента, он не может использовать его для взаимодействия напрямую с агентом своей “домашней” сети. Вместо этого, мобильный компьютер должен послать запрос на регистрацию внешнему агенту, который затем связывается с “домашним” агентом мобильного узла от своего имени. Точно так же внешний агент должен пересыпать полученные им сообщения, предназначенные для мобильного узла сети.

## 19.8. Формат сообщения о регистрации

Все сообщения о регистрации отсылаются по протоколу UDP в порт под номером 434. Агенты прослушивают все сообщения, поступающие через этот порт. Запросы могут отсылаться из любого порта отправителя в порт 434 получателя. При ответе агент меняет местами номера портов отправителя и получателя, поэтому ответ на запрос отсылается из порта отправителя 434 в порт, используемый запрашивающим узлом сети.

Сообщение о регистрации начинается с набора полей фиксированного размера, после которого следуют *расширения* переменной длины. Каждый запрос должен содержать *расширение аутентификации мобильного компьютера в “домашней” сети*, которое позволяет “домашнему” агенту подтвердить подлинность мобильного узла сети. На рис. 19.2 показан формат такого сообщения.

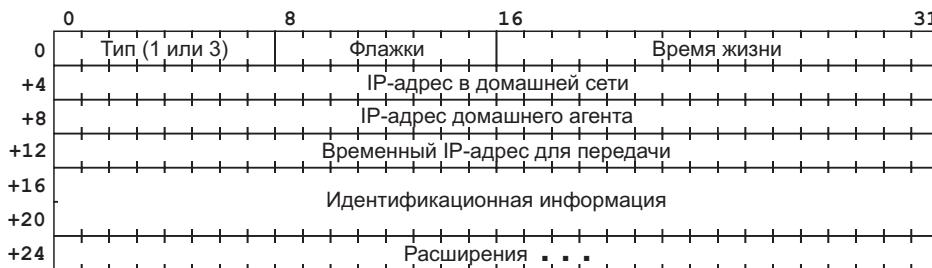


Рис. 19.2. Формат сообщения о регистрации мобильного протокола IP

В поле *типа* указывается, является ли сообщение запросом на регистрацию (1) или ответом на запрос о регистрации (3). В поле *времени жизни* определяется количество секунд, на протяжении которых действительна регистрация (значение ноль обозначает срочную отмену регистрации, а все единицы определяют

бесконечное время жизни). В следующих трех полях указываются IP-адреса мобильного компьютера в “домашней” сети, агента “домашней” сети и временный адрес, присвоенный мобильному компьютеру. В поле *идентификационной информации* находится 64-х битовое число, генерируемое мобильным компьютером, которое используется для сопоставления запросов с входящими ответами, а также для того, чтобы помешать мобильному компьютеру принимать устаревшие сообщения. Биты в поле *флажков* используются для определения особенностей пересылки, которые перечислены в табл. 19.2.

**Таблица 19.2. Формат поля флажков запроса на регистрацию мобильного компьютера**

<i>Номер бита</i>	<i>Описание</i>
0	В сообщении содержится параллельный (дополнительный) адрес, а не замена существующего адреса
1	Мобильный узел сети посыпает запрос “домашнему” агенту на туннелирование копии каждой широковещательной дейтаграммы
2	Мобильный узел сети использует сопряженный временный адрес и сам удаляет инкапсуляцию дейтаграмм
3	Мобильный узел сети посыпает запрос агенту на использование минимальной инкапсуляции
4	Мобильный узел сети посыпает запрос агенту на использование GRE-инкапсуляции
5	Мобильный узел сети посыпает запрос на сжатие заголовков
6–7	Зарезервированы (должны равняться нулю)

Если мобильному компьютеру назначен сопряженный временный адрес, он может отослать запрос на регистрацию непосредственно своему “домашнему” агенту. В противном случае, мобильный компьютер отсылает запрос внешнему агенту, который затем пересыпает его “домашнему” агенту. В последнем случае запрос обрабатывается и внешним, и домашним агентами, поэтому они оба должны его утвердить. Например, либо “домашний”, либо внешний агент может изменить время жизни регистрации.

## 19.9. Взаимодействие с внешним агентом

Как уже упоминалось, внешний агент может назначить в качестве временного адреса один из своих IP-адресов. При этом, с точки зрения внешней сети, мобильному компьютеру не присваивается уникальный адрес. Следовательно, возникает вопрос: как внешний агент и мобильный узел взаимодействуют по внешней сети, если мобильный узел не имеет действительного IP-адреса в этой сети? Для взаимодействия необходимо смягчить правила IP-адресации и применить другую систему привязки адресов. В частности, когда мобильный узел отсылает сообщение внешнему агенту, ему должно быть разрешено использовать свой основной адрес в качестве IP-адреса отправителя. Более того, когда внешний агент отсылает дейтаграмму мобильному узлу, агенту также должно быть разрешено использовать основной адрес мобильного узла в качестве IP-адреса получателя.

Хотя существует возможность применения основного адреса мобильного узла во внешней сети, агент не может для его преобразования в физический адрес использовать протокол ARP (т.е. протокол ARP можно использовать для преобразования только IP-адресов, которые являются действительными в данной сети). Чтобы выполнить привязку адресов, не прибегая к использованию протокола ARP,

агент должен зарегистрировать всю информацию о мобильном узле во время поступления запроса на регистрацию и хранить эту информацию на протяжении всего взаимодействия. В частности, агент должен зарегистрировать физический адрес мобильного узла. Посылая дейтаграмму мобильному узлу, агент должен просмотреть зарегистрированную информацию и определить его физический адрес. Таким образом, даже не используя протокол ARP, агент может отсылать дейтаграммы мобильному узлу в режиме одноадресатной передачи на уровне сетевого оборудования. Подведем итог всему сказанному выше.

*Если мобильный узел не имеет уникального адреса во внешней сети, внешний агент должен для взаимодействия с ним использовать его основной адрес. Для привязки адресов агент не использует протокол ARP, а получив запрос на регистрацию, сохраняет у себя физический адрес мобильного узла и пользуется им в дальнейшем для привязки адресов.*

## 19.10. Передача и получение дейтаграмм

Пройдя регистрацию, мобильный узел может взаимодействовать с любым компьютером во внешней сети. Для этого мобильный узел создает дейтаграмму, в поле адреса получателя которой указывается IP-адрес целевого компьютера, а в поле адреса отправителя — основной IP-адрес мобильного узла<sup>5</sup>. Дейтаграмма будет передана по кратчайшему маршруту из внешней сети к конечному получателю. Однако, ответное сообщение не будет следовать по кратчайшему маршруту напрямую к мобильному узлу. Вместо этого, ответное сообщение попадет в “домашнюю” сеть мобильного узла. “Домашний” агент, который знает о местонахождении мобильного узла из зарегистрированной информации, перехватывает дейтаграмму и использует инкапсуляцию *IP-в-IP* для туннелирования дейтаграммы по временному адресу. Если мобильному узлу назначен сопряженный временный адрес, инкапсулированная дейтаграмма передается непосредственно этому узлу, который удаляет внешнюю дейтаграмму и затем обрабатывает внутреннюю. Если мобильный узел использует для взаимодействия временный адрес, выделенный внешним агентом, инкапсулированная дейтаграмма поступает на обработку внешнему агенту. Получив дейтаграмму от “домашнего” агента, внешний агент удаляет внешнюю оболочку дейтаграммы, сверяет полученные данные со своей таблицей зарегистрированных мобильных компьютеров и передает дейтаграмму по локальной сети соответствующемуциальному узлу. Итак, можно подвести некоторые итоги.

*Поскольку при отправке дейтаграмм мобильный узел в качестве адреса отправителя использует свой основной адрес, каждое ответное сообщение пересыпается в “домашнюю” сеть мобильного узла. Агент этой сети перехватывает дейтаграмму, инкапсулирует ее в другую дейтаграмму и пересыпает ее либо непосредственно мобильному узлу, либо внешнему агенту, используемому мобильным узлом для пересылки информации.*

## 19.11. Проблема двух пересечений

Из приведенного выше описания становится понятен недостаток мобильного протокола IP — неэффективная маршрутизация. Поскольку в качестве адреса отправителя мобильный узел использует основной адрес, ответная дейтаграмма

---

<sup>5</sup> Администрация внешней сети и провайдер Internet, через которого она подключена к остальной части объединенной сети, должны разрешить пересылку дейтаграмм с произвольным адресом отправителя.

посыпается сначала в его “домашнюю” сеть и только затем — самому мобильному узлу. Эту проблему решить непросто, поскольку взаимодействие между компьютерами зачастую представляет собой пример *пространственного расположения связей* (*spatial locality of reference*). Это означает, что мобильный компьютер, подключившийся ко внешней сети, по всей вероятности вступит во взаимодействие с подключенными к этой сети компьютерами. Чтобы понять, почему мобильный протокол IP плохо справляется с проблемой пространственного расположения связей, рассмотрим изображенную на рис. 19.3 топологию сети.

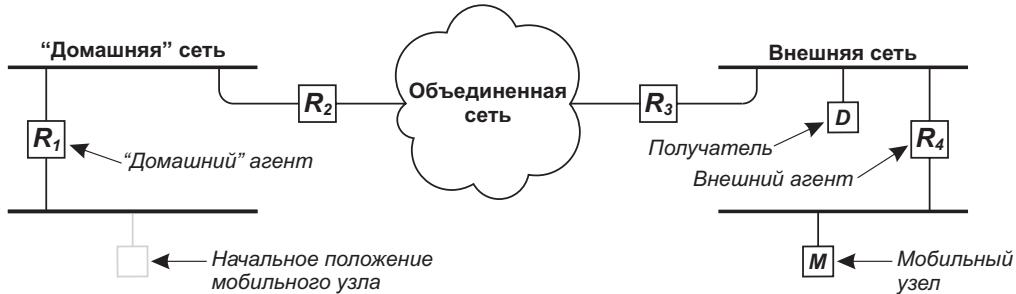


Рис. 19.3. Пример топологии сети, в которой маршрутизация, выполняющаяся мобильным протоколом IP, является неэффективной. Когда мобильный узел  $M$  вступает во взаимодействие с локальным получателем  $D$ , дейтаграммы от получателя  $D$  следуют по объединенной сети к “домашнему” агенту мобильного узла, а затем обратно к мобильному узлу

На рис. 19.3 мобильный узел  $M$  переместился из своей исходной сети во внешнюю сеть. Предположим, что мобильный узел зарегистрировал информацию о себе у своего “домашнего” агента (маршрутизатора  $R_1$ ), который будет ему пересыпать дейтаграммы. Теперь рассмотрим взаимодействие между мобильным узлом и получателем  $D$ , который находится в том же сетевом центре, что и мобильный сетевой узел. Дейтаграммы от узла  $M$  до получателя  $D$  проходят через маршрутизатор  $R_4$ , а затем доставляются получателю  $D$ . Однако, поскольку получатель  $D$  посыпает дейтаграммы мобильному узлу  $M$  по его основному адресу, они следуют через маршрутизатор  $R_3$  и затем по объединенной сети в “домашнюю” сеть мобильного узла. Когда дейтаграммы достигают маршрутизатора  $R_1$  (“домашнего” агента мобильного узла), они туннелируются обратно по объединенной сети во внешний сетевой центр (непосредственно мобильному узлу  $M$  либо его внешнему агенту). Поскольку стоимость прохождения дейтаграмм по объединенной сети намного выше, чем по локальной сети, описанную выше ситуацию называют *проблемой двух пересечений*. Иногда ее также называют *проблемой 2Х*<sup>6</sup>.

Мобильный протокол IP не гарантирует решение проблемы 2Х. Однако возможно несколько оптимизировать процесс маршрутизации. В частности, если в данном сетевом центре существует большая вероятность активного взаимодействия подключившегося к нему мобильного узла с локальными компьютерами, можно создать специфичные маршруты к этому мобильному узлу (*host-specific route*). Чтобы обеспечить правильную маршрутизацию, этот специфичный маршрут необходимо удалить, когда мобильный узел покинет сеть. Конечно, проблема остается при взаимодействии мобильного узла с получателем, расположенным за пределами зоны, в которой действует специфичный для мобильного

<sup>6</sup> Если получатель  $D$  находится на большом расстоянии от мобильного узла, проблема несколько смягчается. Ее называют проблемой *пересылки по треугольнику*, или *пересылкой по ломаной линии*.

узла маршрут. Предположим, что мобильные узлы часто перемещаются из одной корпоративной сети в другую, одна из которых находится в городе А, а другая — в городе Б. Сетевые администраторы двух сетевых центров могут договориться о том, чтобы распространять специфичные маршруты для всех подключающихся к их сетям мобильных узлов. Это означает, что если мобильный компьютер вступает во взаимодействие с другими компьютерами во внешнем сетевом центре, трафик остается локальным по отношению к сетевому центру. Однако, поскольку использование маршрутов, специфичных для отдельных узлов сети, ограничиваются только двумя корпоративными сетевыми центрами, обмен информацией между мобильным узлом и любым другим получателем, находящимся в чужом городе, приведет к тому, что ответные сообщения будут пересыпаться через “домашнего” агента мобильного компьютера. Таким образом, проблема 2Х возникает при взаимодействии мобильного узла сети с любым получателем, расположенным за пределами корпоративных сетевых центров. Подведем итог всему сказанному выше.

*При использовании мобильного протокола IP возникает проблема неэффективной маршрутизации, которая называется проблемой 2Х. Проблема возникает, когда мобильный компьютер подключается к внешней сети, находящейся на большом расстоянии от его “домашней” сети, и затем взаимодействует с компьютером, находящимся недалеко от внешнего сетевого центра. Каждая посланная мобильному компьютеру дейтаграмма следует по объединенной сети к “домашнему” агенту мобильного компьютера, который затем пересыпает дейтаграмму обратно во внешний сетевой центр. Чтобы устранить эту проблему, необходимо обеспечить распространение маршрутной информации, специфичной для мобильного узла. Проблема остается нерешенной для тех получателей, для которых явно не задан маршрут.*

## 19.12. Взаимодействие с компьютерами “домашней” сети

Уже было сказано, что при подключении мобильного узла ко внешней сети его “домашней” агент должен перехватывать все посланные этому узлу дейтаграммы. Как правило, “домашний” агент — это маршрутизатор, через который “домашняя” сеть мобильного узла подключается к остальной части объединенной сети. Таким образом, все поступающие для мобильного узла дейтаграммы проходят через “домашнего” агента. Прежде чем переслать дейтаграмму, “домашний” агент должен проанализировать свою таблицу мобильных узлов и определить, находится ли в данный момент узел получателя в “домашней” сети или он подключился ко внешней сети.

Хотя “домашний” агент может легко перехватить все дейтаграммы, которые поступают для мобильного узла из других сетей, существует еще один случай, который должен учитывать агент. Трудности возникают с дейтаграммами, которые отправляются мобильному узлу из его “домашней” сети. В частности, рассмотрим, что происходит, когда произвольный узел, расположенный в “домашней” сети мобильного компьютера, отсылаетциальному узлу дейтаграмму. Поскольку в протоколе IP в таких случаях предусмотрена непосредственная доставка пакетов получателю по локальной сети, отправитель не будет пересыпать дейтаграмму маршрутизатору. Вместо этого отправитель с помощью протокола ARP должен определить физический адрес мобильного узла, инкапсулировать дейтаграмму в сетевой фрейм и переслать ее по локальной сети.

Если мобильный узел переместился во внешнюю сеть, домашний агент должен перехватывать все дейтаграммы, включая те, которые были отосланы узлами

“домашней” сети. Чтобы перехватить дейтаграммы, поступающие от локальных узлов, “домашний” агент должен использовать технологию агента ARP (proxy ARP). Это означает, что “домашний” агент должен перехватывать все ARP-запросы, в которых в качестве целевого объекта указан IP-адрес мобильного узла, и отвечать на них, указав в сообщении собственный физический адрес. Агент ARP является полностью “прозрачной” технологией для локальных компьютеров. Другими словами, любой локальный компьютер, который использует ARP для преобразования адреса мобильного узла, получит корректный ответ на свой запрос и перешлет дейтаграмму обычным образом “домашнему” агенту.

С помощью использования агента ARP также решается проблема нескольких подключений. Если в “домашней” сети мобильного узла находится несколько маршрутизаторов, подключенных к различным частям объединенной сети, только один из них должен выполнять функции “домашнего” агента данного мобильного узла. Остальные маршрутизаторы ничего не знают о мобильных компьютерах и используют обычным образом протокол ARP для преобразования адресов. Таким образом, поскольку “домашний” агент отвечает на ARP-запросы, другие маршрутизаторы пересыпают дейтаграммы, не делая различий между мобильными и немобильными узлами сети.

### 19.13. Резюме

Мобильный протокол IP позволяет компьютеру перемещаться из одной сети в другую без изменения своего IP-адреса. При этом все маршрутизаторы не должны распространять специфичные для этого мобильного узла маршруты. Когда мобильный компьютер перемещается из исходной сети во внешнюю, он должен получить дополнительный временный адрес, который будет выполнять функции адреса для передачи. В приложениях, запущенных на мобильном компьютере, используется основной адрес мобильного узла. Временный адрес используется только низкоуровневым сетевым программным обеспечением для пересылки и доставки пакетов по внешней сети.

Как только мобильный компьютер подключится ко внешней сети, ему либо присваивается сопряженный временный адрес, либо он выполняет поиск внешнего мобильного агента и посыпает ему запрос на получение временного адреса. Получив временный адрес, мобильный компьютер проходит регистрацию у своего “домашнего” агента (напрямую или косвенно, через внешнего агента) и отсылает ему запрос на пересылку дейтаграмм.

После завершения регистрации, мобильный компьютер может вступать во взаимодействие с любым компьютером объединенной сети. Посланные мобильным компьютером дейтаграммы пересыпаются непосредственно конечному получателю. Однако каждая дейтаграмма, посланная в ответ мобильному компьютеру, следует в его “домашнюю” сеть, где ее перехватывает “домашний” агент, инкапсулирует в другую IP-дейтаграмму и затем туннелирует мобильному компьютеру во внешнюю сеть.

### Материал для дальнейшего изучения

Перкинс (Perkins) в [RFC 2002] описывает мобильный протокол IP и рассматривает форматы сообщений. В проекте стандарта Internet [draft-ietf-mobileip-v2-00.txt] описана версия 2 мобильного протокола IP. Перкинс в [RFC 2003] и [RFC 2004], Ханкс (Hanks) и др. в [RFC 1701] описывают подробности трех систем инкапсуляции IP-в-IP. Монтенегро (Montenegro) в [RFC 2344] рассматривает систему обратного туннелирования для мобильного протокола IP. И наконец, Перкинс и Джонсон (Johnson) в [draft-ietf-mobileip-optim-07.txt] рассматривают процесс оптимизации маршрутизации для мобильного протокола IP.

## Упражнения

- 19.1. Сравните системы инкапсуляции, описанные в [RFC 2003] и [RFC 2004]. Назовите преимущества и недостатки каждой из них.
- 19.2. Внимательно ознакомьтесь со спецификацией мобильного протокола IP. Как часто маршрутизатор должен посыпать анонс об агенте мобильного узла? Обоснуйте свой ответ.
- 19.3. Обратитесь к описанию стандарта мобильного протокола IP. Когда внешний агент пересыпает запрос на регистрацию “домашнему” агенту мобильного компьютера, какие порты протокола при этом используются? Объясните, почему.
- 19.4. В стандарте мобильного протокола IP предусмотрено, что один и тот же маршрутизатор может выполнять функции как “домашнего” агента текущей сети, так и внешнего агента для мобильных компьютеров, подключившихся к этой сети. В чем преимущества и недостатки использования одного маршрутизатора для выполнения обеих функций?
- 19.5. В спецификации мобильного протокола IP определены три концептуально независимые формы аутентификации: мобильный компьютер — “домашний” агент; мобильный компьютер — внешний агент; и внешний агент — “домашний” агент. В чем преимущество разделения аутентификации на три формы? В чем недостатки?
- 19.6. Обратитесь к спецификации мобильного протокола IP и выясните, как мобильный узел может присоединиться к многоадресатной группе. Как многоадресатные дейтаграммы могут направляться мобильному узлу? Какая из систем маршрутизации является оптимальной?

# 20

## Взаимодействие частных сетей (NAT, VPN)

### 20.1. Введение

В предыдущих главах объединенная сеть описывалась как одноуровневая абстракция, которая состоит из сетей, связанных между собой маршрутизаторами. В этой главе в качестве альтернативы рассмотрена двухуровневая структура объединенной сети, в которой организации имеют частные (или закрытые) объединенные сети, соединенные между собой посредством открытой объединенной сети (или открытых каналов связи).

В этой главе рассмотрены сетевые технологии, использующиеся в подобной двухуровневой структуре. Одна из технологий предназначена для решения проблемы ограниченного адресного пространства, а другая — позволяет расширить функциональные возможности сетевой технологии, чтобы не допустить посторонних к просмотру закрытых данных. Речь идет об обеспечении *секретности* (*privacy*) в объединенной сети.

### 20.2. Частные и гибридные сети

Один из основных недостатков одноуровневой структуры сети Internet — отсутствие секретности. Если у организации есть несколько сетевых центров, то при пересылке информации между ними через открытую объединенную сеть дейтаграммы проходят через сети, принадлежащие сторонним организациям. При этом их содержимое может быть просмотрено посторонними лицами. В двухуровневой сетевой структуре проводится различие между *внутренними* и *внешними* дейтаграммами (т.е. дейтаграммами, пересылаемыми между двумя компьютерами в пределах одной организации, и дейтаграммами, пересылаемыми между компьютерами, находящимися в разных организациях). Это сделано для того, чтобы гарантировать *секретность* внутренних дейтаграмм при их пересылке по открытым каналам связи.

Простейшим способом обеспечения секретности при передаче информации между компьютерами организации является создание полностью изолированной *частной* объединенной сети, которую обычно называют *частной сетью*. То есть, организация создает собственную объединенную сеть на основе протокола TCP/IP и изолирует ее от глобальной сети. В каждом сетевом центре частной сети для объединения внутренних сетей используются маршрутизаторы, а для соединения сетевых центров между собой используется закрытый цифровой выделенный канал связи. Поскольку никто из посторонних не имеет доступа ни к какой части закрытой сети, все данные при этом остаются секретными. Более

того, поскольку частная сеть изолирована от глобальной объединенной сети, в ней могут использоваться произвольные IP-адреса.

Естественно, полная изоляция не всегда желательна. Поэтому многие организации выбирают *гибридную сетевую структуру*, сочетающую в себе преимущества открытой сети с возможностью соединения с глобальной сетью Internet. Это значит, что организация подключает каждый сетевой центр к глобальной сети Internet, используя при этом реальные IP-адреса (действительные во всей глобальной сети). Преимуществом подобной структуры является то, что при необходимости компьютеры внутренней сети организации могут получить доступ к глобальной сети Internet, и при этом гарантируется полная секретность информации, пересылаемой по внутренним сетям. Для примера рассмотрим гибридную сетевую структуру, представленную на рис. 20.1. В подобной системе два сетевых центра организации соединены посредством закрытого канала связи, и каждый из центров имеет соединение с сетью Internet.



Рис. 20.1. Пример гибридной сети. В дополнение к закрытому выделенному каналу связи, соединяющему два сетевых центра, каждый из этих центров имеет соединение с глобальной сетью Internet

На рис. 20.1 закрытый выделенный канал связи между маршрутизаторами  $R_2$  и  $R_4$  обеспечивает секретность потока информации, передаваемой между сетевыми центрами. Поэтому процесс маршрутизации между центрами организован так, чтобы поток информации преимущественно направлялся через выделенный канал связи, а не через глобальную сеть Internet.

### 20.3. Виртуальная частная сеть

Основным недостатком и полностью открытой сети, и гибридной системы является их высокая стоимость, поскольку аренда выделенного канала связи (например, типа T1) — довольно дорогое удовольствие. Поэтому многие организации ищут более дешевые альтернативные варианты. Один из путей снижения стоимости заключается в использовании альтернативных сетевых технологий. Например, организация постоянного виртуального канала (permanent virtual circuit, или PVC) на основе технологий ретрансляции кадров (Frame Relay) или ATM через одного из национальных операторов связи может стоить гораздо дешевле, чем выделенный Т-канал с такой же пропускной способностью. Другой путь снижения стоимости состоит в использовании меньшего количества выделенных каналов. Очевидно, что минимальной стоимости системы удается достичь, отказавшись от использования выделенных каналов связи и пусть весь трафик через глобальную сеть Internet.

На первый взгляд может показаться, что использование глобальной сети Internet для взаимосвязи между сетевыми центрами не позволяет достичь такой

же секретности, как в случае полностью закрытой сети. Поэтому возникает следующий вопрос.

*Как может организация, которая использует глобальную сеть Internet для соединения своих центров, сохранить секретность передаваемых данных?*

Этого можно достичь с помощью технологии, позволяющей конфигурировать виртуальную частную сеть (*Virtual Private Network*, или *VPN*)<sup>1</sup>. Технология VPN является полным аналогом реальной частной (или закрытой) сети, поскольку гарантирует, что соединение между любой парой компьютеров в VPN остается секретным для посторонних. Слово “виртуальная” в названии технологии VPN говорит о том, что для соединения сетевых центров не используется реальный выделенный канал связи. Вместо этого для передачи информации от одного узла VPN к другому используется глобальная сеть Internet.

В основе работы VPN лежат два фундаментальных понятия: *туннелирование* и *шифрование*. Туннелирование уже было описано в главах 17, “Режим многоадресатной передачи в объединенной сети”, и 19, “Протокол мобильной связи с IP-сетями”. Для построения виртуальных частных сетей используется та же идея. Между маршрутизаторами, находящимися в разных сетевых центрах, прокладывается туннель через глобальную сеть Internet. При этом для пересылки дейтаграмм по туннелю используется инкапсуляция типа *IP-в-IP*.

Несмотря на использование аналогичной идеи, VPN-туннель в значительной степени отличается от туннелей, описанных в предыдущих главах. В частности, чтобы гарантировать секретность передаваемых данных, в VPN каждая выходная дейтаграмма перед инкапсуляцией в другую дейтаграмму для передачи шифруется<sup>2</sup> (рис. 20.2).

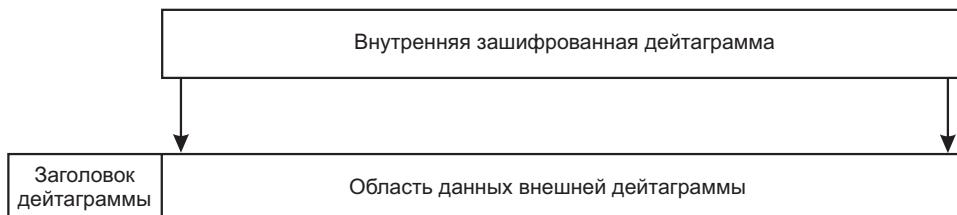


Рис. 20.2. Инкапсуляция типа *IP-в-IP*, используемая в VPN. Для обеспечения секретности внутренняя дейтаграмма зашифровывается перед помещением ее во внешнюю дейтаграмму для отправки

Как показано на рис. 20.2, вся внутренняя дейтаграмма, включая заголовок, зашифровывается перед инкапсуляцией. После прохождения дейтаграммы через туннель, ее область данных расшифровывается принимающим маршрутизатором, в результате чего восстанавливается внутренняя дейтаграмма, которая затем пересыпается получателю. Хотя при движении через туннель внешняя дейтаграмма может проходить через любое количество сетей, принадлежащих третьим лицам, посторонние компьютеры не могут расшифровать ее содержимое, поскольку они не знают ключа шифрования. Более того, они даже не смогут определить отправителя и конечного получателя внутренней дейтаграммы, поскольку ее заголовок также зашифрован. Из заголовка внешней дейтаграммы можно определить только IP-адреса маршрутизаторов, расположенных

<sup>1</sup> Это название не совсем правильное, поскольку в действительности эта технология обеспечивает создание виртуальной частной объединенной сети.

<sup>2</sup> Вопросы безопасности протокола IP и процесс инкапсуляции, использующийся в протоколе IPsec, рассматривается в главе 32, “Безопасность в объединенной сети и брандмауэры (IPsec)”.

в начале и конце туннеля. Эти адреса помещаются в поля адресов отправителя и получателя внешней дейтаграммы. Из всего сказанного выше можно сделать такой вывод.

*В технологии VPN данные пересыпаются через открытую сеть Internet, а для обеспечения секретности содержимое всех передаваемых между сетевыми центрами дейтаграмм шифруется.*

## 20.4. Адресация и маршрутизация в виртуальной частной сети

Легче всего понять, как происходит процесс адресации и маршрутизации в виртуальной частной сети, представив себе, что каждый VPN-туннель является аналогом выделенного канала связи в реальной частной сети. Как и в случае частной сети, в маршрутизаторе явно указываются маршруты ко всем внутренним сетям организации. Однако вместо передачи данных по выделенному каналу связи, в VPN данные передаются через туннель. В качестве примера на рис. 20.3 показана виртуальная частная сеть, которая эквивалента реальной частной сети, представленной на рис. 20.1, а также содержимое таблицы маршрутизации устройства  $R_1$ , управляющего туннелированием.

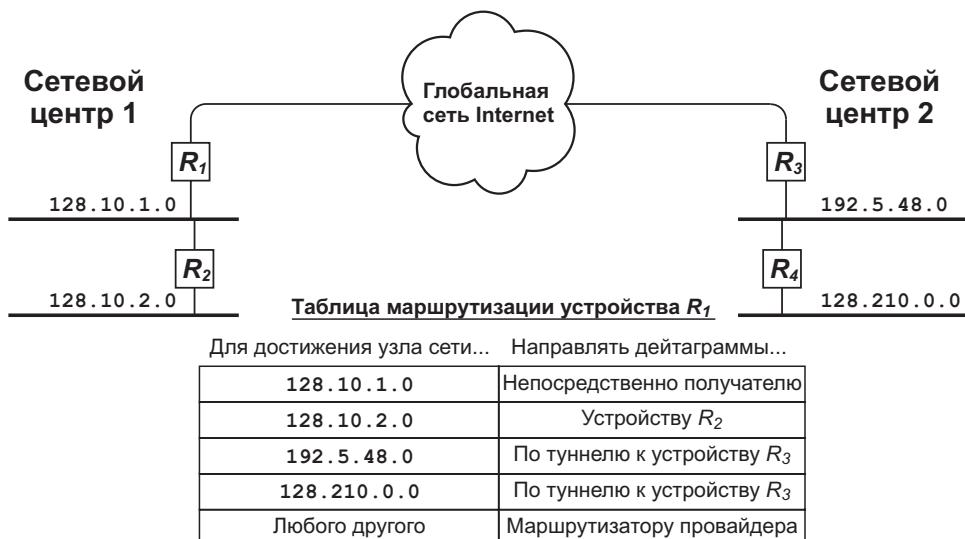


Рис 20.3. Виртуальная частная сеть, которая соединяет два сетевых центра и таблица маршрутизации устройства  $R_1$ . Туннель между маршрутизаторами  $R_1$  и  $R_3$  сконфигурирован по типу двухточечного выделенного канала связи

В качестве примера пересылки данных по виртуальной частной сети рассмотрим дейтаграмму, отправленную с компьютера, находящегося в сети 128.10.2.0 на компьютер, находящийся в сети 128.210.0.0. Отправляющий сетевой узел пересыпает дейтаграмму маршрутизатору  $R_2$ , который пересыпает ее маршрутизатору  $R_1$ . Согласно таблице маршрутизации устройства  $R_1$ , дейтаграмма должна быть отправлена по туннелю маршрутизатору  $R_3$ . Поэтому маршрутизатор  $R_1$  зашифровывает дейтаграмму, помещает ее в область данных внешней дейтаграммы, которую адресует получателю  $R_3$ . Затем маршрутизатор  $R_1$  пересыпает

внешнюю дейтаграмму маршрутизатору местного провайдера Internet, который отправляет ее по сети Internet маршрутизатору  $R_3$ . После того как дейтаграмма поступит на маршрутизатор  $R_3$ , последний анализирует ее содержимое и определяет, что дейтаграмма пришла по туннелю от маршрутизатора  $R_1$ . Затем устройство  $R_3$  расшифровывает область данных и восстанавливает исходную дейтаграмму. После этого просматривается локальная таблица маршрутизации и определяется, что дейтаграмма должна быть переслана устройству  $R_4$  для дальнейшей доставки конечному получателю.

## 20.5. Виртуальная частная сеть с локальными адресами

Применение технологии VPN предоставляет организациям такие же возможности адресации, как и технологии реальной частной сети. Если сетевые узлы в VPN не подключаются к глобальной сети Internet, то в VPN можно использовать произвольные IP-адреса. Если же сетевые узлы в VPN должны обмениваться данными с глобальной сетью Internet, можно использовать гибридную схему адресации. При такой схеме внутри виртуальной частной сети используются локальные адреса, а для организации через Internet туннеля между сетевыми центрами, их маршрутизаторам назначается по одному реальному IP-адресу (рис. 20.4)



Рис. 20.4. Гибридная адресация в виртуальной частной сети, два полностью защищенных сетевых центра которой соединены через глобальную сеть Internet. Компьютерам каждого сетевого центра назначены локальные адреса

Как показано на рис. 20.4, в сетевом центре 1 используется подсеть 10.1.0.0/16, а в сетевом центре 2 — подсеть 10.2.0.0/16. В такой системе необходимы только два реальных IP-адреса, которые используются для доступа маршрутизаторов  $R_1$  и  $R_2$  к глобальной сети Internet. В Таблицах маршрутизации компьютеров, принадлежащих сетевым центрам, используются локальные адреса. Таким образом, два реальных IP-адреса необходимы только для работы программы, управляющей туннелированием. Две копии этой программы запускаются на маршрутизаторах  $R_1$  и  $R_2$ .

В виртуальных частных сетях используется такая же структура адресации, как и в реальной частной сети. Сетевым узлам в полностью изолированной VPN могут назначаться произвольные адреса. Однако для доступа к глобальной сети Internet должна применяться гибридная система адресации с использованием реальных IP-адресов, действительных во всей сети Internet. Остается открытым вопрос: как может сетевой центр обеспечить доступ к глобальной сети Internet всем своим компьютерам без назначения каждому сетевому узлу реального IP-адреса? Существует два решения этой проблемы.

Первый подход состоит в использовании так называемого *шлюза уровня приложений* (*application gateway*). Он позволяет сетевым узлам осуществлять доступ к службам Internet без установки непосредственного соединения с ними на уровне

протокола IP. В каждом сетевом центре имеется многоадресный узел, подключенный как к глобальной сети Internet (при этом одному из его интерфейсов назначается реальный IP-адрес), так и к внутренней сети (второму интерфейсу назначается локальный IP-адрес). На этом узле запускается ряд прикладных программ, которые называются *шлюзами уровня приложений*, каждая из которых управляет работой одной из служб. Узлы в сетевом центре не отправляют дейтаграммы непосредственно в глобальную сеть Internet. Они отправляют каждый запрос многоадресному узлу, точнее, соответствующему шлюзу уровня приложений, который в свою очередь обращается к службам глобальной сети Internet, а затем пересыпает полученную от них информацию по внутренней сети отправителю запроса. Пример шлюза уровня приложений приведен в главе 27, “Приложения: система электронной почты (SMTP, POP, IMAP, MIME)”. Там описан шлюз электронной почты (e-mail gateway), который позволяет пересыпать почтовые сообщения между внешними и внутренними сетевыми узлами.

Основное преимущество использования шлюза уровня приложений заключается в том, что такая система может работать без изменения низкоуровневой сетевой инфраструктуры или системы адресации. Основной недостаток этого подхода состоит в том, что его нельзя считать универсальным и общеприменимым. Напрашивается следующий вывод.

*Каждый шлюз уровня приложений управляет работой только одной из служб, поэтому для поддержки нескольких служб приходится использовать несколько разных шлюзов.*

Хотя при соблюдении определенных условий шлюзы уровня приложений могут быть полезны, они не позволяют решить проблему доступа в Internet в общем. Поэтому и было придумано второе решение.

## 20.6. Преобразование сетевых адресов (NAT)

Для решения общей проблемы доступа между узлами сетевого центра и остальной частью глобальной сети Internet на уровне протокола IP была разработана технология, которая в сетевой терминологии называется *преобразованием сетевого адреса* (*Network Address Translation*, или *NAT*). При использовании этой технологии не требуется, чтобы каждый узел в сетевом центре имел реальный IP-адрес. Для ее реализации необходимо, чтобы сетевой центр имел по крайней мере одно соединение с глобальной сетью Internet и один реальный IP-адрес,  $G$ , действительный во всей глобальной сети Internet. Адрес  $G$  предназначен для компьютера (многоадресного узла или маршрутизатора), через который сетевой центр подключается к глобальной сети Internet и на котором запущена программа поддержки NAT. Компьютер, на котором запущены средства NAT, называют *NAT-блоком* (*NAT box*). Через NAT-блок проходят все дейтаграммы, следующие из сетевого центра в Internet и, наоборот, из Internet в сетевой центр.

Программа поддержки NAT выполняет преобразование адресов как в исходящих, так и во входящих дейтаграммах. При этом адрес отправителя каждой исходящей дейтаграммы заменяется на  $G$ , а в каждой входящей дейтаграмме выполняется обратное преобразование (т.е. адрес получателя  $G$  заменяется на локальный адрес соответствующего узла сетевого центра). Таким образом, с точки зрения внешнего получателя все дейтаграммы, отправленные из сетевого центра, приходят от адреса NAT-блока, а все отклики возвращаются на NAT-блок. Для внутренних сетевых узлов NAT-блок является маршрутизатором, который имеет доступ к глобальной сети Internet.

Преимущество технологии NAT заключается в том, что она является универсальной и прозрачной для узлов сети. Кроме того, использование NAT предпочтительнее, чем использование шлюза уровня приложений, поскольку она позволяет обеспечить доступ любого внутреннего узла сети к любой службе глобальной сети Internet. Технология NAT является прозрачной, поскольку она позволяет любому внутреннему узлу сети отправлять и получать дейтаграммы, используя локальные (т.е. немаршрутизуемые) адреса. Подводя итоги можно сказать:

*технология преобразования сетевого адреса (Network Address Translation, или NAT) обеспечивает для любого узла сети прозрачный доступ к глобальной сети Internet на уровне протокола IP помошью локальных (т.е. немаршрутизуемых) адресов.*

## 20.7. Создание таблицы преобразования сетевых адресов

При рассмотрении технологии NAT был опущен важный аспект, а именно: как NAT определяет, какому из внутренних узлов сети предназначена дейтаграмма, пришедшая из глобальной сети Internet. Для решения этой задачи в NAT поддерживается специальная таблица, которая используется для преобразования адресов. В каждом элементе таблицы указывается два значения: IP-адрес узла глобальной сети Internet и внутренний IP-адрес узла сетевого центра. При получении дейтаграммы из глобальной сети Internet программа NAT ищет адрес ее отправителя в таблице преобразования адресов. Когда нужный элемент таблицы найден, из него извлекается соответствующий адрес внутреннего узла сетевого центра, который помещается в поле адреса получателя дейтаграммы<sup>3</sup>, после чего дейтаграмма пересыпается получателю по локальной сети.

Понятно, что таблица преобразования адресов должна составляться заблаговременно. В противном случае при получении из Internet дейтаграммы программа NAT никак не сможет определить адрес внутреннего узла сети, на который нужно ее переправить. Поэтому возникает вопрос: как и когда надо инициализировать таблицу? Существует несколько методов инициализации.

- *Ручная инициализация.* Таблица преобразования адресов создается вручную администратором сети до начала работы программы NAT.
- *Исходящие дейтаграммы.* Таблица создается одновременно с отправкой дейтаграмм. При поступлении дейтаграммы от внутреннего узла сети программа NAT создает элемент в таблице преобразования, в котором фиксируется локальный адрес отправителя и глобальный адрес получателя.
- *Входящие запросы на преобразование имен.* Таблица создается одновременно с обработкой входящих запросов на преобразование доменных имен. Когда узел в глобальной сети Internet посылает запрос на преобразование доменного имени, соответствующего внутреннему узлу сети, в IP-адрес<sup>4</sup>, сервер имен создает элемент в таблице NAT, а затем отвечает на запрос, отсылая адрес G. Таким образом, для внешнего получателя все выглядит так, как будто именам внутренних узлов сети соответствует один глобальный адрес G.

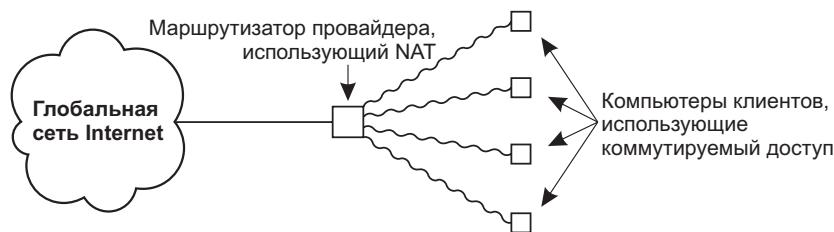
Каждый из методов инициализации имеет преимущества и недостатки. Ручная инициализация обеспечивает постоянную привязку адресов и позволяет отправлять IP-дейтаграммы в любое время в любом направлении. Использование

<sup>3</sup> Естественно, всякий раз при замене IP-адреса в заголовке дейтаграммы программа NAT должна пересчитать значение контрольной суммы заголовка.

<sup>4</sup> О том, как работает система доменных имен (Domain Name System, или DNS), речь пойдет в главе 24, “Система доменных имен (DNS)”.

исходящих дейтаграмм позволяет автоматизировать процесс инициализации таблицы преобразования адресов, однако, с другой стороны, при использовании этого метода невозможно инициировать соединение извне. Применение третьего метода инициализации требует модификации программного обеспечения, обрабатывающего запросы на преобразование доменных имен. Этот метод позволяет инициировать соединение из-за пределов сетевого центра, однако он работает только в том случае, если отправитель посыпает запрос на преобразование доменного имени до отправки дейтаграмм.

В большинстве реализаций NAT для инициализации таблицы преобразования адресов используется метод исходящих дейтаграмм. Он наиболее популярен среди поставщиков услуг Internet. Чтобы понять, почему, рассмотрим мелкого провайдера Internet, обеспечивающего своим клиентам доступ к сети по коммутируемому соединению с помощью обычного модема. Структурная схема такой системы показана на рис. 20.5.



*Рис. 20.5. Структурная схема сети мелкого провайдера Internet, в которой используется технология NAT, и коммутируемый доступ к сети со стороны клиентов. Применение NAT позволяет провайдеру назначать клиентам локальные IP-адреса*

Каждый раз при подключении клиента к сети провайдера ему назначается IP-адрес. Технология NAT позволяет провайдеру Internet присваивать всем своим клиентам локальные IP-адреса. Например, первому клиенту присваивается адрес 10.0.0.1, второму — 10.0.0.2 и т.д. Когда клиент отправляет дейтаграмму получателю, расположенному в сети Internet, программа NAT использует содержимое ее заголовка для инициализации таблицы преобразования адресов.

## 20.8. Многоадресная NAT

Выше была описана работа технологии NAT в самом простом случае, когда между внешними и внутренними адресами существует однозначное соответствие. Другими словами, однозначное соответствие позволяет только одному компьютеру в сетевом центре получить доступ в произвольный момент времени к некоторой машине в глобальной сети Internet. Однако на практике используются более сложные формы NAT, которые позволяют различным узлам сетевого центра получать доступ к одному внешнему получателю.

В одном из вариантов NAT одновременный доступ клиентов осуществляется за счет сохранения принципа однозначного соответствия адресов в NAT-блоке, которому дополнительно назначено несколько реальных IP-адресов. Этот метод называется *многоадресной NAT*. Суть его состоит в том, что NAT-блоку назначается  $k$  реальных IP-адресов, действительных во всей глобальной сети Internet,  $G_1, G_2, \dots, G_k$ . Когда первый внутренний узел сети отправляет дейтаграмму получателю в Internet, NAT-блок выбирает адрес  $G_1$ , от которого она будет послана в Internet, добавляет элемент в таблицу преобразования адресов и отправляет дейтаграмму получателю. Если другой узел сети попытается отправить дейтаграмму

тому же получателю, NAT-блок выберет адрес  $G_2$ , и т.д. Таким образом, *многоадресная NAT* позволяет  $k$  внутренним узлам сети одновременно получать доступ к одному получателю.

## 20.9. Преобразование сетевого адреса с распределением по портам

Другой широко распространенный вариант NAT обеспечивает одновременный доступ клиентов путем преобразования не только IP-адресов, но и номеров портов протоколов TCP или UDP. Иногда этот метод называют *преобразованием сетевого адреса и порта* (*Network Address Port Translation*, или *NAPT*). В нем таблица преобразования сетевых адресов расширяется за счет присоединения дополнительных полей. Кроме пары IP-адресов (отправителя и получателя), в таблице указывается пара номеров портов (локального и удаленного), а также номер порта протокола, используемый NAT-блоком. Пример таблицы преобразования адресов приведен в табл. 20.1.

**Таблица 20.1. Пример таблицы преобразования адресов, используемой в NAPT**

Локальный адрес	Локальный порт	Внешний адрес	Внешний порт	Порт NAT	Используемый протокол
10.0.0.5	21023	128.10.19.20	80	14003	tcp
10.0.0.1	386	128.10.19.20	80	14010	tcp
10.0.2.6	26600	207.200.75.200	21	14012	tcp
10.0.0.3	1274	128.210.1.5	80	14007	tcp

В табл. 20.1 показаны элементы для четырех внутренних компьютеров, которые в настоящий момент имеют доступ к Internet. Для всех соединений используется протокол TCP. Интересно, что в таблице показаны два внутренних узла сети — 10.0.0.5 и 10.0.0.1, которые оба осуществляют подключение через порт 80 протокола (Web-сервер) к компьютеру 128.10.19.20. В данном случае оказалось, что номера двух локальных портов, использующихся для двух соединений, отличаются. Однако уникальность номера локального порта нельзя гарантировать, поскольку два внутренних узла сети могут совершенно случайно выбрать один и тот же номер локального порта. Поэтому, чтобы избежать возможного конфликта, в NAT каждому соединению с сетью Internet присваивается уникальный номер порта. Напомним, что в протоколе TCP каждое соединение идентифицируется набором из четырех взаимосвязанных величин (4-мерный кортеж), которые представляют собой IP-адреса и номера портов отправителя и конечного получателя. Таким образом, первые два элемента в таблице преобразования адресов соответствуют TCP-соединениям, которые внутренние узлы сети идентифицируют при помощи следующих 4-мерных кортежей:

$$(10.0.0.5, 21023, 128.10.19.20, 80)$$

$$(10.0.0.1, 386, 128.10.19.20, 80)$$

В свою очередь, компьютер в сети Internet, получивший дейтаграммы после преобразования адреса с помощью механизма NAPT, идентифицирует те же два соединения с помощью двух других 4-мерных кортежей:

$$(G, 14003, 128.10.19.20, 80)$$

$$(G, 14010, 128.10.19.20, 80),$$

где  $G$  — реальный IP-адрес NAT-блока.

Основное преимущество технологии NAPT — ее универсальность, которая достигается за счет применения лишь одного реального IP-адреса. Ее основной недостаток состоит в том, что в процессе взаимодействия стороны должны использовать только протоколы TCP или UDP. Только в этом случае технология NAPT позволяет без взаимных помех внутреннему компьютеру получить доступ к нескольким внешним компьютерам, а также некоторым внутренним компьютерам получить доступ к одному и тому же внешнему компьютеру. Под номером порта отводится 16 бит, поэтому одновременно до  $2^{16}$  пар приложений могут одновременно взаимодействовать между собой. Можно сделать следующий вывод.

*Существует несколько вариантов реализации технологии NAT, включая ее популярную форму NAPT, в которой преобразование IP-адреса выполняется на основе номеров портов протокола TCP или UDP.*

## 20.10. Взаимодействие между NAT и ICMP

Даже самые простые изменения, внесенные в структуру IP-адресации, могут привести к самым неожиданным последствиям в протоколах более высокого уровня. В частности, чтобы сохранить иллюзию “прозрачности” технологии NAT, в ней должна поддерживаться обработка ICMP-сообщений. Предположим, например, что внутренний сетевой узел использует программу ping для проверки доступности получателя в сети Internet. При этом программа посыпает получателю ICMP-запрос на эхо и переходит в режим ожидания отклика. Таким образом, в технологии NAT входящие ICMP-сообщения, являющиеся откликами на посланные ICMP-запросы, должны пересыпаться на соответствующие узлы сети. Однако на самом деле далеко не все ICMP-сообщения, полученные из Internet, пересыпаются локальным получателем. Например, если NAT-блок содержит некорректные маршруты, то ICMP-сообщение о *перенаправлении* должно быть обработано локально. Поэтому когда ICMP-сообщение приходит из сети Internet, NAT-блок должен первым делом определить, нужно ли обработать его локально или отправить на внутренний узел сети для дальнейшей обработки. Прежде чем направить ICMP-сообщение на внутренний узел сети, NAT-блок должен преобразовать в нем адреса.

Чтобы понять необходимость преобразования адресов, рассмотрим ICMP-сообщение о недоступности получателя. В нем содержится заголовок дейтаграммы,  $D$ , которая вызвала ошибку. К сожалению, NAT-блок преобразует адреса в заголовке дейтаграммы  $D$  до ее отправления. Поэтому в поле адреса ее отправителя не будет указан IP-адрес внутреннего узла сети. Следовательно, перед пересыпкой ICMP-сообщения внутреннему узлу сети, NAT-блок должен открыть его содержимое и преобразовать адреса в заголовке дейтаграммы  $D$  так, чтобы они соответствовали адресам, используемым внутренним узлом сети. После внесения изменений NAT-блок должен пересчитать контрольную сумму заголовка дейтаграммы  $D$ , контрольную сумму заголовка ICMP-сообщения и контрольную сумму заголовка внешней дейтаграммы.

## 20.11. Взаимодействие между NAT и прикладными программами

В предыдущем разделе было показано, что поддержка протокола ICMP в NAT-блоке существенно усложняет его реализацию. Однако еще хуже дело обстоит с поддержкой протоколов уровня приложений. Через NAT не будут работать те приложения, которое пересыпают IP-адреса или номера портов в качестве данных. Например, когда две программы используют протокол передачи

*файлов (File Transfer Protocol, или FTP), описанный в главе 26, “Приложения: передача файлов и удаленный доступ к ним (FTP, TFTP, NFS)”, для соединения между собой они используют протокол TCP. Согласно протоколу (FTP), одна из программ должна получить на локальной машине номер TCP-порта, преобразовать его в ASCII-код и отправить результат через TCP-соединение другой программе. Если пакеты между этими программами по пути от внутреннего узла сети до узла в сети Internet проходят через NAPT-блок, то номер порта, помещенный в поток данных внутренним узлом сети, должен быть изменен на номер порта, выбранный NAPT-блоком. По сути, если NAPT-блок не сможет открыть поток данных и изменить в нем номер порта, то программы не смогут обменяться данными по протоколу FTP. На сегодняшний день существуют реализации NAT, которые автоматически распознают дейтаграммы популярных протоколов, наподобие FTP, и делают необходимые изменения в потоке данных. Однако существуют приложения, которые не могут обмениваться данными через NAT. Резюме такое.*

*NAT-блок изменяет содержимое ICMP-сообщений, а также дейтаграмм, относящихся к протоколам более высокого уровня. Поэтому, за исключением нескольких стандартных протоколов наподобие FTP, протоколы уровня приложений, которые передают IP-адреса или номера портов в потоке данных, не будут правильно работать при использовании NAT.*

Передаваемые в потоке данных элементы, которые должны быть изменены NAPT-блоком, увеличивают сложность его реализации по двум причинам. Во-первых, это означает, что NAPT-блок должен располагать подробной информацией о формате передаваемых каждым приложением данных (для того, чтобы “на ходу” внести в них изменения). Во-вторых, если приложения передают номера портов в ASCII-коде, как в случае протокола FTP, то их изменение может повлечь за собой изменение общего количества передаваемых через TCP-соединение данных. Вставить даже один дополнительный октет в поток данных крайне трудно, поскольку каждый октет в потоке имеет порядковый номер. Поскольку отправитель не знает, что в процессе передачи потока данных в него были вставлены дополнительные данные, он продолжает присваивать порядковые номера октетам без учета этих данных. Более того, получатель будет воспринимать дополнительные данные как часть потока, передаваемого отправителем и присыпать на них сигналы подтверждения приема. Таким образом, после вставки NAT-блоком дополнительных данных должны быть изменены порядковые номера октетов в каждом выходящем сегменте и каждом входящем сигнале подтверждения приема.

## 20.12. Абстрактные домены адресов

В предыдущих разделах технология NAT была описана с точки зрения подключения частных сетей к глобальной сети Internet. На самом деле NAT может использоваться для взаимодействия между любыми двумя *доменами адресов*. Следовательно, технология NAT может использоваться для связи двух корпоративных частных сетей, каждой из которых назначен один и тот же блок локальных адресов, например 10.0.0.0. Более важно то, что NAT может использоваться на двух уровнях: как между локальными доменами адресов клиента и провайдера, так и между доменом адресов провайдера и глобальной сетью Internet. Наконец, технология NAT может использоваться при создании гибридных виртуальных частных сетей с использованием локальных адресов в пределах организации. При этом NAT позволяет каждому узлу частной сети получить доступ к глобальной сети Internet.

В качестве примера использования технологии NAT на различных уровнях, рассмотрим домашнюю локальную сеть, связывающую несколько компьютеров. При этом компьютерам домашней сети можно назначить локальные адреса, а для связи ее с корпоративной локальной сетью использовать NAT. В корпоративной сети также могут использоваться локальные адреса. При этом для связи ее с глобальной сетью Internet также применяется NAT.

### 20.13. Программы *slirp* и *masquerade*

В настоящее время стали особенно популярными две реализации технологии NAT. Обе были разработаны для операционной системы UNIX. Исходный код программы *slirp* входил в поставку Unix версии 4.4BSD. Эта программа была предназначена для обеспечения связи пользовательских компьютеров с Internet по коммутируемому соединению (как показано на рис. 20.5). Поэтому, кроме NAT, она поддерживает протокол двухточечного соединения PPP. Программа *slirp* должна запускаться на компьютере, имеющем реальный IP-адрес, постоянную связь с Internet и несколько модемов для подключения удаленных пользователей. Преимущество *slirp* заключается в том, что она может предоставить доступ к сети Internet через учетную запись пользователя системы UNIX. При этом процесс установки соединения выглядит следующим образом. Пользователь с удаленного компьютера по модему подключается к серверу провайдера и вводит свое имя и пароль. После этого на сервере запускается программа *slirp*, которая переводит сеанс связи с тестового режима (ввода ASCII-команд оболочки UNIX) на бинарный и запускает программу поддержки протокола PPP. Для получения доступа к Internet, например к службе Web, пользователь на своем компьютере также должен запустить программу PPP.

В программе *slirp* реализована технология NAPT, т.е. получатели в локальной сети для поступивших дейтаграмм определяются по номерам портов. При отправке дейтаграмм в Internet программа *slirp* заменяет в дейтаграмме как IP-адрес отправителя, так и его номер порта. Все это позволяет нескольким компьютерам локальной сети одновременно получить доступ к сети Internet посредством запуска одной копии программы *slirp* на сервер UNIX.

Другая популярная реализация технологии NAT была разработана для операционной системы LINUX. Программа называется *masquerade* и обеспечивает поддержку NAPT. В отличие от *slirp*, программа *masquerade* не требует, чтобы компьютеры пользователей подключались к серверу через modem. Более того, перед запуском этой программы не нужно, чтобы пользователь регистрировался в системе UNIX. В программе *masquerade* предусмотрено большое количество параметров. Ее можно сконфигурировать так, чтобы она работала как маршрутизатор между двумя сетями, поддерживала различные разновидности NAT, включая те, что были рассмотрены в этой главе, в том числе и многоадресную.

### 20.14. Резюме

Несмотря на то что частная сеть гарантирует секретность информации, стоимость ее развертывания может быть очень высокой. Более дешевой альтернативой является технология виртуальной частной сети (VPN). Она позволяет организации соединить свои сетевые центры через глобальную сеть Internet. При этом для сохранения секретности потока данных, протекающего между сетевыми центрами, дейтаграммы шифруются. Как и реальная частная сеть, VPN может быть полностью или частично изолирована от Internet. В первом случае узлам сети присваиваются локальные адреса, а во втором — используется гибридная структура,

которая позволяет узлам сети устанавливать соединение с получателями, находящимися в глобальной сети Internet.

Существует два способа, с помощью которых можно обеспечить соединение между узлами сети, находящимися в различных доменах адресов: шлюз уровня приложений (application gateway) и преобразование сетевого адреса (NAT). Шлюз уровня приложений выполняет функции посредника, или proxy-сервера. Он получает запрос от узла сети, принадлежащего одному домену, пересыпает его к получателю, находящемуся в другом домене, после чего возвращает результат исходному узлу сети. При этом для каждой службы Internet на proxy-сервере должен быть запущен свой шлюз уровня приложений.

Технология преобразования сетевого адреса обеспечивает “прозрачный” доступ от сетевого узла, имеющего локальный адрес, к глобальной сети Internet на уровне протокола IP. Технология NAT особенно популярна среди провайдеров Internet, поскольку позволяет клиентам получать доступ к произвольным службам сети Internet с помощью локальных IP-адресов. Однако через NAT-блок не будут работать те приложения, которые передают получателю в потоке данных IP-адреса или номера портов, если конечно не запрограммировать NAT-блок так, чтобы он автоматически распознавал эти данные и вносил в них необходимые изменения. В большинстве реализаций NAT распознается только несколько стандартных служб Internet.

## Материал для дальнейшего изучения

Технология виртуальной частной сети поддерживается во многих маршрутизаторах и программных продуктах сторонних разработчиков. При этом каждый производитель выбирает свои методы шифрования и систему адресации. Для получения дополнительной информации обратитесь к документации, поставляемой с соответствующим оборудованием или программой.

На сегодняшний день существует несколько коммерческих версий NAT. С целями и задачами группы IETF, разрабатывающей NAT, можно ознакомиться по адресу: <http://www.ietf.org/html.charters/nat-charter.html>. Кроме того, терминология NAT описана Шришурешом (Srisuresh) и Холдридже (Holdrege) в [RFC 2663]. В хранилище предварительных стандартов (черновиков) Internet по адресу: <http://www.ietf.org/ID.html> содержится несколько документов, относящихся к NAT.

Подробную информацию о программе masquerade можно получить, обратившись к ее документации, которая находится по адресу <http://ipmasq.cjb.net>.

Документацию по программе slirp можно найти в Internet по адресу <http://blitzen.canberra.edu.au/slirp>.

## Упражнения

- 20.1. Почему при пересылке одних и тех же данных через сеть Internet и VPN в последнем случае передается существенно большее количество пакетов? (*Подсказка.* Рассмотрите процесс инкапсуляции дейтаграмм.)
- 20.2. Обратитесь к документации по программе slirp и изучите процесс трансформации номеров портов (port redirection). Зачем она нужна?
- 20.3. Какие могут возникнуть проблемы если связать три домена адресов посредством двух блоков NAT?
- 20.4. Из предыдущего упражнения определите, сколько раз будет преобразовываться адрес получателя, а сколько — адрес отправителя.

- 20.5.** Предположим, что ICMP-сообщение о недоступности узла сети посыпается через два NAT-блока, связывающих три домена адресов. Сколько раз будут преобразовываться содержащиеся в нем IP-адреса и номера портов?
- 20.6.** Представьте себе, что вам поручили создать новую сеть Internet, работающую параллельно с существующей сетью Internet, в которой используются два одинаковых пространства IP-адресов. Можно ли с помощью технологии NAT соединить две сети Internet произвольного размера, использующих одинаковое адресное пространство? Если — да, то объясните, как это сделать. Если — нет, то объясните почему.
- 20.7.** Является ли технология NAT полностью прозрачной для узла сети? Чтобы ответить на этот вопрос, попробуйте найти последовательность пакетов, которые узел сети должен передать, чтобы определить, расположен ли он за NAT-блоком.
- 20.8.** Каковы преимущества и недостатки применения технологии NAT в VPN?
- 20.9.** Установите на своем сервере копию программы slirp и выполните измерение параметров сети. Вносит ли программа slirp дополнительную задержку при обработке дейтаграмм? Если — да, то почему?
- 20.10.** Установите одну из программ, реализующих технологию NAT, на своем UNIX-сервере и свяжите с ее помощью домен локальных адресов и сеть Internet. Какие широко распространенные службы Internet будут при этом работать, а какие нет?
- 20.11.** Существует еще одна разновидность технологии NAT, которая называется *двойной NAT* (*twice NAT*). Она позволяет любому узлу сети, находящемуся с любой стороны NAT-блока, инициализировать соединение. Ознакомьтесь в литературе с ее описанием. Каким образом в технологии двойной NAT гарантируется непротиворечивость преобразований IP-адресов? Будет ли данная технология полностью прозрачна для всех узлов сети, если для связи трех доменов адресов используются два блока двойной NAT?

# 21

## Модель взаимодействия клиент/сервер

### 21.1. Введение

В предыдущих главах были описаны детали технологии TCP/IP, протоколы, обеспечивающие работу основных служб сети, структура системы маршрутизации и объяснена необходимость обмена маршрутной информацией. Теперь, когда понятны основополагающие принципы, можно перейти к рассмотрению прикладных программ, работа которых тесно связана с функционированием объединенной сети TCP/IP. Несмотря на то что рассматриваемые в этой главе примеры приложений представляют практическую ценность и интересны сами по себе, основное внимание уделяется не им, а моделям взаимодействия между обменивающимися информацией прикладными программами. В основу модели взаимодействия между “сотрудничающими” приложениями положен принцип *клиент/сервер*<sup>1</sup>. Взаимодействие по принципу клиент/сервер положено в основу работы большинства сетей передачи данных. Оно является фундаментальным, потому что позволяет понять суть алгоритмов работы распределенных систем. В этой главе в общих чертах рассмотрена модель взаимодействия между клиентом и сервером, а в следующих главах изложение материала будет расширено за счет рассмотрения конкретных примеров.

### 21.2. Модель взаимодействия клиент/сервер

Под термином *сервер* мы будем понимать любую программу, в которой реализована служба, доступная через сеть. Сервер принимает запрос по сети, выполняет необходимую обработку и возвращает результат отославшей запрос программе. В простейшем случае каждый запрос доставляется на сервер в виде одной IP-дейтаграммы, а сервер возвращает ответ в другой дейтаграмме.

Выполняющаяся программа становится *клиентом*, когда она посыпает запрос на сервер и ожидает от него ответа. Поскольку модель клиент/сервер является удобным и естественным средством расширения возможностей межпроцессного взаимодействия на одной машине, на ее основе легко создавать программы, обменивающиеся данными друг с другом.

Задачи, выполняемые серверами, могут быть как очень простыми, так и невероятно сложными. Например, *сервер текущего времени* просто возвращает текущую дату и время всякий раз, когда клиент посыпает ему запрос. *Web-сервер* по-

---

<sup>1</sup> В современной литературе вместо термина *клиент/сервер* иногда используют термин *приложение/сервер*. При этом суть подхода не меняется.

лучает запросы от броузера на доставку содержимого той или иной Web-страницы. Затем сервер получает копию файла этой страницы и возвращает ее броузеру.

Обычно серверы реализуются в виде прикладных программ<sup>2</sup>. Преимущество такого подхода заключается в том, что прикладные программы могут выполняться в любой вычислительной системе, которая поддерживает взаимодействие по протоколу TCP/IP. Таким образом, сервер для конкретной службы может выполняться как в системе, работающей в режиме разделения времени наряду с другими программами, так и на персональном компьютере. Одна и та же служба может быть реализована в виде нескольких серверов, которые могут быть запущены как на одной, так и на разных машинах. Чаще всего системные администраторы запускают копии одного сервера на разных компьютерах, чтобы повысить надежность системы или улучшить ее производительность. Если для работы программы-сервера выделяется целый компьютер, то под термином “сервер” может также подразумеваться и этот компьютер. Поэтому нередко можно услышать: “Машина А — это наш файловый сервер”.

### 21.3. Простой пример: эхо-сервер UDP

В простейшем случае для взаимодействия по технологии клиент/сервер используется ненадежная служба доставки дейтаграмм (протокол UDP), с помощью которой сообщения передаются от клиента на сервер и обратно. В качестве примера давайте рассмотрим *эхо-сервер*, использующий протокол UDP. Принцип его работы очень прост и проиллюстрирован на рис. 21.1. Перед началом работы процесс, выполняющийся на сервере, посыпает запрос операционной системе на право использования порта протокола UDP, зарезервированного для службы эха (порт UDP номер 7). Как только процесс, обслуживающий наш эхо-сервер, получит разрешение на использование порта, он входит в бесконечный цикл, в котором выполняются следующие три действия.

1. Ожидается поступление дейтаграммы на порт эха.
2. После получения дейтаграммы сервер должен поменять местами адреса отправителя и получателя<sup>3</sup> (включая IP-адреса и номера портов протокола UDP).
3. Сервер возвращает дейтаграмму отправителю.

Обычно клиентская программа запускается на другом компьютере (хотя это и не обязательно). Сначала она запрашивает у операционной системы номер одного из неиспользуемых портов протокола UDP. Затем программа-клиент посыпает запрос эхо-серверу и переходит в состояние ожидания ответа от него. В нашем случае сервер вернет клиенту те же данные, которые он отправил серверу.

Служба эха, использующая протокол UDP, позволяет проиллюстрировать два важных момента, которые в целом справедливы для любого взаимодействия по технологии клиент/сервер. Первый из них — сдвиг по времени между началом работы сервера и клиента.

*Сервер запускается до начала взаимодействия с клиентом и (обычно) продолжает работать в непрерывном режиме (т.е. принимать запросы и отсылать на них ответы). Клиент — это любая программа, которая посыпает запрос серверу и ожидает от него ответа. Обычно клиентская программа посыпает серверу ограниченное количество запросов, после чего завершает свое выполнение.*

<sup>2</sup> В большинстве современных операционных систем запущенная прикладная программа называется *процессом, пользовательским процессом, или задачей*.

<sup>3</sup> В одном из упражнений в конце главы предложено рассмотреть этот этап более подробно.

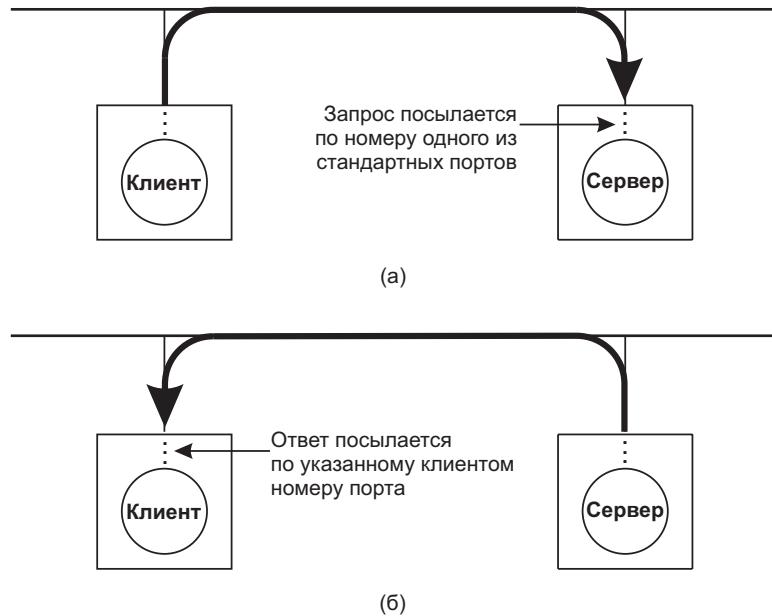


Рис. 21.1. Иллюстрация принципа работы модели клиент/сервер на основе эхо-сервера протокола UDP. Клиент посылает запрос серверу по его IP-адресу и одному из стандартных номеров портов протокола UDP (а); сервер возвращает ответ клиенту по указанному им номеру порта протокола UDP (б)

Второй момент, который является более техническим, касается использования зарезервированных и свободных номеров порта.

*Сервер ожидает поступления запроса по одному из стандартных номеров портов, зарезервированных для той службы, которую он поддерживает. Для взаимодействия с сервером клиент запрашивает у операционной системы один из свободных номеров портов.*

При взаимодействии по методу клиент/сервер только один из двух номеров портов должен быть зарезервирован. Для упрощения процедуры создания приложений-клиентов и приложений-серверов каждой стандартной службе присвоен уникальный номер порта.

Кто использует эхо-службу? Это не та служба, которая могла бы быть интересной для обычного пользователя. Однако программисты, которые занимаются проектированием, практической реализацией, оценкой производительности и модификацией программного обеспечения для работы в сети, а также сетевые администраторы, которые тестируют маршруты, выявляют и устраняют проблемы связи, часто используют эхо-серверы для тестирования. Эхо-служба может использоваться, например, чтобы определить, возможно ли получить доступ к удаленной машине.

## 21.4. Служба времени и даты

Описанный выше эхо-сервер чрезвычайно прост. Для реализации его серверной и клиентской частей требуется написать совсем немного программного кода (конечно, при условии, что операционная система предоставляет удобные средства

для доступа к низкоуровневым протоколам UDP/IP). Второй пример, сервер времени, показывает, что даже в основе простого взаимодействия по методу клиент/сервер можно создать достаточно полезную службу. Сервер времени предназначен для решения проблемы установки часов реального времени, которые являются одним из устройств компьютера. В них хранится текущая дата и время, которую могут опросить прикладные программы. После того как время на часах реального времени установлено, они продолжают “идти” точно так же, как и наручные часы.

В некоторых операционных системах проблема установки часов реального времени решается очень просто — дату и время вводит оператор при начальной загрузке компьютера. После этого операционная система периодически (например, несколько раз в секунду) увеличивает показания внутреннего таймера. Когда прикладная программа запрашивает дату или время, операционная система обращается к таймеру и представляет дату и время в удобочитаемой для человека форме. Взаимодействие типа клиент/сервер может использоваться для автоматической установки системных часов при начальной загрузке машины. Для этого администратор должен запустить на одной из машин (обычно той, у которой часы идут наиболее точно) сервер реального времени. Тогда при загрузке других машин для получения текущего времени они должны обратиться к серверу.

#### 21.4.1. Представление даты и времени

В каком виде операционная система должна хранить дату и реальное время? Чаще всего дата и время хранятся в виде количества секунд, прошедших с некой эпохальной даты. Например, в операционной системе UNIX в качестве точки отсчета даты и времени используется нулевая секунда 1 января 1970 года. В семействе протоколов TCP/IP дата и время также отсчитываются в виде количества секунд, прошедших от начала “эпохи”. Для протокола TCP/IP началом “эпохи” является нулевая секунда 1 января 1900 года, а время хранится в виде 32-битового целого числа. В таком виде могут быть представлены все даты в ближайшем будущем.

Хранение даты в виде количества секунд, прошедших от начала “эпохи”, является очень компактным и позволяет легко сравнивать время. Такое представление связывает дату и время и позволяет измерять текущее время путем прращения одного двоичного целого числа.

#### 21.4.2. Местное и универсальное время

К какому часовому поясу следует относить текущее время, отсчитываемое от эпохальной даты? Очевидно, что при взаимодействии двух компьютеров, находящихся на значительном расстоянии, использование часовогопояса какого-либо из них становится проблематичным. Поэтому они должны договориться об использовании стандартного часовогопояса, чтобы поддерживаемые значения даты и времени были сопоставимы. Таким образом, кроме определения формата представления времени и выбора эпохальной даты, в стандарте сервера времени протокола TCP/IP указывается, что все значения должны отсчитываться относительно определенного часовогопояса. Таким часовым поясом является зона, через которую проходит нулевой (Гринвичский) меридиан. Поэтому первоначально это время называли *всемирным* или *средним временем по Гринвичу* (Greenwich Mean Time, GMT). Однако теперь его называют *всемирным скординированным временем* (*universal time coordinated, UTC*), или просто *всемирным временем*.

Взаимодействие между клиентом и сервером, на котором запущена служба времени, происходит почти так же, как и между клиентом и эхо-сервером, рассмотренным выше. Запущенное на машине сервера приложение получает разрешение на использование зарезервированного для службы времени номера порта.

После этого программа переходит в состояние ожидания поступления UDP-сообщения непосредственно на этот порт. В ответ на полученное сообщение программа посыпает UDP-сообщение, содержащее текущее время в виде целого 32-битового числа. Можно сделать такой вывод.

*Отправка дейтаграммы серверу времени эквивалентна созданию запроса на получение текущего времени. В ответ на это сервер возвращает UDP-сообщение, содержащее текущее время.*

## 21.5. Сложность серверных программ

До сих пор приводились примеры только простых серверных программ (или серверов), которые работали последовательно. Другими словами, программа сервера обрабатывала в определенный момент времени только один запрос. Получив запрос, она формировалась ответ, отправляла его клиенту и возвращалась в исходное состояние. Только в исходном состоянии программа сервера могла проверить, прибыл ли другой запрос. Здесь неявно предполагается, что если программа сервера занята обработкой текущего запроса, то все вновь прибывающие запросы будут помещаться операционной системой в очередь на обработку. Очевидно, что длина очереди не может стать слишком большой, поскольку описываемый нами сервер выполняет элементарные действия очень быстро.

Процесс написания серверных приложений обычно гораздо сложнее, чем клиентских. Все дело в том, что серверные программы должны обслуживать большое количество одновременно поступающих запросов, причем для обработки отдельного запроса иногда требуется значительное время. В качестве примера рассмотрим сервер передачи файлов, отвечающий за копирование файла на другую машину, с которой был получен запрос. Как правило, серверы состоят из двух частей: отдельной главной программы, отвечающей за принятие новых запросов, и ряда подчиненных программ, отвечающих за обработку индивидуальных запросов. Главная программа сервера выполняет следующие действия.

- 1. Открытие порта.** Главная программа сервера открывает стандартный порт, через который к ней можно получить доступ.
- 2. Ожидание поступления запроса от клиента.** Главная программа сервера переходит в состояние ожидания до поступления нового запроса от клиента.
- 3. Выбор порта.** При необходимости главная программа сервера выделяет для поступившего запроса новый номер локального порта и информирует об этом клиента. (Ниже будет показано, что этот этап не нужен при использовании протокола TCP и в большинстве случаев использования протокола UDP).
- 4. Запуск подчиненной программы.** Главная программа сервера запускает независимую подчиненную программу, выполняющуюся параллельно с ней, которая должна закончить обработку поступившего запроса. Например, в системах UNIX она порождает параллельную ветвь (`fork`) для своего процесса. Обратите внимание, что подчиненная программа обрабатывает один (текущий) запрос и завершает работу. Она не ждет поступления запросов от других клиентов.
- 5. Продолжение работы.** Главная программа сервера возвращается в состояние ожидания и продолжает принимать новые запросы, в то время как вновь созданная подчиненная программа параллельно обрабатывают предыдущий запрос.

Поскольку главная программа сервера запускает подчиненную программу для каждого нового запроса, обработка поступающих запросов происходит параллельно. Поэтому запросы, для обработки которых требуется небольшое время, могут выполняться быстрее тех, для обработки которых требуется более длительное время, независимо от порядка их поступления. В качестве примера предположим, что первый клиент запрашивает у файлового сервера большой файл, передача которого будет длиться несколько минут. Если второй клиент запросит у сервера небольшой файл, передача которого будет длиться всего несколько секунд, то обработка второго запроса может начаться и закончиться, в то время как передача первого файла все еще будет продолжаться.

Кроме обработки параллельных запросов, серверные программы должны еще выполнять авторизацию клиентов и обеспечивать определенные защитные функции. Это связано с тем, что обычно серверные программы выполняются с самыми высокими привилегиями, поскольку они должны читать системные файлы, вести журналы и иметь доступ к защищенным данным. Кроме того, операционная система не должна ограничивать доступ серверной программы к файлам пользователей. Таким образом, серверы не могут “вслепую” выполнять запросы, поступающие от других машин. Поэтому каждый сервер берет на себя ответственность за выполнение доступа к системным ресурсам и определенных правил защиты.

И наконец, серверы должны уметь защищаться от неверно сформированных запросов или от запросов, которые могут привести к аварийному прекращению работы самой серверной программы. Здесь зачастую трудно предвидеть потенциальные проблемы. Например, один из проектов, выполнявшихся в университете Пердью, предусматривал разработку файлового сервера, который бы позволял компьютерам студентов получать доступ к файлам системы UNIX, работавшей в режиме разделения времени. Студенты обнаружили, что отправка серверу запроса на открытие файла `/dev/tty` приводила к аварийному прекращению работы серверной программы. Так происходило потому, что в системе UNIX этот файл связан с управляющим терминалом, с которым должна взаимодействовать программа. Сервер, запущенный при начальной загрузке системы, не имел такого терминала. После аварийного прекращения работы серверной программы клиенты не могли получить доступ к файлам, пока системный администратор не перезапускал серверную программу.

О более серьезном примере уязвимости сервера стало известно осенью 1988 года, когда студент Корнельского университета (Cornell University) написал программу *червя* (*worm*), которая атаковала компьютеры, подключенные к глобальной сети Internet. Запущенный на какой-либо машине червь искал в сети Internet компьютеры с серверами, с которыми он умел взаимодействовать, и использовал их для воссоздания себя в большом количестве копий. Один из методов нападения червя был основан на ошибке в программе сервера UNIX `fingerd`. Поскольку сервер не проверял поступающие запросы, червь посыпал некорректную строку входных данных, в результате обработки которой сервер перезаписывал части своих внутренних областей данных. Поскольку сервер выполнялся с самыми высокими привилегиями, такое неадекватное его поведение позволило червю создавать свои копии. Обсуждение серверных программ можно подытожить следующим образом.

*Создавать серверные программы, как правило, намного труднее, чем клиентские приложения. Несмотря на то что сервер можно реализовать в виде обычной прикладной программы, она должна контролировать доступ к системным ресурсам и обеспечивать определенные правила защиты от несанкционированного доступа того компьютера, на котором она выполняется. Кроме того, серверная программа должна защищать себя от всех возможных ошибочных запросов, поступающих со стороны клиентов.*

## **21.6. Сервер RARP**

В приведенных выше примерах взаимодействия типа клиент/сервер подразумевалось, что клиент должен знать полный адрес сервера. В главе 6, “Определение IP-адреса при начальной загрузке (RARP)”, был описан протокол RARP, который может служить примером взаимодействия типа клиент-сервер несколько иного типа. Вспомним, что клиентский компьютер может использовать протокол RARP для нахождения своего IP-адреса при начальной загрузке. Клиенты RARP не взаимодействуют непосредственно с сервером, а посылают запросы в широковещательном режиме. Ответ присыпает одна или несколько машин, на которых запущены серверы RARP. Каждая из машин возвращает пакет, содержащий запрашиваемую информацию.

Между сервером RARP и серверами эха или времени, использующими протокол UDP, имеются два существенных различия. Во-первых, пакеты RARP передаются непосредственно во фреймах физической сети, а не в IP-дейтаграммах. Таким образом, в отличие от эхо-сервера, использующего протокол UDP и позволяющего клиенту обращаться к серверу из любой точки объединенной сети, сервер RARP требует, чтобы клиент находился в той же физической сети. Во-вторых, сервер RARP не может быть реализован в виде прикладной программы. Серверы эха и времени могут быть запущены как обычные прикладные программы, поскольку они используют протокол UDP. В отличие от них сервер RARP нуждается в доступе к низкоуровневым аппаратным фреймам и зависит от используемого типа сетевого оборудования.

## **21.7. Альтернативные модели взаимодействия**

Существуют ли альтернативные механизмы взаимодействия, кроме рассмотренной выше модели клиент/сервер? В каких случаях их применение может считаться оправданным? Ответ на эти вопросы будет рассмотрен в данном разделе.

В модели клиент/сервер программы обычно выступают в качестве клиентов, когда им требуется получить от сервера данные. Однако иногда важно минимизировать количество таких взаимодействий. В качестве примера рассмотрим протокол ARP, который был описан в главе 5, “Преобразование IP-адресов в физические адреса (ARP)”. Для получения адресной привязки в нем используется модифицированная форма взаимодействия типа клиент/сервер. Клиентские машины помещают ответы сервера ARP в свой внутренний кэш, благодаря чему увеличивается эффективность обработки последующих запросов. Кэширование увеличивает эффективность взаимодействия типа клиент/сервер, если в последующих запросах клиент запрашивает у сервера те же данные.

Хотя кэширование и увеличивает эффективность, оно не меняет сути взаимодействия типа клиент/сервер, в основе которого лежит допущение о том, что процесс обработки является системой, управляемой по запросу. Выше уже было сказано о том, что программа может работать автономно до тех пор, пока ей не потребуются данные. Для получения данных от сервера программа должна выполнить роль клиента. Подобное представление модели взаимодействия в виде системы, управляемой по запросу, вполне естественно и основано на накопленном опыте. Кэширование помогает снизить стоимость получения информации за счет уменьшения стоимости ее поиска для повторных запросов.

Как же можно понизить стоимость поиска информации для первого запроса? В распределенной системе можно запустить в фоновом режиме параллельные программы, которые собирают и распространяют информацию *до того*, как какая-либо программа запросит ее. Это позволяет существенно снизить стоимость

поиска информации даже для первого запроса. Более важно то, что предварительный сбор информации позволяет системе продолжать работу даже в том случае, когда другие машины или соединяющие их сети выйдут из строя.

Предварительный сбор информации лежит в основе работы команды `ruptime` системы 4BSD UNIX. Эта команда отображает степень загрузки центрального процессора каждой машины локальной сети и время, прошедшее с момента ее начальной загрузки. Для реализации такой возможности на каждом компьютере локальной сети запускается специальная фоновая программа, которая периодически рассыпает в широковещательном режиме информацию о нем, используя протокол UDP. Эта же программа собирает поступающую информацию и помещает ее в файл. Поскольку компьютеры локальной сети все время передают данные, каждый из компьютеров будет располагать самой свежей информацией о своих соседях, а клиенту, ищущему эту информацию, не требуется доступ к сети. Вместо этого он должен считать данные из файла и вывести их в удобочитаемой для человека форме.

Главным преимуществом предварительного сбора информации и сохранения ее на локальной машине является скорость доступа к ней. Команда `ruptime` отвечает на вызов немедленно, не ожидая, пока сообщения пройдут через сеть. Другое преимущество заключается в том, что клиент может получить информацию о машинах, которые уже не работают. В частности, если машина прекращает широковещательную передачу информации, клиент всегда может узнать, сколько времени прошло с момента последней передачи (то есть, какое количество времени машина работала автономно).

Предварительный сбор информации имеет один недостаток: если никто не интересуется собираемыми данными, время процессора и пропускная способность сети тратятся впустую. Например, широковещательные сообщения команды `ruptime` рассыпаются все время, даже ночью. Постоянно происходит и сбор данных, хотя ночью они мало кому могут понадобиться. Если к локальной сети подключено всего несколько машин, то стоимость предварительного сбора информации незначительна. В этом случае работа фоновой программы не оказывает какого-либо влияния на загрузку центрального процессора и сети. Однако для сетей с большим количеством узлов процесс предварительного сбора информации и передачи ее с помощью широковещательных сообщений, требует значительных ресурсов, что делает его слишком дорогим. В частности, стоимость чтения и обработки широковещательных сообщений становится высокими. Таким образом, предварительный сбор информации не относится к наиболее популярным альтернативам модели взаимодействия типа клиент/сервер.

## 21.8. Резюме

Распределенные программы в процессе работы взаимодействуют друг с другом по сети. Часто такое взаимодействие можно описать в виде модели клиент/сервер. Суть его состоит в том, что на специальном компьютере (сервере) запускается фоновый процесс, который ожидает поступления запроса и, получив его, выполняет указанное в нем действие. После выполнения действия клиенту обычно посыпается ответное сообщение. Клиентская программа формирует запрос, отправляет его на сервер и затем ожидает ответа.

В этой главе были рассмотрены примеры клиентов и серверов. Было показано, что некоторые клиенты отсылают запросы напрямую серверу, в то время как другие отсылают запросы в широковещательном режиме. Режим широковещания обычно применяется в локальной сети, когда клиентской машине неизвестен адрес сервера.

Также в этой главе отмечалось, что если на сервере используется один из протоколов семейства TCP/IP (например, UDP), то он может принимать и отвечать на запросы клиентов, поступающие через объединенную сеть. Если же взаимодействие клиента с сервером происходит на уровне сетевых фреймов и физических адресов сетевого оборудования, то зона взаимодействия ограничена отдельной физической сетью.

И наконец, мы рассмотрели альтернативную модель взаимодействия между клиентами и серверами, в которой для уменьшения задержек используется предварительный сбор информации. В качестве примера такого взаимодействия была рассмотрена служба сбора информации о состоянии компьютеров локальной сети *ruptime*, используемая в 4BSD UNIX.

## Материал для дальнейшего изучения

Служба эха протокола UDP описана Постелом (Postel) в [RFC 862]. Описание команды *ruptime* и связанной с ней команды *rwho* можно найти в справочнике программиста системы UNIX (*UNIX Programmer's Manual*).

В работе Файнлера (Feinler) и др. [52] подробно описано много стандартных серверных протоколов, не рассмотренных в этой главе, включая такие как аннулирование (discard), формирование символов (character generation), дата и время (day and time), активные пользователи (active users) и цитата дня (quote of the day). В следующих главах будут рассмотрены другие протоколы семейства TCP/IP.

## Упражнения

- 21.1. Создайте приложение — клиент эхо-сервера протокола UDP, которое должно посыпать дейтаграмму по известному адресу сервера и ожидать от него ответа. Получив ответ, приложение-клиент должно сравнить полученные данные с первоначальным сообщением.
- 21.2. Внимательно ознакомьтесь с процессом обработки IP-адресов в эхо-сервере протокола UDP. При каких условиях обмен IP-адресов отправителя и получателя в ответном сообщении сервера является некорректным?
- 21.3. Как вы уже знаете, серверы можно реализовать в виде самостоятельных прикладных программ или путем встраивания кода сервера в протокольное программное обеспечение операционной системы. Каковы преимущества и недостатки запуска пользовательских процессов на сервере?
- 21.4. Предположим, что IP-адрес локальной машины, на которой запущен эхо-сервер протокола UDP, неизвестен. Однако известно, что она отвечает на запросы, посланные по номеру порта номер 7. Существует ли такой IP-адрес, с помощью которого можно связаться с эхо-сервером?
- 21.5. Создайте приложение — клиент службы времени протокола UDP.
- 21.6. Опишите ситуации, при которых сервер и клиенты могут располагаться в разных физических сетях. Могут ли сервер и клиенты RARP располагаться в разных физических сетях? Обоснуйте свой ответ.
- 21.7. Какой основной недостаток метода периодической широковещательной передачи информации о состоянии машин локальной сети?

- 21.8.** Ознакомьтесь с форматом данных, которые передают в широковещательном режиме серверы поддержки команды `ruptime` системы 4BSD UNIX. Какую информацию, кроме состояния машины, может получить клиент?
- 21.9.** Какие серверы запущены на компьютерах вашего сетевого центра? Если у вас нет доступа к системным файлам конфигурации, в которых перечислены серверы, запущенные на компьютере, проверьте, есть ли в вашей системе команда, которая выводит список открытых портов протоколов TCP и UDP (например, команда UNIX `netstat`).
- 21.10.** Некоторые серверы позволяют администратору корректно завершить свою работу, а также перезапустить их. Каковы преимущества подобного способа завершения работы сервера?

# 22

## Интерфейс сокетов

### 22.1. Введение

До настоящего момента мы обсуждали принципы и понятия, лежащие в основе работы семейства протоколов TCP/IP, не углубляясь в описание интерфейса между приложениями и программами поддержки протокола. В этой главе рассматривается один из примеров *интерфейса прикладных программ* (*Application Program Interface*, или *API*), который обеспечивает взаимодействие между приложениями и программами протокола семейства TCP/IP. Существует две причины, по которым обсуждение интерфейсов API пока можно отложить. Во-первых, интерфейсы и протоколы TCP/IP необходимо рассматривать отдельно, так как в стандартах явно не определено, каким образом приложения должны взаимодействовать с программами поддержки протокола. Таким образом, структура интерфейса не является стандартизированной, и его конструкция не входит в область рассмотрения семейства протоколов. Во-вторых, на практике не стоит привязывать протоколы к какому-либо определенному интерфейсу API, поскольку ни одна из структур интерфейса не гарантирует оптимальную функциональность для всех систем. В частности, поскольку программы поддержки протокола встроены в операционную систему компьютера, детали интерфейса зависят от операционной системы.

Несмотря на ограниченность стандарта, рассмотрение одного из примеров интерфейса поможет понять, каким образом программисты используют семейство протоколов TCP/IP. Хотя выбранный нами пример взят из операционной системы BSD UNIX, он по сути превратился в стандарт, который широко распространен и используется во многих других системах. В частности, он формирует основу интерфейса *Windows Sockets*<sup>1</sup> компании Microsoft. Читатель должен иметь в виду, что мы намерены только привести конкретный пример, а не давать рекомендации, как нужно разрабатывать интерфейсы API. Читатель также должен не забывать, что перечисленные в этой главе операции не являются частью стандартов семейства протоколов TCP/IP.

### 22.2. Принцип ввода-вывода в системе UNIX

Система UNIX была создана в конце 1960-х — начале 1970-х годов и изначально предназначалась для работы в режиме разделения времени на однопроцессорных компьютерах. Она представляла собой операционную систему с планированием процессов, в которой каждая прикладная программа выполнялась на пользовательском уровне. Прикладная программа взаимодействовала с операционной

---

<sup>1</sup> Программисты часто используют термин *WINSOCK* вместо Windows Sockets.

системой путем *системных вызовов* (*system calls*). С точки зрения программиста системные вызовы выглядят и функционируют точно так же, как и вызовы обычных процедур. В качестве параметров им может передаваться несколько переменных, и возвращать они могут одно или несколько значений. В качестве параметров могут использоваться обычные значения (например, целые числа), либо указатели на объекты, находящиеся в прикладной программе (например, адрес входного буфера, куда помещаются введенные с клавиатуры символы).

Элементарные операции ввода-вывода системы UNIX были заимствованы из первых операционных систем, например из системы Multics и других. При их создании разработчики руководствовались принципом, который иногда называют *open-read-write-close* (*открыть-прочитать-записать-закрыть*). Прежде чем пользовательский процесс сможет выполнять операции ввода-вывода, он должен вызвать функцию `open`. В качестве параметров ей передается имя файла или устройства, которое предполагается использовать, а также атрибуты прав доступа, которые должны быть назначены при открытии этому файлу или устройству. Функция `open` возвращает небольшое целое число — *дескриптор файла*<sup>2</sup> (*file descriptor*), который используется процессом при выполнении операций ввода-вывода на открытом файле или устройстве. После открытия объекта (файла или устройства) пользовательский процесс может инициировать передачу данных, вызвав один или несколько раз функции `read` или `write`. С помощью функции `read` данные из файла или устройства передаются на обработку пользовательскому процессу. Функция `write` выполняет обратную операцию — передает данные из области памяти пользовательского процесса в файл или устройство. Функциям `read` и `write` передаются три параметра, определяющие дескриптор используемого файла, адрес буфера и количество предназначенных для передачи байтов. По завершении передачи данных пользовательский процесс должен вызвать функцию `close` и тем самым сообщить операционной системе, что использование указанного объекта завершено. Следует отметить, что операционная система автоматически закроет все открытые дескрипторы файлов, если процесс завершится без вызова функции `close`.

### 22.3. Возможности сетевого ввода-вывода в системе UNIX

Создатели системы UNIX изначально разработали операции ввода-вывода с учетом описанного выше принципа *open-read-write-close*. Поэтому в их проект были включены операции ввода-вывода для символьных устройств, таких как клавиатура, и блочных устройств, таких как диски и файлы данных. В одной из первых реализаций семейства протоколов TCP/IP для системы UNIX также использовался принцип *open-read-write-close*; операции ввода-вывода выполнялись с особым файлом `/dev/tcp`.

Группа разработчиков, включившая сетевые протоколы в систему BSD UNIX, решила, что, поскольку сетевые протоколы сложнее стандартных устройств ввода-вывода, взаимодействие между пользовательскими процессами и сетевыми протоколами также должно быть сложнее, чем взаимодействие между пользовательскими процессами и стандартными средствами ввода-вывода. В частности, с помощью API-интерфейса протокола программисты должны иметь возможность создавать как программу-сервер, пассивно ожидающую подключений, так и программу-клиент, которая открывает соединения в активном режиме. Более того, желательно, чтобы при отправке каждой дейтаграммы прикладная программа

---

<sup>2</sup> Возникновение термина “дескриптор файла” связано с тем, что все устройства в системе UNIX отражены в пространство имен файловой системы. В большинстве случаев нет различий между операциями ввода-вывода, выполняемыми с файлами или устройствами.

могла явно указать адрес ее получателя, а не задавать его один раз при вызове функции `open`. Поэтому, чтобы учесть все приведенные выше соображения, разработчики решили отказаться от традиционного принципа *open-read-write-close* и добавить в операционную систему несколько новых вызовов, а также новые библиотечные подпрограммы. Таким образом, добавление сетевых протоколов в систему UNIX основательно усложнило интерфейс ввода-вывода.

Дополнительные трудности при разработке структуры API-интерфейса сетевых протоколов системы UNIX возникают из-за стремления разработчиков создать общий механизм, который бы позволял одновременно поддерживать множество протоколов. Такая обобщенность позволяет, например, наряду с семейством протоколов TCP/IP включить в операционную систему программы поддержки других типов сетевых протоколов. При этом любая прикладная программа может одновременно использовать несколько типов протоколов. В результате прикладная программа не может просто так передать операционной системе 32-битовый адрес и ожидать от операционной системы правильной его интерпретации. Прикладная программа должна явно указать, что это 32-битовое число представляет IP-адрес.

## 22.4. Концепция сокетов

В основу API-интерфейса сетевых операций ввода-вывода системы UNIX положена концепция *сокетов (sockets)*<sup>3</sup>. Под сокетом будем подразумевать обобщенную форму механизма доступа к файлам системы UNIX, обеспечивающего конечную точку для взаимодействия. Как и при получении доступа к файлам, прикладные программы при необходимости посыпают запрос операционной системе на создание сокета. Система возвращает небольшое целое число, которое затем используется в прикладной программе для идентификации только что созданного сокета. Основное отличие между дескриптором файла и дескриптором сокета заключается в том, что операционная система связывает дескриптор файла с определенным файлом или устройством при вызове прикладной программой функции `open`. Однако при создании сокета прикладная программа может и не привязывать его к определенному адресу получателя. Адрес получателя может указываться в прикладной программе каждый раз при использовании сокета (например, при отсылке дейтаграммы). Кроме того, можно связать с сокетом и адрес определенного получателя. Тогда при использовании сокета не нужно будет каждый раз указывать адрес получателя (как, например, при открытии TCP-соединения).

В определенном смысле можно сказать, что сокеты функционируют аналогично файлам или устройствам системы UNIX. Поэтому их можно использовать совместно с традиционными функциями, такими как `read` или `write`. Например, после того, как прикладная программа создаст сокет и свяжет его с определенным TCP-соединением, ведущим к получателю, который находится во внешней сети, она может использовать функцию `write` для отсылки по этому соединению потока данных. Прикладная программа, выполняющаяся на другом конце соединения, может использовать функцию `read` для получения этих данных. Чтобы обеспечить возможность использования элементарных функций, таких как `read` или `write` как с файлами, так и с сокетами, операционная система назначает в качестве дескрипторов сокета и файла целые числа, выбранные из одного набора. Причем, если данное число назначено в качестве дескриптора файла, то оно не может быть одновременно назначено дескриптором сокета, и наоборот.

---

<sup>3</sup> Пока мы будем считать, что сокеты является частью операционной системы, и опишем концепцию их использования так, как они реализованы в UNIX. В последующих разделах будет показано, как в других операционных системах с помощью библиотечных процедур реализуется API-интерфейс сокетов.

## 22.5. Создание сокета

Сокеты создаются с помощью функции `socket` как только в них возникает потребность. Этой функции передается три целочисленных параметра, а возвращает она целое число:

```
результат = socket (семейство, тип, протокол)
```

Параметр *семейство* определяет семейство протоколов, которое будет использоваться с сокетом. Другими словами, этот параметр указывает операционной системе, как следует интерпретировать указанные в прикладной программе адреса. В настоящее время существует несколько семейств протоколов. Среди прочих можно выделить следующие<sup>4</sup>:

- TCP/IP (PF\_INET);
- PUP (PF\_PUP), разработанный корпорацией Xerox;
- AppleTalk (PF\_APPLETALK), созданный фирмой Apple Computer, Inc.
- файловая система UNIX (PF\_UNIX).

Параметр *тип* задает тип желаемого взаимодействия. Здесь можно указать:

- надежную потоковую службу доставки (SOCK\_STREAM);
- службу доставки дейтаграмм, не требующую установки соединения с получателем (SOCK\_DGRAM);
- низкоуровневый тип (SOCK\_RAW), с помощью которого привилегированные программы могут получить доступ к протоколам низкого уровня и драйверам сетевых интерфейсов.

Кроме перечисленных выше, планировалось создать еще два типа, но эти планы так и не были осуществлены.

Хотя обобщенный подход, при котором семейство протоколов и тип желаемого взаимодействия рассматриваются независимо друг от друга, может показаться достаточным для того, чтобы учесть все возможные случаи, на самом деле это не так. Во-первых, бывает, что семейством протоколов не поддерживается один или несколько возможных типов служб. Например, в системе UNIX предусмотрен механизм межпроцессного взаимодействия, который называется *каналом (pipe)*. Несмотря на то что в нем используется надежная потоковая служба доставки, однако в нем не предусмотрен механизм для упорядоченной доставки пакетов. Следовательно, не все комбинации семейства протоколов и типа службы имеют смысл. Во-вторых, в некоторые семейства могут быть включены несколько протоколов, поддерживающих один тип службы. Например, может случиться так, что одно семейство протоколов поддерживает две службы доставки дейтаграмм, не требующих установки соединения с получателем. Поэтому, чтобы можно было работать с несколькими протоколами, входящими в одно семейство, в функции `socket` был предусмотрен третий параметр, который используется для выбора определенного протокола. Чтобы указать значение третьего параметра, программист должен в достаточной степени разбираться в структуре семейства протоколов и располагать информацией о типе службы, предоставляемой каждым из протоколов.

Поскольку разработчики попытались охватить большинство стандартных операций системы UNIX с помощью одного механизма сокетов, возникла необходимость

---

<sup>4</sup> В прикладных программах, написанных для системы UNIX, для обозначения семейства протоколов используются символические имена, например PF\_INET. Соответствующие им числовые значения определены в системных заголовочных файлах.

в методе, с помощью которого можно было бы смоделировать механизм каналов системы UNIX. Нам нет необходимости углубляться во все детали работы этого механизма, интерес представляет лишь одно его ярко выраженное свойство: механизм каналов отличается от стандартных сетевых операций, поскольку вызывающий процесс одновременно создает обе конечные точки взаимодействия. Для поддержки в системе сокетов механизма каналов разработчики добавили функцию `socketpair`, которая имеет следующий синтаксис:

```
socketpair(семейство, тип, протокол, массив-сокетов)
```

По сравнению с функцией `socket`, функция `socketpair` имеет дополнительный параметр, называемый `массив-сокетов`. В нем указывается адрес двухэлементного массива целых чисел. Функция `socketpair` одновременно создает два сокета и помещает их дескрипторы в два элемента массива, адрес которого указан в качестве параметра `массив-сокетов`. Читатели должны понимать, что нет смысла использовать функцию `socketpair` для семейства протоколов TCP/IP. Мы упоминаем ее только для того, чтобы сделать наше описание интерфейса более полным.

## 22.6. Наследование сокетов и завершение процессов

Для запуска новых прикладных программ в системе UNIX используются функции `fork` и `exec`. Этот процесс состоит из двух этапов. На первом этапе функция `fork` создает отдельную копию выполняющейся в текущий момент прикладной программы. На втором этапе новая копия заменяется вновь запускаемой прикладной программой. Когда программа вызывает функцию `fork`, только что созданная копия унаследует права доступа ко всем открытым сокетам (точно так же, как и ко всем открытым файлам). Когда программа вызывает функцию `exec`, для нового приложения сохраняются права доступа ко всем открытым сокетам. Ниже будет показано, что главная программа сервера использует свойство наследования сокетов при создании подчиненных программ, обрабатывающих запросы, которые поступают через заданное соединение. Что касается внутренней структуры, операционная система ведет счетчик числа обращений к каждому сокету, поэтому ей всегда точно известно, сколько прикладных программ (процессов) имеют к нему доступ.

И старый, и новый процессы имеют одинаковые права доступа к существующим сокетам, и оба могут получить к ним доступ. Таким образом, ответственность за обеспечение корректного использования общего сокета двумя процессами несет программист.

По завершении использования сокета процесс вызывает функцию `close`, которая имеет следующий синтаксис:

```
close(сокет)
```

Здесь параметр `сокет` задает дескриптор сокета, который необходимо закрыть. Если процесс по какой-либо причине прекращается, система закрывает все сокеты, которые остались открытыми. Что касается внутренней структуры, вызов функции `close` уменьшает на единицу значение счетчика числа обращений к конкретному сокету. Если значение счетчика становится равным нулю, сокет аннулируется.

## 22.7. Определение локального адреса

При создании сокета он не связывается ни с адресом локальной машины, ни с удаленным адресом получателя. При использовании семейства протоколов TCP/IP это означает, что сокету не присваивается ни номер локального порта

протокола, ни адрес порта получателя, ни его IP-адрес. В большинстве случаев прикладным программам не нужна информация о локальном адресе машины, на который они работают, поскольку выбор одного из локальных адресов осуществляется программами поддержки протокола. Однако серверные процессы, которые должны быть привязаны к одному из стандартных портов, должны уметь указывать этот порт системе. Поэтому после создания сокета сервер с помощью функции `bind` назначает ему локальный адрес. Функция `bind` имеет следующий синтаксис:

```
bind(сокет, локальный-адрес, длина-адреса)
```

Параметр `сокет` представляет собой целочисленный дескриптор сокета, для которого выполняется привязка. Параметр `локальный-адрес` является адресом структуры `sockaddr`, определяющей локальный адрес, к которому нужно привязать сокет. Параметр `длина-адреса` представляет собой целое число, которое определяет длину адреса в байтах. Вместо того чтобы просто задать адрес в виде последовательности байтов, разработчики решили использовать для представления адресов структуру под общим названием `sockaddr`, которая изображена на рис. 22.1.



Рис. 22.1. Структура `sockaddr`, которая используется при передаче протокольного адреса интерфейсу сокета

Структура `sockaddr` начинается с 16-битового поля, определяющего семейство протоколов, к которому принадлежит адрес. После него следует сам адрес длиной до 14 октетов. При объявлении в языке C, структура `sockaddr` должна быть объединением (union) структур, определяющих все возможные семейства адресов.

Значение в поле *идентификатора семейства адресов* определяет формат остальных полей структуры. Например, если значение идентификатора равно 2, то это означает, что в остальных октетах адреса задан адрес протокола TCP/IP<sup>5</sup>. Способ интерпретации октетов в поле адреса структуры `sockaddr` зависит от семейства протоколов. Для передачи интерфейсу сокета адресов протокола TCP/IP используется структура, которая называется `sockaddr_in`. В ней указывается IP-адрес машины и ее номер порта протокола (рис. 22.2).

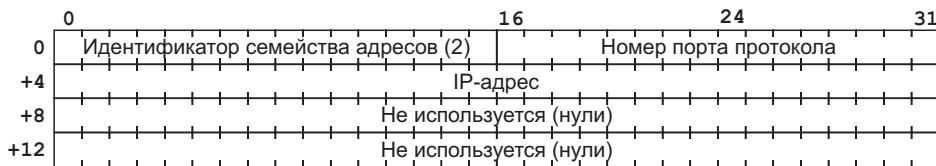


Рис. 22.2. Формат адресной структуры сокета (`sockaddr_in`) при использовании протокола TCP/IP. В структуре указывается как IP-адрес, так и расположенный по этому адресу номер порта протокола

<sup>5</sup> В системе UNIX для обозначения адресов протокола TCP/IP используется символическое имя `PF_INET`.

Хотя при вызове функции `bind` в адресной структуре можно указать произвольные значения, не все возможные привязки являются корректными. Например, вызывающая программа может запросить номер локального порта протокола, который уже используется другой программой, или указать недействительный IP-адрес. В таких случаях функция `bind` завершает свою работу аварийно и возвращает соответствующий код ошибки.

## 22.8. Связывание сокетов с адресами получателей

Сразу после создания сокет находится в *неподключенном состоянии* (*unconnected state*). Это означает, что сокет не связан с каким-либо адресом внешнего получателя. Для привязки сокета к некоторому постоянному получателю и перевода его в *подключенное состояние* (*connected state*) используется функция `connect`. Таким образом, прежде чем начать передачу данных через надежный потоковый сокет, прикладная программа должна установить соединение с получателем, вызвав функцию `connect`. Если сокет используется со службами доставки дейтаграмм, не требующих установки соединения с получателем, то функцию `connect` можно и не вызывать. Однако тогда каждый раз при передаче данных придется указывать адрес получателя. Функция `connect` имеет следующий синтаксис:

```
connect(сокет, адрес-получателя, длина-адреса)
```

Параметр `сокет` представляет собой целочисленный дескриптор сокета, который переводится в подключенное состояние. Параметр `адрес-получателя` представляет собой адрес структуры сокета, определяющего адрес получателя, к которому необходимо подключить сокет. Параметр `длина-адреса` определяет длину в байтах адреса получателя.

Принцип работы функции `connect` зависит от используемых сетевых протоколов. Выбор надежной потоковой службы доставки дейтаграмм из семейства `PF_INET` предполагает использование протокола TCP. В таких случаях функция `connect` устанавливает TCP-соединение с получателем и возвращает ошибку, если соединение установить невозможно. В случае использования службы, не требующей установки соединения с получателем, функция `connect` всего лишь сохраняет адрес получателя локально.

## 22.9. Пересылка данных через сокет

После создания и настройки параметров сокета, прикладная программа может использовать его для передачи данных. Для этой цели существует пять возможных функций: `send`, `sendto`, `sendmsg`, `write` и `writev`. Функции `send`, `write` и `writev` можно использовать только с подключенными сокетами, поскольку они не позволяют вызывающей программе указать адрес получателя. Отличия между этими тремя видами функций незначительны. Функция `write` имеет три параметра:

```
write(сокет, адрес-буфера, длина)
```

Параметр `сокет` содержит целочисленный дескриптор сокета (функцию `write` также можно использовать и с другими типами дескрипторов). Второй параметр (`адрес-буфера`) задает адрес предназначенных для отсылки данных, а третий параметр (`длина`) определяет количество пересылаемых байтов. Вызов функции `write` приостанавливает (блокирует) выполнение текущей программы до тех пор, пока данные не будут полностью переданы (точнее блокировка выполнения

программы функцией `write` происходит только тогда, когда внутренние системные буферы сокета уже заполнены). Подобно большинству системных вызовов функция `write` возвращает вызвавшей ее программе код ошибки, что позволяет программисту узнать о результате выполнения операции.

Системная функция `writev` работает наподобие функции `write`. Но, в отличие от последней, она позволяет прикладной программе передать данные, находящиеся в разных участках памяти без предварительного их копирования в непрерывный участок памяти. Функция `writev` имеет следующий синтаксис:

```
writev(сокет, массив-указателей, длина-массива)
```

Второй аргумент (*массив-указателей*) задает адрес массива типа `iovec`. В нем находится список адресов блоков памяти, из которых должно быть сформировано сообщение. Как показано на рис. 22.3, после каждого адреса блока указывается его длина в байтах. Третий аргумент (*длина-массива*) определяет число записей в массиве `iovec`.

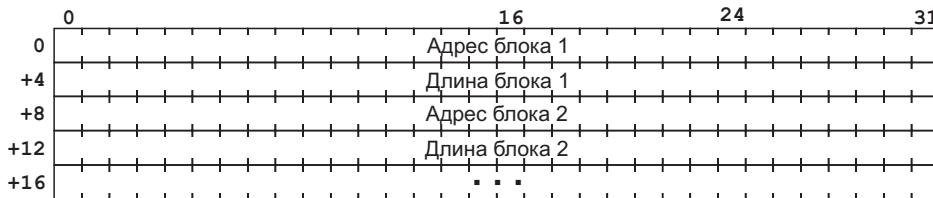


Рис. 22.3. Формат массива указателей типа `iovec`, который используется в функции `writev` и `readv`.

Функция `send` имеет следующий синтаксис:

```
send(сокет, адрес-сообщения, длина-сообщения, флагги)
```

Здесь параметр *сокет* определяет дескриптор сокета, через который будут передаваться данные. Второй параметр (*адрес-сообщения*) определяет адрес предназначенных для отсылки данных, а третий параметр (*длина-сообщения*) — количество байтов для пересылки. Параметр *флагги* предназначен для управления процессом передачи. В частности, с помощью специального значения, заданного в виде *флажка*, отправитель может указать, что сообщение должно быть послано как экстренное через сокет, который поддерживает такую возможность. Вспомните главу 13, “Надежная потоковая транспортная служба (TCP)”, в которой говорилось, что экстренные сообщения соответствуют понятию срочных данных в протоколе TCP. Еще одно значение *флажка* позволяет вызывающей программе послать запрос на пересылку сообщения без использования локальных таблиц маршрутизации. Если процессом маршрутизации управляет вызывающая программа, можно создавать программы для отладки сетевых приложений. Конечно, сокеты поддерживают не все запросы, поступающие от программ. Для некоторых запросов необходимо, чтобы программа располагала особыми привилегиями, а другие просто не поддерживаются во всех сокетах.

С помощью функций `sendto` и `sendmsg` вызывающая программа может послать сообщение через неподключенный сокет. Для работы этих функций вызывающая программа должна определить адрес получателя. Функция `sendto`, которой в качестве параметра передается адрес получателя, имеет следующий синтаксис:

```
sendto(сокет, адрес-сообщения, длина-сообщения, флагги, адрес-получателя, длина-адреса)
```

Первые четыре параметра этой функции аналогичны параметрам, используемым в функции `send`. Последние два параметра определяют адрес получателя и задают длину этого адреса. Для указания адреса получателя используется структура `sockaddr_in`, изображенная на рис. 22.2.

Программист может прибегнуть к функции `sendmsg` в тех случаях, когда использование длинного списка параметров, необходимых для выполнения функции `sendto`, делает программу неэффективной или трудной для чтения. Функция `sendmsg` имеет следующий синтаксис:

```
sendmsg(сокет, адрес-структуры, флагги)
```

Здесь параметр `адрес-структурь` представляет собой адрес структуры, формат которой изображен на рис. 22.4. Структура содержит информацию о пересылаемом сообщении, его длину, адрес получателя, а также длину адреса. Использование этой функции особенно эффективно, поскольку существует соответствующая операция ввода (описанная ниже), с помощью которой создается структура сообщения точно в таком же формате.

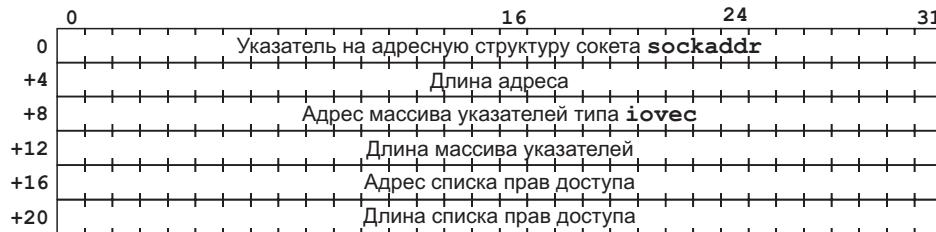


Рис. 22.4. Формат структуры сообщения, используемый в функции `sendmsg`

## 22.10. Получение данных через сокет

Наряду с пятью операциями вывода в API-интерфейсе сокетов предусмотрено пять функций, которые могут использоваться процессом для получения данных через сокет: `read`, `readv`, `recv`, `recvfrom` и `recvmsg`. Стандартную функцию ввода, `read`, можно использовать только тогда, когда сокет подключен. Она имеет следующий синтаксис:

```
read(дескриптор, адрес-буфера, длина-буфера)
```

Здесь параметр `дескриптор` задает целочисленный дескриптор сокета или дескриптор файла, из которого необходимо считать данные. Параметр `адрес-буфера` определяет участок памяти, куда будут записываться считываемые данные, а параметр `длина-буфера` — максимальное количество считываемых байтов (или длину буфера).

Альтернативная форма функции под названием `readv` позволяет вызывающей программе считать данные в разные участки памяти. Функция `readv` имеет следующий синтаксис:

```
readv(дескриптор, массив-указателей, длина-массива)
```

Параметр `массив-указателей` задает адрес массива типа `iovec` (см. рис. 22.3), где находится список адресов блоков памяти, в которые должны быть записаны считанные данные. Параметр `длина-массива` определяет число записей в массиве типа `iovec`.

Помимо стандартных операций ввода, имеются еще три функции для ввода сетевых сообщений. Для получения данных от подключенного сокета процесс может вызывать функцию `recv`. Эта функция имеет следующий синтаксис:

```
recv(сокет, адрес-буфера, длина-буфера, флагги)
```

Параметр `сокет` определяет дескриптор сокета, из которого необходимо считать данные. Параметр `адрес-буфера` определяет адрес участка в памяти, куда нужно поместить входящее сообщение, а параметр `длина-буфера` определяет длину буферной области. И наконец, параметр `флагги` позволяет вызывающей программе управлять процессом приема данных. В числе возможных значений, используемых в качестве `флажков`, предусмотрено значение, позволяющее вызывающей программе узнать, есть ли еще данные в буфере. При этом ей передается копия следующего входящего сообщения, но само сообщение не удаляется из сокета.

С помощью функции `recvfrom` вызывающая программа может считать данные из отключенного сокета. Для указания места в памяти, куда нужно будет поместить адрес отправителя, в этой функции предусмотрены дополнительные параметры. Функция `recvfrom` имеет следующий синтаксис:

```
recvfrom(сокет, адрес-буфера, длина-буфера, флагги, адрес-отправителя, длина-адреса)
```

Два дополнительных параметра, `адрес-отправителя` и `длина-адреса`, представляют собой указатель на адресную структуру сокета, куда помещается адрес отправителя, и целое число, определяющее длину адреса. Получив сообщение, операционная система помещает адрес его отправителя в область памяти, определяемой параметром `адрес-отправителя`. Длина адреса отправителя указывается в качестве параметра `длина-адреса`. Обратите внимание, что в описанной выше функции вывода `sendto` формат адреса получателя совпадает с форматом адреса отправителя, генерируемым функцией `recvfrom`. Это сделано для удобства отправки ответных сообщений.

Последняя используемая для ввода функция под названием `recvmsg` аналогична функции вывода `sendmsg`. Функция `recvmsg` функционирует подобно функции `recvfrom`, но для ее выполнения необходимо меньшее количество параметров. Она имеет следующий синтаксис:

```
recvmsg(сокет, адрес-структурь, флагги)
```

Здесь параметр `адрес-структурь` задает адрес структуры, где хранится адрес буфера для помещения входящего сообщения, а также место для адреса отправителя. Структура, создаваемая функцией `recvmsg`, совпадает со структурой, используемой в функции `sendmsg`, что позволяет им успешно функционировать в паре.

## 22.11. Получение локального и удаленного адресов сокета

Уже было сказано, что вновь созданные процессы наследуют набор открытых сокетов от процесса, который их создал. Иногда только что созданному процессу нужно определить адрес получателя, к которому подключен сокет, а также локальный адрес сокета. Подобную информацию можно получить с помощью таких функций, как `getpeername` и `getsockname`. (Несмотря на их названия, обе функции имеют дело с тем, что мы понимаем под словом “адрес”.)

Чтобы определить адрес удаленного получателя (`peer`), к которому подключен сокет, программа должна вызвать функцию `getpeername`. Она имеет следующий синтаксис:

```
getpeername(сокет, адрес-получателя, длина-адреса)
```

Первый параметр (*сокет*) определяет дескриптор сокета, для которого нужно узнать адрес удаленного получателя. Второй аргумент (*адрес-получателя*) представляет собой указатель на адресную структуру сокета *sockaddr* (см. рис. 22.1), в которую будет помещен адрес удаленного получателя. И наконец, параметр *длина-адреса* представляет собой указатель на переменную целого типа, в которую помещается длина адреса. Функцию *getpeername* можно использовать только с подключенными сокетами.

Функция *getsockname* возвращает связанный с определенным сокетом локальный адрес. Она имеет следующий синтаксис:

```
getsockname(сокет, локальный-адрес, длина-адреса)
```

Как и ожидалось, параметр *сокет* определяет дескриптор сокета, для которого необходимо получить локальный адрес. Параметр *локальный-адрес* является указателем на структуру типа *sockaddr*, которая будет содержать адрес, а параметр *длина-адреса* — указателем на переменную целого типа, в которую помещается длина адреса.

## 22.12. Получение и установка параметров сокета

Помимо привязки сокета к локальному адресу или его подключения к адресу получателя, необходим механизм, который бы позволял прикладным программам управлять самим сокетом. Например, при использовании протоколов, в которых для обеспечения надежности передачи данных применяется метод повторной передачи по истечении тайм-аута, прикладной программе иногда необходимо определить либо установить значение тайм-аута. Кроме того, иногда требуется изменить количество областей буферной памяти, определить, может ли сокет передавать сообщения в широковещательном режиме, а также управлять обработкой срочных данных. Чтобы не добавлять новые функции для каждой новой операции управления, разработчики решили создать единый механизм, в который включили две функции: *getsockopt* и *setsockopt*.

С помощью функции *getsockopt* прикладная программа может запрашивать информацию о сокете. Вызывающая программа определяет сокет, параметры которого ее интересуют, а также место в буфере, где необходимо хранить затребованную информацию. Операционная система анализирует свои внутренние структуры данных, относящиеся к сокету, и передает затребованную информацию вызывающей программе. Функция *getsockopt* имеет следующий синтаксис:

```
getsockopt(сокет, уровень, ид-параметра, значение-параметра, длина)
```

Параметр *сокет* определяет дескриптор сокета, о котором необходимо получить информацию. Параметр *уровень* определяет, применяется ли операция к самому сокету или к используемому в данный момент базовому протоколу. Параметр *ид-параметра* определяет один параметр, к которому относиться запрос. Пара параметров *значение-параметра* и *длина* являются указателями. Первый из них задает адрес буфера, в который система помещает затребованное значение, а второй — адрес целочисленной переменной, куда система помещает длину значения параметра.

Функция *setsockopt* позволяет прикладной программе установить указанный параметр сокета, используя для этого набор значений, полученных с помощью функции *getsockopt*. Вызывающая программа указывает сокет, для которого необходимо изменить параметр, идентификатор параметра, а также его значение. Функция *setsockopt* имеет следующий синтаксис:

```
setsockopt(сокет, уровень, ид-параметра, значение-параметра, длина)
```

Ее параметры сходные параметрам функции `getsockopt`, за исключением аргумента `длина`, в котором указывается длина передаваемого системе значения параметра. Вызывающая программа должна указать для параметра, корректное значение, а также правильную длину этого значения. Конечно, не все параметры применимы ко всем сокетам. Корректность и формат отдельных запросов зависит от текущего состояния сокета, а также от используемых в данный момент базовых протоколов.

## 22.13. Определение длины очереди для сервера

Один из относящихся к сокетам параметров используется так часто, что ему посвящена отдельная функция. Чтобы понять происхождение этой функции, рассмотрим, что представляет собой программа-сервер. При запуске она создает сокет, привязывает его к одному из стандартных портов протокола и ожидает поступления запроса. Если сервер использует надежную потоковую службу доставки сообщений или на формирование ответного сообщения уходит много времени, новый запрос может поступить до того, как сервер закончит обработку ранее поступившего запроса. Чтобы программа поддержки базового протокола не отвергала и не аннулировала входящие запросы, сервер должен сообщить ей, что все входящие запросы должны помещаться в очередь и ожидать обработки.

Функция `listen` позволяет серверу подготовить сокет для приема входящих соединений. С точки зрения базовых протоколов функция `listen` переводит сокет в состояние готовности к приему соединений в пассивном режиме. Вызывая функцию `listen`, сервер также сообщает операционной системе о том, что программы поддержки протокола должны поставить в очередь несколько одновременно поступивших в сокет запросов. Эта функция имеет следующий синтаксис:

```
listen(сокет, длина-очереди)
```

Параметр `сокет` задает дескриптор сокета, который необходимо подготовить для использования сервером, а параметр `длина-очереди` определяет длину очереди запросов для этого сокета. После вызова этой функции система сможет поставить в очередь на обработку указанное во втором параметре (`длина-очереди`) количество одновременно поступивших запросов к данному сокету. Если при поступлении очередного запроса окажется, что очередь уже заполнена, операционная система отвергнет текущее соединение и аннулирует запрос. Функция `listen` применяется только к тем сокетам, в которых используется надежная потоковая служба доставки сообщений.

## 22.14. Процесс принятия соединений сервером

Как уже было сказано, серверная программа использует функции `socket`, `bind` и `listen` для того, чтобы создать сокет, привязать его к стандартному порту протокола и определить длину очереди для входящих запросов на соединение. Обратите внимание, что вызов функции `bind` связывает сокет со стандартным портом протокола, но сам сокет не подключается к какому-либо внешнему получателю. Однако при вызове этой функции внешний получатель должен быть определен в виде шаблона (*wildcard*), что позволит сокету получать запросы на соединение от любого клиента.

После создания сокета сервер должен перейти в состояние ожидания установки соединения. Для этого используется функция `accept`. После ее вызова серверное приложение переводится в состояние ожидания до тех пор, пока не поступит запрос на соединение. Эта функция имеет следующий синтаксис:

```
новый-сокет = accept(сокет, адрес-отправителя, длина-адреса)
```

Параметр `сокет` определяет дескриптор сокета, для которого необходимо ожидать поступления запроса. Параметр `адрес-отправителя` является указателем на структуру типа `sockaddr`, а параметр `длина-адреса` — указателем на переменную целого типа. При поступлении запроса система заполняет указанную структуру типа `sockaddr` адресом клиента, приславшего запрос, и помещает в переменную `длина-адреса` длину этого адреса. Затем система создает новый сокет, подключает его к клиенту, приславшему запрос, и возвращает дескриптор нового сокета вызывающей программе. При этом в исходном сокете сохраняется шаблон внешнего получателя, и он остается открытым. Таким образом, главная серверная программа может по-прежнему принимать запросы через исходный сокет.

Когда поступает запрос на соединение, функция `accept` возвращает управление вызвавшей ее программе. Сервер может обрабатывать запросы как последовательно, так и параллельно. При последовательной обработке запросов серверная программа обрабатывает запрос, закрывает новый сокет и вызывает функцию `accept` для получения следующего запроса на соединение. При параллельной обработке запроса, после возврата из функции `accept` главная серверная программа создает подчиненную программу для обработки запроса (если применить терминологию системы UNIX, то сервер порождает процесс для обработки запроса). Подчиненный процесс наследует копию нового сокета, поэтому может сразу перейти к обслуживанию запроса. Закончив обработку текущего запроса, подчиненный процесс закрывает сокет и завершает работу. После запуска подчиненного сервера исходный (главный) серверный процесс закрывает свою копию нового сокета. Затем он вызывает функцию `accept` для получения следующего запроса на соединение.

Принцип параллельной обработки запросов серверными программами может показаться запутанным, поскольку при этом несколько процессов используют один и тот же номер локального порта протокола. Ключ к пониманию этого механизма кроется в методе, посредством которого базовые протоколы работают с портами протоколов. Вспомним, что в протоколе TCP соединение определяется парой конечных точек (см. главу 13, “Надежная потоковая транспортная служба (TCP)”). Таким образом, не имеет значения, сколько процессов использует данный номер локального порта протокола, если они подключены к различным получателям. В случае параллельной обработки запросов сервером для каждого клиента создается по одному процессу, кроме того, существует еще один дополнительный процесс, принимающий соединения. В сокете, используемом процессом главного сервера, задан шаблон для внешнего получателя, что позволяет ему устанавливать соединение с любым внешним получателем. Каждый оставшийся процесс подключен только к определенному внешнему клиенту. Когда поступает TCP-сегмент, он пересыпается тому сокету, который подключен к машине отправителя этого сегмента. Если такого сокета не существует, сегмент пересыпается сокету, шаблон которого позволяет установить связь с этим внешним получателем. Более того, поскольку данный сокет еще не имеет открытого соединения с внешним получателем, он будет обрабатывать только TCP-сегменты, запрашивающие создание нового соединения.

## 22.15. Серверы, предоставляющие несколько служб

API-интерфейс сокетов обеспечивает еще одну интересную возможность построения сервера: один процесс может ожидать поступления соединения через несколько сокетов. Для этого предусмотрена функция `select`, которая входит

в общий набор операций ввода-вывода и может также применяться к сокетам<sup>6</sup>. Функция `select` имеет следующий синтаксис:

```
число-дескрипторов = select(число-дескрипторов, входная-маска,
                                выходная-маска, маска-исключений, тайм-аут)
```

В двух словах можно сказать, что функция `select` переводит вызывающий ее процесс в состояние ожидания до тех пор, пока один из набора дескрипторов, заданных в виде параметров, не перейдет в состояние готовности. Параметр `число-дескрипторов` определяет количество дескрипторов, которые нужно проверить (это число находится в пределах от 2 до `число-дескрипторов` минус 1). Параметр `входная-маска` — это указатель на битовую маску, определяющую дескрипторы, которые необходимо проверить на предмет готовности ко вводу данных. Параметр `выходная-маска` является указателем на битовую маску, определяющую дескрипторы, которые необходимо проверить на предмет готовности к выводу данных. Параметр `маска-исключений` — указатель на битовую маску, определяющую дескрипторы, которые необходимо проверить на предмет возникновения исключительных ситуаций. И наконец, если значение параметра `тайм-аут` не равняется нулю, то оно определяет адрес переменной целого типа, в которой хранится значение тайм-аута, по прошествии которого управление возвращается в вызывающую программу. Нулевое значение параметра `тайм-аут` переводит вызывающую программу в состояние ожидания, пока один из дескрипторов не перейдет в состояние готовности. Поскольку параметр `тайм-аут` содержит адрес переменной целого типа, в которой хранится значение тайм-аута, а не сам тайм-аут, процесс может запросить нулевую задержку, передав функции `select` адрес целого числа, содержащего ноль. Нулевое значение тайм-аута позволяет вызывающей программе опросить указанные дескрипторы на предмет их готовности к выполнению операций ввода-вывода.

Функция `select` возвращает вызывающей программе количество дескрипторов из указанного набора, готовых к выполнению операций ввода-вывода. Она также изменяет битовые маски, определенные параметрами `входная-маска`, `выходная-маска` и `маска-исключений`, чтобы сообщить прикладной программе, какие из выбранных дескрипторов находятся в состоянии готовности. Таким образом, прежде чем вызвать функцию `select`, прикладная программа должна установить те биты, которые соответствуют предназначенным для проверки дескрипторам. После вызова функции те из битов, которые по-прежнему установлены в единицу, соответствуют готовому к работе дескриптору.

Чтобы осуществить взаимодействие через несколько сокетов одновременно, процесс должен сначала создать все необходимые ему сокеты, а затем с помощью функции `select` определить, какой из сокетов первым перейдет в состояние готовности к выполнению операций ввода-вывода. Обнаружив готовый к работе сокет, для выполнения через него операций ввода-вывода процесс использует описанные выше функции.

## 22.16. Получение и назначение имен узлов сети

В большинстве операционных систем используется понятие внутреннего имени узла сети. Для подключенных к объединенной сети машин внутреннее имя узла обычно представляет собой доменное имя основного сетевого интерфейса машины. Функция `gethostname` дает возможность пользовательским процессам определить имя узла сети, а функция `sethostname` позволяет

---

<sup>6</sup> Версия функции `select` для интерфейса сокетов системы Windows может работать только с дескрипторами сокета.

привилегированным процессам назначить имя узлу сети. Функция `gethostname` имеет следующий синтаксис:

```
gethostname(адрес-буфера, длина)
```

Параметр `адрес-буфера` задает адрес массива байтов, в который система помещает имя узла, а параметр `длина` представляет собой целое число, определяющее длину этого массива. Чтобы назначить имя узлу сети, привилегированный процесс вызывает функцию `sethostname`, синтаксис которой имеет следующий вид:

```
sethostname(адрес-имени, длина)
```

Параметр `адрес-имени` задает адрес массива, в котором хранится имя, а параметр `длина` представляет собой целое число, определяющее длину этого массива.

## 22.17. Получение и назначение имени внутреннего домена узла сети

В сетевой операционной системе существует внутренняя строковая переменная, содержащая имя домена, к которому принадлежит компьютер. Имя домена назначается сетевому центру административным органом объединенной сети. После этого в сетевом центре формируется строка символов, идентифицирующая принадлежащую ему часть пространства доменных имен. Эта строка назначается в качестве имени домена всем компьютерам, принадлежащим этому центру. Например, машины в домене `cs.purdue.edu` имеют имена, взятые из легенды о короле Артуре: `merlin`, `arthur`, `guenevere` и `lancelot`. Самому домену присвоено имя `camelot`, поэтому операционной системе, работающей на машинах этой группы, необходимо сообщить, что она находится в домене `camelot`. Для этого привилегированный процесс операционной системы использует функцию `setdomainname`, которая имеет следующий синтаксис:

```
setdomainname(адрес-имени, длина)
```

Параметр `адрес-имени` задает адрес массива байтов, содержащий имя домена, а параметр `длина` определяет длину этого имени.

Для получения имени домена пользовательские процессы используют функцию `getdomainname`, которая имеет следующий синтаксис:

```
getdomainname(адрес-буфера, длина)
```

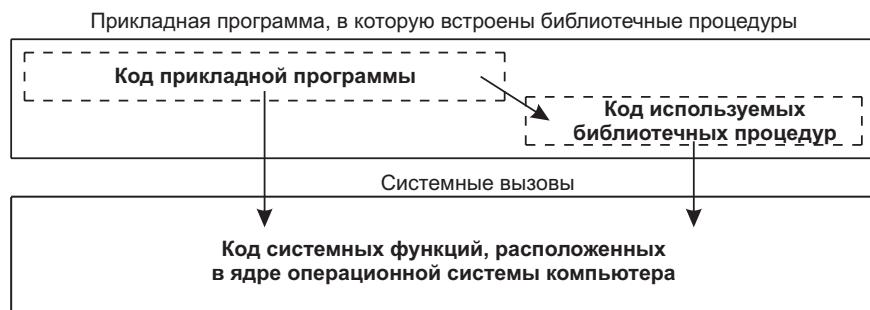
Здесь параметр `адрес-буфера` определяет адрес массива, в который помещается имя домена, а параметр `длина` является целым числом, определяющим длину массива.

## 22.18. Функции библиотеки сокетов

Кроме описанных выше функций в API-интерфейс сокетов включен набор библиотечных процедур, которые выполняют различные полезные функции, связанные с работой в сети. На рис. 22.5 показано, чем отличаются системные вызовы и библиотечные процедуры. Системные вызовы передают управление операционной системе компьютера, в то время как библиотечные процедуры напоминают обычные подпрограммы, которые программисты включают в программу.

Многие процедуры, входящие в библиотеку сокетов, предоставляют доступ к службам баз данных, позволяющих процессу определить имена машин и сетевых служб, номера портов протокола и другую необходимую информацию. Например, существует набор библиотечных процедур, которые предоставляют доступ

к базе данных сетевых служб. Запись в базе данных сетевых служб состоит из трех полей, информация в которых хранится в удобном для восприятия человеком виде: имени сетевой службы, типа используемого протокола и номера стандартного порта протокола этой службы. С помощью соответствующих библиотечных процедур прикладной процесс может извлечь нужную информацию из соответствующего элемента базы данных и представить ее в требуемом виде.



*Рис. 22.5. Отличие между библиотечными процедурами, помещенными в прикладную программу, и системными вызовами, которые являются частью операционной системы. Программа может выполнять оба вызова; библиотечные процедуры могут вызывать другие библиотечные процедуры или выполнять системные вызовы*

В следующих разделах рассматриваются группы библиотечных процедур, объясняется их назначение, а также приведена информация о способе их использования. Как будет показано ниже, наборы библиотечных процедур, представляющих доступ к базе данных с последовательной организацией, функционируют по одному образцу. Каждый набор позволяет приложению:

- подключиться к базе данных;
- извлечь из базы данных содержащуюся в ее записях информацию (по одной записи за одно обращение);
- закрыть соединение с базой данных.

Используемые для выполнения этих операций библиотечные процедуры называются `setXent`, `getXent` и `endXent`, где *X* обозначает имя базы данных. Например, библиотечные программы для работы с базой данных узлов сети (*host database*) называются `sethostent`, `gethostent` и `endhostent`. Поэтому ниже при их описании будет приведена только общая информация о вызове без повторения подробностей использования.

## 22.19. Процедуры для преобразования сетевого порядка следования байтов

Вспомним, что разные типы компьютеров отличаются между собой способом хранения в памяти величин целого типа, а также то, что в семействе протоколов TCP/IP определен независимый порядок следования байтов. Поэтому в API-интерфейсе сокетов предусмотрены четыре библиотечные функции, которые обеспечивают преобразование порядка следования байтов локальной машины в принятый в сети стандарт, и наоборот. Для обеспечения мобильности программ, они должны быть написаны таким образом, чтобы каждый раз при копировании

целого значения из локальной машины в сетевой пакет, или наоборот, вызывались процедуры преобразования порядка следования байтов.

Все четыре процедуры преобразования являются функциями, которые возвращают значение переданного им единственного параметра с перегруппированными байтами. Например, чтобы преобразовать короткое (2-байтовое) целое число из принятого в сети порядка следования байтов в порядок байтов, установленный на локальном узле сети, программист должен вызвать функцию `ntohs` (*network to host short*). Эта функция имеет следующий синтаксис:

```
локальное-значение = ntohs(сетевое-значение)
```

Параметр `сетевое-значение` представляет собой 2-байтовое (16-битовое) целое число, представленное в принятом в сети порядке следования байтов, а возвращаемый параметр `локальное-значение` представлен в порядке байтов, который установлен для локального узла сети.

В языке программирования С 4-байтовые (32-битовые) целые числа имеют тип `long` и называются длинными целыми. Функция `ntohl` (*network to host long*) преобразует 4-байтовые длинные целые числа из принятого в сети порядка байтов в порядок байтов, установленный для локального узла. При вызове функции `ntohl` ей в качестве параметра передается длинное целое число, представленное в сетевом порядке следования байтов:

```
локальное-значение = ntohl(сетевое-значение)
```

Две аналогичные функции позволяют программисту преобразовать порядок байтов целого числа, который принят на локальном узле сети, в сетевой порядок. Функция `htons` преобразует 2-байтовое (короткое) целое число в сетевой порядок следования байтов. При вызове функции `htons` ей в качестве параметра передается короткое целое число:

```
сетевое-значение = htons(локальное-значение)
```

Последняя подпрограмма преобразования, `htonl`, преобразует длинные целые числа в принятый в сети порядок следования байтов. Подобно другим подпрограммам, `htonl` представляет собой функцию:

```
сетевое-значение = htonl(локальное-значение)
```

Очевидно, что в подпрограммах преобразования должны сохраняться следующие математические соотношения:

```
сетевое-значение = htons(ntohs(сетевое-значение))
```

и

```
локальное-значение = ntohs(htons(локальное-значение))
```

Похожие соотношения сохраняются и для подпрограмм преобразования длинных целых чисел.

## 22.20. Подпрограммы обработки IP-адресов

Поскольку во многих приложениях требуется выполнить преобразование 32-битового IP-адреса в точечный десятичный формат, и наоборот, в библиотеку сокетов входят соответствующие процедуры, выполняющие такое преобразование. Процедуры `inet_addr` и `inet_network` преобразуют IP-адрес из точечного десятичного формата в 32-битовое целое число, представленное в сетевом порядке следования байтов. Процедура `inet_addr` формирует 32-битовый IP-адрес узла сети, а процедура `inet_network` формирует 32-битовый IP-адрес сети, в котором

биты, относящиеся к узлу сети, заменены нулями. Процедуры имеют следующий синтаксис:

```
IP-адрес = inet_addr(строка)
и
IP-адрес = inet_network(строка)
```

Здесь параметр *строка* задает адрес ASCII-строки, содержащей IP-адрес, представленный в точечной десятичной форме записи. В этой форме IP-адрес может иметь от 1 до 4 сегментов цифр, разделенных точками. Если представление состоит из четырех сегментов, то каждый из них соответствует одному байту полученного в результате 32-битового целого числа. Если сегментов меньше четырех, в оставшиеся байты помещается значение последнего сегмента.

Процедура *inet\_ntoa* осуществляет обратную процедуре *inet\_addr* операцию — преобразует 32-х битовое целое число в ASCII-строку, содержащую IP-адрес, представленный в точечной десятичной форме записи. Эта процедура имеет следующий синтаксис:

```
строка = inet_ntoa(IP-адрес)
```

Здесь параметр *IP-адрес* — 32-битовый IP-адрес, представленный в сетевом порядке следования байтов, а параметр *строка* — адрес буфера, куда будет помещена сформированная ASCII-строка.

Часто обрабатывающие IP-адреса программы должны объединить адрес сети с локальным адресом узла и сформировать единый IP-адрес. Такое объединение выполняет процедура *inet\_makeaddr*. Она имеет следующий синтаксис:

```
IP-адрес = inet_makeaddr(адрес-сети, адрес-узла)
```

Параметр *адрес-сети* представляет собой 32-битовый IP-адрес сети, заданный в локальном порядке следования байтов, а параметр *адрес-узла* — целое число, представляющее адрес локального узла данной сети, выраженный в локальном порядке следования байтов.

Процедуры *inet\_netof* и *inet\_lnaof* выполняют операцию обратную процедуре *inet\_makeaddr*, разделяя сетевую и локальную часть IP-адреса. Они имеют следующий синтаксис:

```
адрес-сети = inet_netof(IP-адрес)
```

и

```
адрес-узла = inet_lnaof(IP-адрес),
```

Здесь параметр *IP-адрес* представляет собой 32-битовый IP-адрес, заданный в сетевом порядке следования байтов, а полученные значения возвращаются в локальном порядке следования байтов.

## 22.21. Получение доступа к системе доменных имен<sup>7</sup>

Для доступа к системе доменных имен семейства протоколов TCP/IP в библиотеке сокетов предусмотрен набор из пяти процедур. Прикладные программы, вызывающие эти подпрограммы, становятся клиентами одной большой системы доменных имен, отсылая запросы одному или нескольким серверам и получая на них ответы.

Идея заключается в том, что программа делает запрос одному серверу и ожидает от него ответа. Поскольку при этом существует большое количество разных

<sup>7</sup> Система доменных имен подробно рассматривается в главе 24.

параметров, в подпрограммы передаются только основные параметры, а остальные хранятся в глобальной структуре, которая называется `res`. Например, одно из полей структуры `res` отвечает за рассылку отладочных сообщений, в то время как другое позволяет установить тип протокола (UDP или TCP), который будет использоваться для отправки запросов прикладной программой. Для большинства полей структуры `res` назначены подходящие значения по умолчанию, поэтому без особой необходимости их менять не нужно.

Перед использованием других процедур прикладная программа должна вызывать функцию `res_init` без параметров:

```
res_init()
```

Функция `res_init` считывает файл конфигурации, содержащий такую информацию, как имя машины, на которой запущен сервер доменных имен, и сохраняет результаты в глобальной структуре `res`.

Процедура `res_mkquery` формирует запрос к серверу доменных имен и помещает его во внутренний буфер памяти. Она имеет следующий синтаксис:

```
res_mkquery(код, адрес-имени, класс, тип, адрес-данных, длина-данных,  
не-исп, адрес-буфера, длина-буфера)
```

Первые семь параметров соответствуют полям запроса, посылаемого серверу доменных имен. Параметр `код` определяет затребованную операцию; параметр `адрес-имени` задает адрес массива символов, содержащего доменное имя; параметр `класс` — это целое число, которое указывает класс запроса; параметр `тип` — целое число, которое указывает тип запроса; параметр `адрес-данных` указывает адрес массива данных, которые будут включены в запрос, а параметр `длина-данных` представляет собой целое число, которое определяет длину данных. Помимо библиотечных процедур, в API сокетов предусмотрены специальные символические определения для важных констант, которые используются в прикладных программах. Таким образом, программисты могут использовать систему доменных имен, не располагая при этом глубокими знаниями о деталях работы протокола. Два последних параметра, `адрес-буфера` и `длина-буфера`, определяют, соответственно, адрес области, в которую необходимо поместить запрос, и выраженную целым числом длину этой области. Следует отметить, что в текущей реализации параметр `не-исп` не используется.

Сформировав запрос, программа вызывает функцию `res_send`, чтобы отослать его серверу доменных имен и получить на него ответ. Эта функция имеет следующий синтаксис:

```
res_send(адрес-буфера, длина-буфера, адрес-ответа, длина-ответа)
```

Параметр `адрес-буфера` является указателем на область памяти, в которой хранится предназначенное для отсылки сообщение, сформированное приложением (обычно с помощью вызова процедуры `res_mkquery`). Параметр `длина-буфера` — целое число, определяющее длину сообщения. Параметр `адрес-ответа` задает адрес буфера, в который будет помещен ответ сервера, а целочисленный параметр `длина-ответа` определяет длину этого буфера.

Помимо подпрограмм, создающих и отсылающих запросы, в библиотеку сокетов входят две подпрограммы, которые выполняют преобразование доменных имен между стандартным форматом ASCII и сжатым форматом, используемым в запросах. Процедура `dn_expand` расширяет доменное имя и представляет его в виде ASCII-строки. Эта процедура имеет следующий синтаксис:

```
dn_expand(адрес-сообщения, конец-сообщения, адрес-сжатого-имени, адрес-  
буфера, длина-буфера)
```

Параметр *адрес-сообщения* задает адрес сообщения службы доменных имен, который содержит имя для расширения. Параметр *конец-сообщения* определяет границы конца сообщения, за пределы которых расширение продолжаться не может. Параметр *адр-сжатого-имени* является указателем на первый байт сжатого имени. Параметр *адрес-буфера* — это адрес массива, в который необходимо записать расширенное имя, а параметр *длина-буфера* — целое число, определяющее длину массива.

Сжатие имени — процесс более сложный, чем его последующее расширение, поскольку для сжатия необходимо удалить общие окончания. При сжатии имен клиент должен учесть появившихся ранее окончаний. Сжатие полного доменного имени выполняется с помощью процедуры `dn_comp`. При этом окончания сверяются со списком использованных ранее окончаний, и самое длинное возможное окончание удаляется. Вызов этой процедуры имеет следующий синтаксис:

```
dn_comp(адрес-имени,adr-сжатого-имени,длина-сжатого-имени,адр-мас-указ,указ-конца-мас)
```

Параметр *адрес-имени* задает адрес полного доменного имени. Параметр *адр-сжатого-имени* указывает на массив байтов, в котором будет храниться сжатое имя, а параметр *длина-сжатого-имени* определяет длину этого массива. Параметр *адр-мас-указ* — адрес массива указателей на ранее сжатые окончания, а параметр *указ-конца-мас* содержит адрес конца массива. Как правило, процедура `dn_comp` сжимает имя и обновляет массив указателей, адрес которого задан параметром *адр-мас-указ*, если было использовано новое окончание.

Процедуру `dn_comp` можно также использовать для преобразования доменного имени из формата ASCII во внутреннюю форму, не прибегая при этом к сжатию (т.е. не удаляя окончания). Для этого при ее вызове в качестве параметра *адр-мас-указ* необходимо указать значение `NULL` (т.е. двоичный нуль).

## 22.22. Получение информации об узлах сети

Существуют библиотечные процедуры, позволяющие процессу извлекать информацию об узле сети, если известно его доменное имя или IP-адрес. При использовании библиотечных программ на машине, которая располагает доступом к серверу доменных имен, процесс становится клиентом системы доменных имен. Для этого библиотечные программы посыпают серверу запрос и ожидают ответа на него. При использовании библиотечных программ в системах, не имеющих доступа к системе доменных имен (например, на машине, не подключенной к сети Internet), программы получают необходимую информацию из базы данных, которая хранится на вспомогательном запоминающем устройстве.

Функция `gethostbyname` возвращает указатель на структуру, содержащую информацию об узле сети, имя которого передано ей в качестве параметра. Вызов функции имеет следующий синтаксис:

```
указатель = gethostbyname(адрес-имени)
```

Параметр *адрес-имени* — это указатель на строку символов, которая содержит доменное имя узла сети. Возвращаемое функцией значение является указателем на структуру, которая содержит следующую информацию: официальное имя узла сети, список зарегистрированных для узла сети псевдонимов, тип адреса узла сети (т.е. является ли его адрес IP-адресом), длину адреса и список одного или нескольких адресов для этого узла сети. Подробную информацию об этой функции можно найти в справочнике для программиста системы UNIX (*UNIX Programmer's Manual*).

С помощью функции `gethostbyaddr` можно получить такую же информацию, как и с помощью функции `gethostbyname`. Отличие между ними заключается в том, что функции `gethostbyaddr` в качестве параметра передается адрес узла сети:

```
указатель = gethostbyaddr (адрес, длина, тип)
```

Параметр `адрес` — это указатель на последовательность байтов, содержащую адрес узла сети. Параметр `длина` — целое число, которое задает длину адреса, а параметр `тип` — целое число, определяющее тип адреса (например, то, что адрес является IP-адресом).

Как уже упоминалось выше, последовательный доступ к базе данных узлов сети выполняется с помощью процедур `sethostent`, `gethostent` и `endhostent`.

## 22.23. Получение информации о сетях

Для получения информации о сетях узлы сети либо используют систему доменных имен, либо ведут свою несложную базу данных. В ней хранится информация обо всех сетях объединенной сети, к которой эти узлы принадлежат. В библиотеку сокетов входит пять процедур, которые позволяют процессу получить доступ к базе данных сетей. Функция `getnetbyname` извлекает и форматирует содержимое элемента базы данных по указанному доменному имени сети. Вызов этой функции имеет следующий синтаксис:

```
указатель = getnetbyname (адрес-имени)
```

Здесь параметр `адрес-имени` является указателем на строку, содержащую доменное имя сети, для которой необходимо получить информацию. Возвращаемое функцией значение представляет собой указатель на структуру, в полях которой хранится официальное имя сети, список зарегистрированных псевдонимов, тип адреса, выраженный числом целого типа, и 32-битовые адреса сети (т.е. IP-адреса, в которых часть, относящаяся к узлу сети, обозначена нулем).

Процесс вызывает библиотечную процедуру `getnetbyaddr`, когда ему необходимо найти информацию о сети по ее адресу. Вызов этой функции имеет следующий синтаксис:

```
указатель = getnetbyaddr (адрес-сети, тип-адреса)
```

Параметр `адрес-сети` — это 32-битовый адрес сети, а параметр `тип-адреса` — целое число, определяющее тип предыдущего параметра. Процедуры `setnetent`, `getnetent` и `endnetent` предоставляют последовательный доступ к базе данных сетей.

## 22.24. Получение информации о протоколах

Доступ к базе данных протоколов, которая хранится на каждом узле сети, выполняется с помощью пяти библиотечных процедур. Каждый протокол имеет свое официальное имя, зарегистрированные псевдонимы и официальный номер протокола. Процедура `getprotobyname` позволяет вызывающей программе получить информацию о протоколе по его имени:

```
указатель = getprotobyname (адрес-имени)
```

Параметр `адрес-имени` — это указатель на ASCII-строку, содержащую имя протокола, о котором необходимо получить информацию. Функция возвращает указатель на структуру, в полях которой указывается официальное имя протокола, список его псевдонимов, а также уникальный целочисленный номер, присвоенный протоколу.

Процедура под названием `getprotobyname` позволяет процессу отыскать информацию о протоколе, используя в качестве параметра его номер:

```
указатель = getprotobyname(номер-протокола)
```

И наконец, последовательный доступ к базе данных протоколов выполняется с помощью процедур `getprotoent`, `setprotoent` и `endprotoent`.

## 22.25. Получение информации о сетевых службах

В главах 12, “Передача пользовательских дейтаграмм (UDP)”, и, 13, “Надежная потоковая транспортная служба (TCP)”, речь шла о том что некоторые номера портов в протоколах UDP и TCP зарезервированы для стандартных служб. Например, порт номер 43 протокола TCP резервируется для службы `whois` (*who is who, или кто есть кто*). Служба `whois` позволяет клиенту, запущенному на одной машине, связываться с сервером, работающим на другой машине, и получать информацию о пользователе, который имеет учетную запись на машине сервера. В базе данных служб для службы `whois` определено официальное имя “`whois`”, протокол TCP и номер порта — 43. Существует пять библиотечных процедур, с помощью которых можно получить информацию о службах и используемых ими портах протокола.

Процедура `getservbyname` позволяет определить по имени службы ее стандартный номер порта протокола:

```
указатель = getservbyname(адрес-имени, номер-протокола)
```

Параметр `адрес-имени` определяет адрес строки, содержащей имя службы, информацию о которой необходимо получить, а целочисленный параметр `номер-протокола` определяет протокол, с которым следует использовать эту службу. Как правило, выбор протоколов ограничивается протоколами TCP и UDP. Возвращаемое процедурой значение представляет собой указатель на структуру, где содержатся имя службы, список ее псевдонимов, идентификатор протокола, с которым используется служба, а также присвоенный этой службе целочисленный номер порта протокола.

Процедура `getservbyport` позволяет вызывающей программе извлечь запись из базы данных служб по присвоенному ей номеру порта. Вызов этой функции имеет следующий синтаксис:

```
указатель = getservbyport(номер-порта, номер-протокола)
```

Параметр `номер-порта` представляет собой присвоенный службе целочисленный номер порта протокола, а параметр `номер-протокола` определяет идентификатор протокола, с которым необходимо использовать службу. Как и в случае других баз данных, процесс может получить последовательный доступ к базе данных служб, используя для этого функции `setservent`, `getservent` и `endservent`.

## 22.26. Пример клиентской части программы

В приведенном ниже примере программы, написанной на языке C, показано, как в приложениях с помощью API-интерфейса сокетов получать доступ к протоколам семейства TCP/IP. В качестве примера приведена программа, которая представляет собой несложную реализацию клиента и сервера службы `whois`. Согласно определению, приведенному в [RFC 954], служба `whois` позволяет клиенту на одной машине получать информацию о пользователе удаленной машины. В данной реализации клиент представляет собой прикладную программу, которую вызывает пользователь, указывая при этом два параметра: имя удаленной

машины и имя пользователя на этой машине, о котором необходимо получить информацию. Программа-клиент вызывает функцию `gethostbyname`, чтобы преобразовать имя удаленной машины в IP-адрес, и функцию `getservbyname`, — чтобы узнать стандартный номер порта службы `whois`. После преобразование имен узла сети и службы клиент создает сокет, указывая, что в сокете будет использоваться надежная потоковая служба доставки (например, протокол TCP). Затем клиент связывает сокет с портом протокола службы `whois` удаленной машины.

```
/* whoisclient.c - main */

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

/*-----
 * Программа: whoisclient
 *
 * Назначение: Приложение для системы UNIX, которое
 * является клиентом службы "whois" сети Internet.
 *
 * Использование: whois имя-узла имя-пользователя
 *
 * Автор: Барри Шейн (Barry Shein), Бостонский Университет
 *
 * Дата написания: Очень давно, еще во время учебы в университете
 *
*-----
 */
main(argc, argv)
int argc; /* Описание стандартных параметров системы UNIX */
char *argv[];
{
    int s; /* Дескриптор сокета */
    int len; /* Длина полученных данных */
    struct sockaddr_in sa; /* Адресная структура сокета Internet */
    struct hostent *hp; /* Результат поиска имени сетевого узла */
    struct servent *sp; /* Результат поиска службы */
    char buf[BUFSIZ+1]; /* Буфер для записи информации,
                           полученной от службы whois */
    char *myname; /* Указатель на имя этой программы */
    char *host; /* Указатель на имя удаленного узла сети */
    char *user; /* Указатель на имя удаленного пользователя */
    myname = argv[0];
    /*
     * Убедимся, что существует два аргумента командной строки
     */
    if(argc != 3) {
        fprintf(stderr, "Usage: %s host username\n", myname);
        exit(1);
    }
    host = argv[1];
    user = argv[2];
    /*
     * Найдем указанное имя узла сети
     */
    if((hp = gethostbyname(host)) == NULL) {
```

```

        fprintf(stderr, "%s: %s: no such host?\n", myname, host);
        exit(1);
    }
    /*
     * Поместим адрес узла сети его тип в адресную структуру сокета
     */
    bcopy((char *)hp->h_addr, (char *)&sa.sin_addr, hp->h_length);
    sa.sin_family = hp->h_addrtype;
    /*
     * Найдем номер порта для службы WHOIS
     */
    if((sp = getservbyname("whois", "tcp")) == NULL) {
        fprintf(stderr, "%s: No whois service on this host\n", myname);
        exit(1);
    }
    /*
     * Поместим номер порта службы whois в адресную структуру сокета.
     */
    sa.sin_port = sp->s_port;
    /*
     * Разместим в памяти адресную структуру открытого сокета
     */
    if((s = socket(hp->h_addrtype, SOCK_STREAM, 0)) < 0) {
        perror("socket");
        exit(1);
    }
    /*
     * Подключимся к удаленному серверу
     */
    if(connect(s, &sa, sizeof sa) < 0) {
        perror("connect");
        exit(1);
    }
    /*
     * Посыпаем запрос
     */
    if(write(s, user, strlen(user)) != strlen(user)) {
        fprintf(stderr, "%s: write error\n", myname);
        exit(1);
    }
    /*
     * Прочитаем ответ и поместим его в выходной буфер
     */
    while((len = read(s, buf, BUFSIZ)) > 0)
        write(1, buf, len);
    close(s);
    exit(0);
}

```

## 22.27. Пример серверной части программы

Пример серверной части программы лишь незначительно сложнее примера клиентской части. Сервер прослушивает сообщения, поступающие через стандартный порт службы whois, и в ответ на поступивший от клиента запрос возвращает затребованную информацию. Информация извлекается из файла паролей (passwd) системы UNIX, находящегося на машине сервера.

```

/* whoisserver.c - main */

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <pwd.h>

/*
 * Программа: whoisserver
 *
 * Назначение: Приложение для системы UNIX, которое
 * выполняет функции сервера службы
 * "whois" локальной машины.
 * Программа прослушивает сообщения, поступающие через
 * стандартный порт WHOIS (43), и отвечает
 * на запросы клиентов. Для выполнения этой программы
 * необходимы права суперпользователя.
 *
 * Использование: whois имя-узла имя-пользователя
 *
 * Автор: Барри Шейн (Barry Shein), Бостонский Университет
 *
 * Дата написания: Очень давно, еще во время учебы в университете
 *
 */
#define BACKLOG      5 /* Количество запросов, которые мы
                     хотим поставить в очередь */
#define MAXHOSTNAME 32 /* Максимально допустимая длина имени узла сети */

main(argc, argv)
int argc; /* Описание стандартных параметров системы UNIX */
char *argv[];
{
    int s, t; /* Дескрипторы сокета */
    int i; /* Целая переменная общего назначения */
    struct sockaddr_in sa, isa; /* Адресная структура сокета Internet */
    struct hostent *hp; /* Результат поиска имени узла сети */
    char *myname; /* Указатель на имя этой программы */
    struct servent *sp; /* Результат поиска службы */
    char localhost[MAXHOSTNAME+1]; /* Имя локального узла сети,
                                    представленное в виде строки символов */

    myname = argv[0];
    /*
     * Найдем информацию о службе WHOIS
     */
    if((sp = getservbyname("whois","tcp")) == NULL) {
        fprintf(stderr, "%s: No whois service on this host\n", myname);
        exit(1);
    }
    /*
     * Получим информацию о нашем узле сети
     */
    gethostname(localhost, MAXHOSTNAME);
}

```

```

if((hp = gethostbyname(localhost)) == NULL) {
    fprintf(stderr, "%s: cannot get local host info?\n", myname);
    exit(1);
}
/*
 * Поместим номер порта службы WHOIS и информацию о нашем
 * адресе в структуру сокета
 */
sa.sin_port = sp->s_port;
bcopy((char *)hp->h_addr, (char *)&sa.sin_addr, hp->h_length);
sa.sin_family = hp->h_addrtype;
/*
 * Разместим открытый сокет для входящих соединений
 */
if((s = socket(hp->h_addrtype, SOCK_STREAM, 0)) < 0) {
    perror("socket");
    exit(1);
}
/*
 * Привязать сокет к порту службы,
 * чтобы можно было прослушивать запросы на соединение
 */
if(bind(s, &sa, sizeof sa) < 0) {
    perror("bind");
    exit(1);
}
/*
 * Зададим максимальное количество соединений,
 * которое может обработать сервер
 */
listen(s, BACKLOG);
/*
 * Входим в бесконечный цикл ожидания новых соединений
 */
while(1) {
    i = sizeof isa;
    /*
     * Для ожидания новых клиентов вызываем функцию accept()
     */
    if((t = accept(s, &isa, &i)) < 0) {
        perror("error");
        exit(1);
    }
    whois(t); /* Выполнить запрос службы WHOIS */
    close(t);
}
}

/*
 * Получим запрос к серверу WHOIS от удаленного узла сети
 * и отформатируем ответ
 */
whois(sock)
int sock;
{
    struct passwd *p;
    char buf[BUFSIZ+1];
    int i;

```

```

/*
 * Получить одну строку запроса
 */
if( (i = read(sock, buf, BUFSIZ)) <= 0)
    return;
buf[i] = '\0'; /* Символ конца строки */
/*
 * Найдем затребованного пользователя и отформатируем ответ
 */
if((p = getpwnam(buf)) == NULL)
    strcpy(buf, "User not found\n");
else
    sprintf(buf, "%s: %s\n", p->pw_name, p->pw_gecos);
/*
 * Возвратим ответ
 */
write(sock, buf, strlen(buf));
return;
}

```

## 22.28. Резюме

Поскольку программы поддержки семейства протоколов TCP/IP находятся внутри операционной системы, реализация конкретного интерфейса между прикладной программой и семейством протоколов TCP/IP зависит от типа операционной системы и не является стандартом семейства протоколов TCP/IP. В этой главе был рассмотрен API-интерфейс сокетов, который первоначально был разработан для версии BSD UNIX, но фактически превратился в стандарт, используемый производителями программного обеспечения, в том числе и фирмой Microsoft. В этой главе было показано, что с помощью сокетов реализован принцип ввода-вывода системы UNIX, который называется *open-read write-close*. Чтобы использовать протокол TCP, программа должна создать сокет, привязать к нему адреса, принять запрос на входящее соединение, а затем перейти к взаимодействию посредством использования базовых функций *read* или *write*. В конечном счете, завершив использование сокета, программа должна его закрыть. Помимо абстрактного понятия сокета и системных вызовов, работающих с сокетом, в версию BSD UNIX также входит ряд библиотечных процедур, которые помогают программистам формировать IP-адреса и манипулировать ими, изменять порядок следования байтов в целых числах в соответствии с тем, который принят в сети и на локальной машине, а также отыскивать различную информацию, такую как адреса сети и т.п.

Интерфейс сокетов приобрел популярность и, в значительной степени, поддержку многих производителей. Производители, не поддерживающие механизм сокетов в своих операционных системах, зачастую предоставляют библиотеку сокетов, что позволяет программистам писать приложения, используя для этого вызовы из библиотеки сокетов, даже если базовая операционная система использует другой набор системных вызовов.

## Материал для дальнейшего изучения

Подробную информацию о функциях для работы с сокетами можно найти в справочнике программиста системы UNIX (*UNIX Programmer's Manual*). В разделе 2 этого справочника имеется описание каждого вызова системы UNIX, а в разделе 3 можно найти описание всех библиотечных процедур. В системе UNIX

также существует электронная копия этого справочника, доступ к которой можно получить с помощью команды `man`. Подробный обзор системы UNIX выполнен Леффлером (Leffler), Мак-Кусиком (MacKusick), Карелсом (Karels) и Квотерманом (Quarterman) в книге [83].

Производители операционных систем часто предоставляют библиотеки процедур, выполняющих в их системах эмуляцию функций сокетов. Чтобы получить более подробную информацию, обратитесь к разработанным этими производителями справочникам по программированию. Расширения сокетов для версии IPv6 рассмотрены Джиллигеном (Gilligan) в [RFC 2133].

В третьем томе этой книги описано, как построены программа-клиент и программа-сервер, а также объяснено, каким образом они используют API-интерфейс сокетов. Описанная в томе 3 версия сокетов BSD включает в себя пример кода для системы UNIX; в версию сокетов Windows входят такие же примеры для системы Microsoft Windows. В версии тома 3 для TLI (*Transport Layer Interface*, или *интерфейс транспортного уровня*) представлены основные понятия об интерфейсе транспортного уровня, который является альтернативным вариантом сокетов, используемых в системе Unix System V.

## Упражнения

- 22.1.** Попробуйте запустить на вашем локальном компьютере пример программы клиента и сервера службы `whois`.
- 22.2.** Создайте несложный сервер, обрабатывающий несколько одновременно поступивших запросов. Чтобы протестировать его, запустите процесс, который обрабатывает текущий запрос, выводит краткое сообщение, устанавливает случайное время задержки, выводит следующее сообщение и выходит из программы.
- 22.3.** Для чего используется функция `listen`?
- 22.4.** Какие библиотечные процедуры предусмотрены на вашем локальном компьютере для доступа к системе доменных имен?
- 22.5.** Продумайте алгоритм работы сервера, который использует только один процесс системы UNIX, но может обрабатывать несколько одновременных TCP-соединений. (*Подсказка*. Вспомните о назначении функции `select` (в системах Unix System V вместо нее используется функция `poll`).
- 22.6.** Изучите описание интерфейса транспортного уровня (TLI) системы AT&T Unix System V и сравните его с интерфейсом сокетов. В чем принципиальные различия?
- 22.7.** Каждая операционная система налагает ограничения на количество одновременно открытых прикладной программой сокетов. Сколько сокетов может открыть программа на вашем локальном компьютере?
- 22.8.** Механизм дескриптора сокета/файла и связанные с ним операции `read` и `write` можно считать формой объектно-ориентированного проектирования. Объясните, почему.
- 22.9.** Рассмотрите альтернативную структуру интерфейса, в которой предусмотрен собственный интерфейс программ поддержки протокола каждого из уровней (например, система позволяет прикладной программе посыпать и получать низкоуровневые пакеты, не используя протокол IP, или посыпать и получать IP-дейтаграммы без помощи протокола UDP или TCP). В чем преимущества и недостатки такого интерфейса?

- 22.10.** Программа-клиент и программа-сервер могут выполняться на одном и том же компьютере и использовать для взаимодействия TCP-сокет. Объясните, как создать программу-клиент и программу-сервер, чтобы они могли взаимодействовать между собой на одной машине, не располагая при этом информацией об IP-адресе этой машины.
- 22.11.** Поэкспериментируйте с описанным в этой главе примером сервера и проверьте, сможет ли ваша программа генерировать такой поток TCP-соединений, который превысил бы установленные сервером лимиты на количество необработанных запросов. Как вы думаете, в каком из случаев выше вероятность переполнения очереди запросов на обработку — когда сервер функционирует на компьютере с одним процессором или с несколькими (например, пятью)? Поясните свой ответ.



# 23

## *Начальная загрузка и автоконфигурация (BOOTP, DHCP)*

### **23.1. Введение**

В этой главе продемонстрировано, как принцип клиент/сервер используется в процессе начальной загрузки компьютера. Мы уже знаем, что, прежде чем компьютер, подключенный к объединенной сети TCP/IP, сможет отсылать или получать дейтаграммы, ему должен быть назначен уникальный IP-адрес. Кроме этого компьютеру необходима и другая информация — адрес ближайшего маршрутизатора, текущая маска подсети и адрес сервера доменных имен. В главе 6, “Определение IP-адреса при начальной загрузке (RARP)”, было описано, как в процессе начальной загрузки компьютер может определить свой IP-адрес при с помощью протокола RARP. В этой главе рассмотрен альтернативный вариант: два тесно связанных между собой протокола начальной загрузки, каждый из которых позволяет узлу сети определить свой IP-адрес, не используя при этом протокол RARP. Удивительно то, что клиент и сервер взаимодействуют при этом между собой посредством протокола UDP (User Datagram Protocol, или протокол передачи пользовательских дейтаграмм), описанного в главе 12, “Передача пользовательских дейтаграмм (UDP)”.

Особенностью процесса начальной загрузки является то, что для передачи сообщений в протоколе UDP используется протокол IP. Сам факт, что для нахождения своего IP-адреса, необходимого для выполнения взаимодействия, компьютер может использовать протокол UDP, может показаться невероятным. Тем не менее это возможно благодаря гибкости механизма транспортировки UDP/IP и использованию особых IP-адресов, о которых упоминалось в главе 4, “Классовая адресация”. В этой главе будет также показано, каким образом сервер может автоматически назначить компьютеру IP-адрес. Такое назначение является особенно важным в тех сетевых окружениях, где разрешены временные подключения к объединенной сети или компьютеры часто перемещаются из одной сети в другую (например, служащий с портативным компьютером переезжает из одного офиса компании в другой).

### **23.2. Необходимость в альтернативе протоколу RARP**

В главе 6, “Определение IP-адреса при начальной загрузке (RARP)”, описана проблема, с которой сталкиваются бездисковые рабочие станции во время начальной загрузки системы. Обычно программа начальной загрузки таких машин находится в энергонезависимом запоминающем устройстве (например, в микросхеме

ПЗУ, которая расположена на плате сетевого интерфейса). С целью минимизации затрат и поддержки взаимозаменяемости частей производители используют одинаковую программу на всех машинах. Поскольку одна и та же программа начальной загрузки должна выполняться на машинах с разными IP-адресами, IP-адрес не может быть жестко “зашит” в машинном коде, содержащемся в ПЗУ. Следовательно, бездисковая машина должна получить свой IP-адрес другим путем. Кроме того, для выполнения начальной загрузки бездисковому компьютеру необходимо “знать” не только свой IP-адрес, но и другую информацию. Как правило, в микросхеме ПЗУ находится только небольшая программа начального запуска. Поэтому бездисковый компьютер также должен загрузить по сети образ памяти программы начальной загрузки и передать ей управление. Более того, каждая бездисковая машина должна как-то определить адрес файлового сервера, где она может хранить данные, и адрес ближайшего IP-маршрутизатора.

Описанный в главе 6, “Определение IP-адреса при начальной загрузке (RARP)”, протокол RARP имеет три недостатка. Во-первых, поскольку он функционирует на самом низком уровне, использующая его прикладная программа должна напрямую взаимодействовать с сетевым оборудованием. Это создает дополнительные трудности для программиста, что в конечном итоге затрудняет реализацию программы начальной загрузки или делает ее вообще невозможной. Во-вторых, хотя в протоколе RARP предусмотрен обмен пакетами между машиной клиента и компьютером, отвечающим на ее запрос, в ответном сообщении содержится только один небольшой фрагмент информации — 4 октета адреса клиента. Этот недостаток вызывает особые неудобства в сетях типа Ethernet, для которых существует такое понятие, как минимальный размер пакета. А это значит, что в ответном пакете можно дополнительность переслать информацию без каких-либо дополнительных затрат, поскольку часть пространства пакета попросту не используется. В-третьих, поскольку в протоколе RARP для идентификации машины отправителя используется аппаратный адрес сетевой платы, его нельзя использовать в тех сетях, где аппаратные адреса назначаются динамически.

Чтобы компенсировать недостатки протокола RARP, исследователи разработали новый протокол начальной загрузки (*Bootstrap Protocol*, или *BOOTP*). Позже был создан протокол динамической конфигурации узла сети (*Dynamic Host Configuration Protocol*, или *DHCP*), который пришел на смену протоколу BOOTP. Поскольку эти протоколы тесно связаны между собой, большая часть материала, описанного в этой главе, относится к обоим протоколам. Чтобы упростить описание, мы сначала рассмотрим протокол BOOTP, а затем покажем, каким образом протокол DHCP расширяет его функциональные возможности и обеспечивает динамическое назначение адресов.

Поскольку в протоколе BOOTP используются протоколы UDP и IP, его можно реализовать с помощью обычной прикладной программы, а не как часть операционной системы. В основу работы протокола BOOTP, также как и RARP, положен принцип взаимодействия типа клиент/сервер; при этом системы обмениваются пакетами только один раз. Несмотря на использование высокоуровневых транспортных протоколов, протокол BOOTP эффективнее, чем RARP, поскольку в одном сообщении протокола BOOTP указывается вся информация, которая необходима для начальной загрузки и инициализации машины, включая IP-адрес компьютера, адрес маршрутизатора и адрес сервера. В ответном сообщении протокола BOOTP также присутствует поле, формат которого определяется производителем оборудования. В это поле производители аппаратного обеспечения могут помещать дополнительную информацию, которая используется только для их типа оборудования<sup>1</sup>.

---

<sup>1</sup> Как будет показано ниже, название этого поля (*Определяемое производителем*), не совсем удачно, поскольку в текущей спецификации стандарта рекомендуется использовать это поле

### **23.3. Определение IP-адреса с помощью протокола IP**

Уже было сказано, что в протоколе BOOTP для передачи сообщений используется протокол UDP и что сообщения протокола UDP инкапсулируются в IP-дейтаграммы для последующей доставки получателю по сети. Чтобы понять, как компьютер посылает сообщение протокола BOOTP в IP-дейтаграмме, не зная своего IP-адреса, обратимся к главе 4, “Классовая адресация”. Там было отмечено, что существует несколько особых IP-адресов. В частности, для указания режима ограниченной широковещательной доставки используется IP-адрес получателя, состоящий из всех единиц: 255.255.255.255. Нюанс состоит в том, что программы поддержки протокола IP, запущенные на узлах сети, могут принимать и широковещательно отсыпать дейтаграммы по ограниченному широковещательному адресу, даже если узлу сети еще не назначен IP-адрес. Суть заключается в следующем.

*Прикладная программа может использовать ограниченный широковещательный IP-адрес, чтобы заставить программу поддержки протокола IP отослать дейтаграмму в широковещательном режиме по локальной сети. При этом узел сети, на котором работает прикладная программа, может не располагать сведениями о своем IP-адресе.*

Предположим, что клиентский компьютер *A* собирается использовать протокол BOOTP для поиска информации, необходимой ему для начальной загрузки (в том числе и своего IP-адреса). Предположим также, что компьютер *B* представляет собой подключенный к той же физической сети сервер, который будет отвечать на запрос. Поскольку машине *A* не известен IP-адрес сервера *B* или IP-адрес сети, она должна отослать свой первоначальный BOOTP-запрос в широковещательном режиме передачи, используя для этого ограниченный широковещательный IP-адрес. А что происходит с ответным сообщением? Может ли сервер *B* послать ответ непосредственно машине *A*? Как правило нет. Не совсем понятно, почему для отсылки ответного сообщения машине *A* сервер *B* должен использовать ограниченный широковещательный адрес, хотя IP-адрес машины *A* ему известен. Чтобы найти объяснение, рассмотрим, что произойдет, если прикладная программа, запущенная на сервере *B*, попытается отослать дейтаграмму машине *A* по ее IP-адресу. После выполнения процесса маршрутизации дейтаграммы на сервере *B*, программы поддержки протокола IP перешлют ее драйверу сетевого интерфейса. Драйвер интерфейса должен преобразовать IP-адрес ближайшей точки перехода в соответствующий физический адрес, по всей вероятности используя для этого протокол ARP, как описано в главе 5, “Преобразование IP-адресов в физические адреса (ARP)”. Однако, поскольку машина *A* еще не получила ответного сообщения протокола BOOTP, она не знает своего IP-адреса, и поэтому не может ответить на отосланный сервером *B* ARP-запрос. Следовательно, у сервера *B* есть только два возможных варианта: переслать ответное сообщение по сети в широковещательном режиме или извлечь информацию из пакета BOOTP-запроса, посланного машиной *A*, и поместить ее в свой кэш протокола ARP. Если операционная система компьютера не разрешает прикладным программам модифицировать кэш протокола ARP, широковещательный режим передачи оказывается единственным решением.

---

для универсальной информации, такой как маска подсети. Поэтому в протоколе DHCP называние этого поля изменено на *Поле параметров*.

## 23.4. Принцип повторной передачи в протоколе BOOTP

В протоколе BOOTP вся ответственность за надежный обмен информацией возлагается на клиента. Поскольку в протоколе UDP для доставки пакетов используется ненадежный протокол IP, сообщения могут задержаться в пути, потеряться, может быть нарушен их порядок доставки либо появиться их дубли. Более того, поскольку в протоколе IP не предусмотрена контрольная сумма для области данных, существует вероятность того, что в поступившей дейтаграмме протокола UDP значение некоторых битов будет искажено. Чтобы предотвратить возможное искажение данных, протокол BOOTP требует, чтобы в протоколе UDP использовались контрольные суммы. В протоколе также определено, что для тех клиентов, у которых недостаточно памяти для сборки дейтаграмм, в заголовке запроса и ответа должен быть установлен бит *запрета фрагментации*. Протокол BOOTP также рассчитан на прием нескольких ответов. При этом принимается и обрабатывается первый из поступивших ответов.

Чтобы предотвратить потерю дейтаграмм, в протоколе BOOTP используется традиционный прием — *повторная передача* пакета по истечении *тайм-аута*. Посылая запрос, клиент запускает таймер. Если после истечения времени таймера он не получит ни одного ответа, клиент должен повторно переслать запрос. Очевидно, что сбой электропитания вызовет одновременную перезагрузку машин в сети, а это повлечет за собой кратковременную перегрузку запросами BOOTP-серверов. Если все клиенты используют одинаковое значение тайм-аута повторной передачи, большинство из них попытается выполнить повторную передачу запросов одновременно. Чтобы избежать возникновения коллизий, в спецификации протокола BOOTP рекомендуется использовать случайно выбранное время задержки. Рекомендуется также выбирать начальное значение таймера случайным образом в интервале от 0 до 4 секунд, а после каждой повторной передачи удваивать его значение. После того как величина тайм-аута достигнет большого значения (60 секунд), клиент больше не удваивает значение таймера, а снова выбирает его случайным образом. Таким образом,

*чтобы предотвратить возникновение дополнительного трафика, создаваемого при использовании протокола BOOTP, в перегруженной сети, значение тайм-аута удваивается после каждой повторной передачи. Для предотвращения одновременной пересылки пакетов начальное значение тайм-аута должно выбираться случайным образом.*

## 23.5. Формат BOOTP-сообщения

Чтобы максимально упростить реализацию, BOOTP-сообщения имеют поля фиксированной длины, а ответные сообщения создаются в том же формате, что и запросы. Хотя было сказано, что клиенты и серверы являются программами, в протоколе BOOTP эти понятия используются в широком смысле. Отсылающая BOOTP-запрос машина считается *клиентом*, а машина, отсылающая ответ, — *сервером*. На рис. 23.1 показан формат BOOTP-сообщения.

Поле *Тип пакета* определяет, является ли сообщение запросом (1) или ответом (2). Как и в протоколе ARP, поля *Тип оборудования* и *Длина физ. адреса* определяют тип платы сетевого интерфейса и длину аппаратного адреса (например, для сети Ethernet указывается тип 1 и длина адреса 6)<sup>2</sup>. При отправке запроса клиент помещает в поле *числа переходов* значение 0. Если сервер

<sup>2</sup> Значения для поля типа оборудования можно найти в последней редакции документа RFC, посвященного назначению уникальных номеров.

протокола BOOTP получает запрос и решает его переслать другой машине (например, разрешить процесс начальной загрузки на нескольких маршрутизаторах), он увеличивает значение в поле *числа переходов* на единицу. В поле *идентификатора транзакции* находится целое число, используемое бездисковой машиной для идентификации ответов на посланные ею запросы. В поле *времени после начала загрузки клиента* указывается количество секунд, прошедших после начала загрузки клиента.

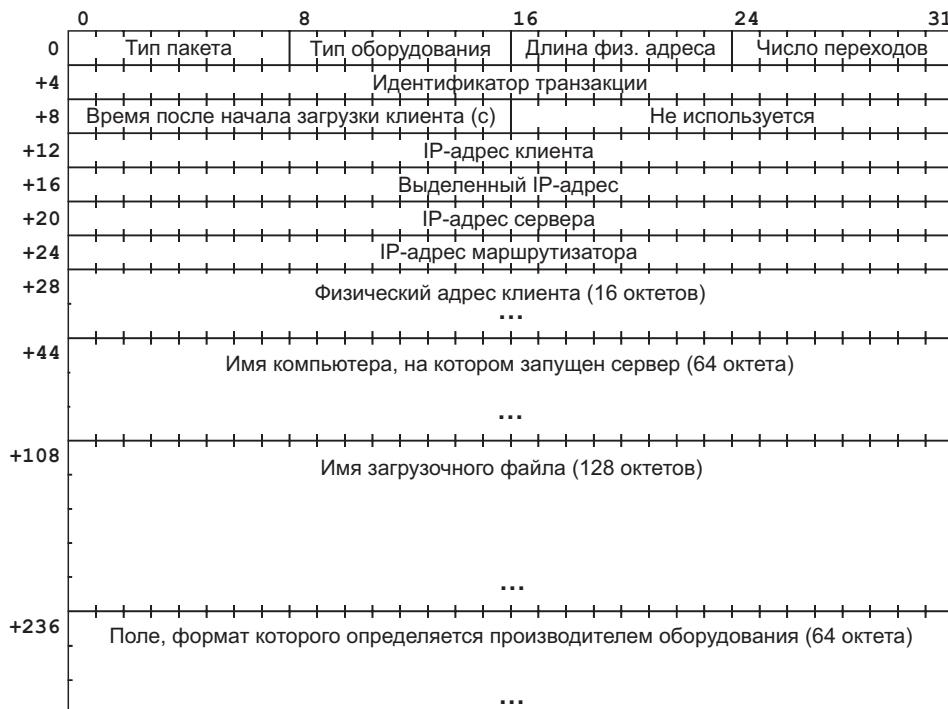


Рис. 23.1. Формат BOOTP-сообщения. Чтобы программы реализации этого протокола смогли поместиться в ПЗУ небольшого размера, все поля сообщения имеют фиксированную длину

В поле *IP-адреса клиента*, а также во всех следующих за ним полях находится самая важная информация. Чтобы обеспечить максимальную гибкость, клиенты заполняют эти поля всей имеющейся у них информацией, а в остальных полях оставляют нулевое значение. Например, если клиенту известно имя или адрес сервера, от которого он хочет получить информацию, он может заполнить поле *IP-адреса сервера* или поле *имени компьютера, на котором запущен сервер*. Если в этих полях установлены отличные от нуля значения, на запрос ответит только сервер с соответствующим именем/адресом. Если же значения в этих полях равны нулю, на запрос ответит любой получивший его сервер.

Протокол BOOTP может использоваться для инициализации клиента, которому уже известен его IP-адрес (например, для получения имени *загрузочного файла*). Клиент, которому известен свой IP-адрес, помещает его в поле *IP-адреса клиента*; в противном случае он должен поместить туда нулевое значение. Если в запросе IP-адрес клиента равен нулю, сервер возвращает IP-адрес клиента, поместив его в поле *выделенного IP-адреса*.

## **23.6. Двухэтапный процесс начальной загрузки**

В протоколе BOOTP используется двухэтапный процесс начальной загрузки. При этом клиенту не передается образ содержимого памяти для выполнения, а только информация, необходимая для получения этого образа. Поэтому, чтобы получить образ содержимого памяти, клиент должен воспользоваться другим протоколом (например, описанным в главе 26, “Приложения: передача файлов и удаленный доступ к ним (FTP, TFTP, NFS)”, протоколом TFTP). Хотя двухэтапный процесс может показаться излишним, он позволяет провести четкую границу между процессом конфигурации устройства и хранением данных. Сервер протокола BOOTP не обязательно должен запускаться на той же машине, где хранятся образы содержимого памяти. Чаще всего сервер протокола BOOTP получает информацию из простой базы данных, в которой находится информация только об именах файлов, содержащих образ содержимого оперативной памяти.

Разделение конфигурационных данных и программ начальной загрузки является весьма важным моментом, поскольку это позволяет администраторам сконфигурировать большое число машин так, чтобы они работали одинаково и независимо друг от друга. Для этого в ответное сообщение протокола BOOTP введено специальное поле *имени загрузочного файла*. Предположим, что в локальной сети существует несколько рабочих станций с разным типом аппаратного обеспечения, при загрузке которых пользователи могут запустить либо систему UNIX, либо одну из локальных операционных систем. Поскольку рабочие станции имеют разную аппаратную архитектуру, для них нельзя создать один загрузочный файл, содержащий образ памяти. Для решения этой проблемы используется поле *имени загрузочного файла* BOOTP-запроса. При отправке запроса рабочая станция может поместить в него некое обобщенное имя, например “*unix*”, что означает следующее: “Я хочу загрузить операционную систему UNIX на этой машине”. Получив такой запрос, сервер протокола BOOTP обращается к своей конфигурационной базе данных и преобразует обобщенное имя в имя загрузочного файла, содержащего образ памяти системы UNIX, который соответствует типу аппаратного обеспечению клиента. При формировании ответа на запрос сервер помещает в поле *имени загрузочного файла* полностью определенное имя этого файла. Конечно, конфигурационная база данных также позволяет автоматизировать процесс начальной загрузки рабочей станции. Если клиент помещает в поле *имени загрузочного файла* нули, то сервер протокола BOOTP подбирает соответствующий этой машине файл образа оперативной памяти. Преимущество метода автоматической начальной загрузки заключается в том, что она позволяет пользователям определять обобщенные имена файлов, которые работают на любой машине. Таким образом, пользователям не нужно запоминать конкретные имена файлов в зависимости от структуры аппаратного обеспечения.

## **23.7. Поле, определяемое производителем**

В поле, формат которого определяется производителем оборудования, помещается дополнительная информация, передаваемая сервером клиенту. Хотя структура этого поля довольно замысловатая, в ней легко разобраться. Первые четыре октета этого поля называются *магическим сигналом (magic cookie)* и определяют формат остальных элементов. В описанном в этом разделе стандартном формате используется значение магического сигнала 99.130.83.99 (в точечном десятичном представлении). После магического сигнала следует список элементов, каждый из которых содержит однооктетное поле *типа*, необязательное однооктетное, поле *длины* и состоящее из нескольких октетов поле

значения<sup>3</sup>. В стандарте определено несколько типов, которые имеют заранее определенные, фиксированные значения длины (табл. 23.1). Поле длины должно обязательно присутствовать для типов 1 и 2, и отсутствовать для типов 0 и 255.

**Таблица 23.1. Информационные элементы поля, определяемого производителем**

Тип элемента	Код элемента	Длина значения	Содержимое
Заполнение	0	—	Ноль — используется только как заполнение для выравнивания длины элемента на заданную границу
Маска подсети	1	4	Маска подсети для локальной сети
Текущее время	2	4	Текущее всемирное время
Конец	255	—	Конец списка элементов

Хотя компьютер может получить информацию о маске подсети с помощью ICMP-запроса, в действующем стандарте рекомендуется, чтобы серверы протокола BOOTP включали эту информацию в каждое ответное сообщение. В результате этого отпадает необходимость в дополнительных ICMP-сообщениях для определения маски подсети.

Во всех дополнительных элементах, помещаемых в поле, определяемое производителем оборудования, также используется TLV-кодирование. Каждый элемент состоит из октета *типа*, октета *длины* и *значения*. В табл. 23.3 перечислены возможные варианты, причем некоторые из них имеют переменную длину.

**Таблица 23.2. Типы и содержимое элементов области определяемой производителем ответного BOOTP-сообщения**

Тип элемента	Код элемента	Длина значения	Содержимое
Маршрутизаторы	3	N	IP-адреса маршрутизаторов, количество которых равно N/4
Серверы времени	4	N	IP-адреса серверов времени, количество которых равно N/4
Серверы IEN116	5	N	IP-адреса серверов IEN116, количество которых равно N/4
Серверы доменных имен	6	N	IP-адреса серверов системы DNS, количество которых равно N/4
Серверы системного журнала	7	N	IP-адреса серверов системного журнала, количество которых равно N/4
Серверы, содержащие цитаты дня (quote servers)	8	N	IP-адреса серверов цитат, количество которых равно N/4
Серверы построчной печати	9	N	IP-адреса серверов построчной печати (lpr), количество которых равно N/4
Серверы потоковой печати	10	N	IP-адреса серверов потоковой печати (lprint servers), количество которых равно N/4

<sup>3</sup> Этот формат представляет собой пример TLV-кодирования. Термин *TLV* — сокращенное название *Type-Length-Value* (*Тип-Длина-Значение*).

*Окончание табл. 23.2*

<i>Тип элемента</i>	<i>Код элемента</i>	<i>Длина значения</i>	<i>Содержимое</i>
Серверы RLP	11	N	IP-адреса серверов RLP, количество которых равно N/4
Имя узла сети	12	N	N байтов имени клиентского компьютера
Размер файла начальной загрузки	13	2	Целочисленный размер файла начальной загрузки, заданный двумя октетами
Зарезервированное поле	128–254	—	Зарезервировано для особых случаев использования в конкретном сетевом центре

### 23.8. Необходимость динамической конфигурации

Протокол BOOTP был разработан для использования в относительно статическом окружении, где каждый узел сети имеет постоянное сетевое подключение. Для каждого узла сети администратор создает отдельный файл конфигурации, в который помещает набор параметров протокола BOOTP. В этот файл не нужно часто вносить изменения, поскольку параметры конфигурации обычно остаются неизменными на протяжении длительного периода времени (как правило, на протяжении нескольких недель).

После появления технологий беспроводной сетевой связи и портативных компьютеров появилась возможность легко перемещать узлы сети из одного места в другое. Протокол BOOTP слабо приспособлен для работы в подобных случаях, поскольку конфигурационную информацию нельзя быстро изменить. В протоколе BOOTP предусмотрено только статическое сопоставление идентификатора узла сети с его набором параметров. Более того, администратор должен ввести набор параметров для каждого узла сети, а затем сохранить эту информацию в файле конфигурации BOOTP-сервера. В протоколе BOOTP не предусмотрен способ, с помощью которого можно динамически менять значения параметров отдельных узлов сети. В частности, администратор должен назначить каждому узлу сети IP-адрес и должен сконфигурировать сервер таким образом, чтобы он преобразовывал идентификатор узла сети в IP-адрес.

Назначение статических параметров эффективно в том случае, если компьютеры все время находятся в одном месте и администратор располагает достаточным количеством IP-адресов, чтобы выделить каждому компьютеру уникальный IP-адрес. Однако в тех случаях, когда компьютеры часто перемещаются или если количество физических компьютеров превышает количество доступных IP-адресов узлов сети, статическое присвоение влечет за собой существенные накладные расходы.

Чтобы понять, почему количество компьютеров может превысить количество доступных IP-адресов, рассмотрим локальную сеть учебного класса, которой присвоен IP-адрес с маской /24. Это позволяет подключить к ней до 254 компьютеров. Поскольку количество рабочих мест в классе рассчитано только на 30 студентов, расписание составлено так, чтобы лабораторные занятия проводились десять раз в неделю в разное время. Это позволяет задействовать в учебном процессе до 300 студентов. Теперь предположим, что каждый студент приносит с собой персональный компьютер (ноутбук), который он использует во время лабораторных занятий. В любой момент времени в сети работает не более 30 компьютеров. Однако, поскольку к сети может подключиться максимум 254 узла, администратор не может присвоить уникальный адрес каждому компьютеру. Таким

образом, хотя количество физических подключений к сети ограничено, реальная потребность в них иногда оказывается гораздо большей. Понятно, что систему нельзя назвать совершенной, если для подключения к сети очередного компьютера, администратор должен изменить файл конфигурации BOOTP-сервера. Здесь нужны средства автоматизации.

### 23.9. Динамическая конфигурация узлов сети

Чтобы обеспечить автоматическое присвоение адресов, группой разработчиков IETF был создан новый протокол *динамической конфигурации узла сети* (*Dynamic Host Configuration Protocol*, или *DHCP*). Он расширил возможности протокола BOOTP в двух направлениях. Во-первых, протокол DHCP позволяет компьютеру получить всю необходимую ему информацию о конфигурации в одном сообщении. Например, помимо IP-адреса, в сообщение протокола DHCP может входить маска подсети. Во-вторых, протокол DHCP позволяет быстро динамически назначить компьютеру IP-адрес. Чтобы использовать механизм динамического распределения адресов протокола DHCP, администратор должен настроить DHCP-сервер и выделить ему набор IP-адресов. Каждый раз при подключении к сети нового компьютера, он посыпает DHCP-серверу запрос на получение адреса. Сервер выбирает один из адресов, определенных администратором, и назначает этот адрес компьютеру.

Для полноты изложения необходимо отметить, что в протоколе DHCP можно использовать три типа присвоения адресов. Причем реакция сервера на поступление запросов со стороны определенных сетей или узлов сети определяется администратором. Подобно протоколу BOOTP, в протоколе DHCP предусмотрена *ручная конфигурация* узлов сети, благодаря чему администратор может назначить определенный адрес заданному компьютеру. В протоколе DHCP также предусмотрена *автоматическая конфигурация* узла сети, т.е. DHCP-сервер может присвоить постоянный адрес узлу сети при первом подключении. И наконец, в протоколе DHCP предусмотрена полностью *динамическая конфигурация* узла сети — сервер выделяет адрес компьютеру на ограниченный период времени.

Подобно протоколу BOOTP, для определения своих дальнейших действий при поступлении запроса DHCP-сервер использует набор уникальных параметров клиента, которые его однозначно определяют. При установке связи с DHCP-сервером клиент отсылает свой уникальный идентификатор, который обычно представляет собой физический адрес сетевой платы. Сервер использует идентификатор клиента и сеть, к которой он подключен, для определения способа присвоения ему IP-адреса. Таким образом, администратор имеет возможность полностью управлять процессом присвоения адресов. Сервер можно сконфигурировать так, чтобы он статически назначал адреса определенным компьютерам (как при использовании протокола BOOTP), позволяя в то же время другим компьютерам динамически получать постоянные или временные адреса.

### 23.10. Динамическое назначение IP-адресов

Динамическое назначение адресов, присущее протоколу DHCP, отличает его от протокола BOOTP. В отличии от статического назначения адресов, используемого в протоколе BOOTP, динамическое назначение адресов не является однозначно определяемым. Поэтому серверу не нужно знать заранее “личность” клиента. В частности, DHCP-сервер можно сконфигурировать таким образом, чтобы произвольный компьютер мог получить у него IP-адрес и начать взаимодействие по сети. Таким образом, протокол DHCP позволяет создавать системы, которые конфигурируются автоматически. После подключения к сети компьютер использует

протокол DHCP для получения IP-адреса, а затем настраивает свои программы поддержки протокола TCP/IP для работы с этим адресом. Естественно, процесс автоматической конфигурации должен находиться под контролем системного администратора. Он вправе запретить или разрешить его на каждом конкретном сервере DHCP. Подведем итог всему сказанному выше.

*Поскольку протокол DHCP позволяет узлу сети получить от сервера все необходимые для взаимодействия параметры без вмешательства администратора, он является автоматически конфигурируемым. Режим автоконфигурации находится под контролем системного администратора.*

Для использования режима автоконфигурации, сетевой администратор должен выделить DHCP-серверу блок IP-адресов и определить набор правил функционирования сервера. Для получения своего адреса DHCP-клиент должен обмениваться сообщениями с сервером. При этом сервер выделяет адрес клиенту, а клиент подтверждает прием этого адреса. Как только клиент принял адрес, он может начать его использовать для рассылки пакетов по сети.

В отличие от статического назначения адресов, когда каждый IP-адрес выделяется определенному узлу сети постоянно, динамическое назначение адресов является временным. Говорят, что DHCP-сервер *выделяет (арендует)* адрес клиенту на определенный период времени. При назначении адреса сервер определяет время его использования. На протяжении этого времени сервер не может выделить этот адрес другому клиенту. Тем не менее, когда время использования адреса подходит к концу, клиент должен его продлить либо прекратить использовать этот адрес.

Как долго должно длиться время использования адреса, выделенного DHCP-сервером? Оптимальное значение времени зависит от сети и потребностей отдельного узла. Например, чтобы гарантировать быстрое повторное использование адресов, для персональных компьютеров студентов, подключенных к сети учебного класса, можно установить короткий промежуток времени использования адреса (например, продолжительностью в один час). В корпоративной сети можно установить время использования адреса продолжительностью в один день или в одну неделю. Поэтому, чтобы учесть все возможные случаи, в протоколе DHCP не определена фиксированная величина продолжительности использования адреса. Вместо этого, в протоколе предусмотрены средства для запроса клиентом определенного времени использования адреса. В ответном сообщении сервер информирует клиента о предоставленном времени. Таким образом, администратор может решить, на какой период времени каждый сервер должен назначать адрес клиенту. В самых крайних случаях в протоколе DHCP предусмотрено выделение адреса на *бесконечно долгий* срок. Благодаря этому выделенный IP-адрес может использоваться клиентом неограниченное время, подобно используемым в протоколе BOOTP постоянным адресам.

## 23.11. Получение нескольких адресов

*Многоадресным (multi-homed)* называется компьютер, подключенный к нескольким сетям. При загрузке такого компьютера ему может понадобиться конфигурационная информация для каждого из своих сетевых интерфейсов. Подобно BOOTP-сообщению, в DHCP-сообщении может содержаться информация только об одном интерфейсе. Компьютер с несколькими интерфейсами должен самостоятельно получить конфигурационную информацию для каждого из них. Поэтому, хотя здесь протокол DHCP описан с точки зрения компьютера, которому необходим только один адрес, читатель не должен забывать, что интерфейсы многоадресного компьютера могут относиться к разным сетевым протоколам.

Чтобы компьютер мог связаться с сервером, расположенным в другой локальной сети, в протоколах BOOTP и DHCP используется понятие *агента пересылки* (*relay agent*). Получив от клиента широковещательный запрос, агент пересыпает его серверу, а затем возвращает поступивший ответ узлу сети. Использование агентов пересылки часто усложняет процесс настройки многоадресного узла, поскольку сервер может получить несколько запросов от одного и того же компьютера. Следует отметить, что в обоих протоколах BOOTP и DHCP используется одно и то же понятие — *идентификатор клиента*. В случае многоадресного клиента оно должно относиться не к компьютеру в целом, а к его определенному сетевому интерфейсу, например содержать его уникальный физический адрес. Таким образом, сервер всегда сможет отличить запросы, поступающие от многоадресного узла сети, даже когда он получает их через агента пересылки.

### 23.12. Состояние получения адреса

При использовании протокола DHCP для получения IP-адреса клиент может находиться в одном из шести состояний. На приведенной на рис. 23.2 диаграмме перехода состояний показаны события и сообщения, которые заставляют клиента изменить свое состояние.

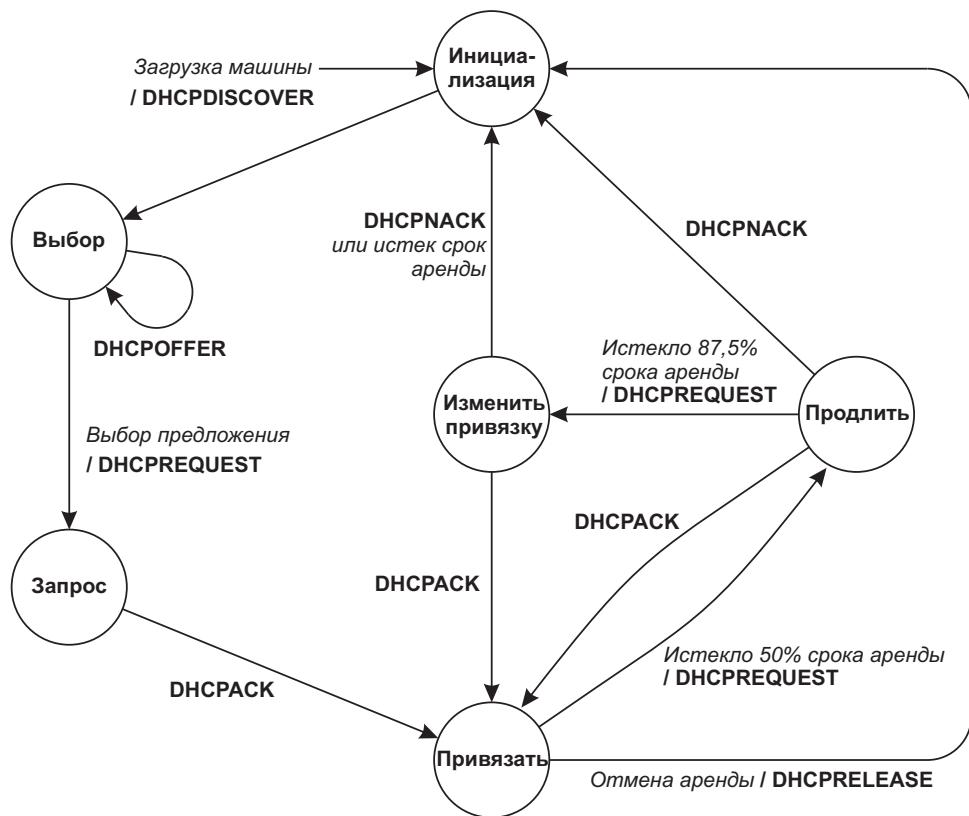


Рис. 23.2. Шесть основных состояний DHCP-клиента и диаграмма перехода состояний между ними. Курсивом на рисунке обозначены входящие сообщения или события, вызывающие изменение состояния клиента, после которого через косую черту указаны типы сообщений, посылаемых клиентом

При начальной загрузке клиент переходит в состояние *инициализации* (*INITIALIZE*). Прежде чем запросить IP-адрес, клиент должен связаться со всеми DHCP-серверами локальной сети. Для этого клиент отсылает в широковещательном режиме передачи сообщение *DHCPDISCOVER* и переходит в состояние *выбора* (*SELECT*). Поскольку протокол DHCP представляет собой расширение протокола BOOTP, клиент посыпает сообщение *DHCPDISCOVER* в UDP-дейтаграмме. При этом в качестве порта получателя указывается номер порта протокола BOOTP (т.е. порт 67). Это сообщение получают все DHCP-серверы локальной сети. Ответ (сообщение *DHCPOFFER*) посыпают только те серверы, которым разрешено взаимодействовать с данным клиентом. В результате клиент может получить сразу несколько ответов либо не получить ни одного.

Находясь в состоянии выбора, клиент анализирует поступившие от DHCP-серверов ответы-предложения *DHCPOFFER*. В каждом предложении содержится конфигурационная информация для клиента вместе с IP-адресом, выделяемым ему сервером. Клиент должен выбрать один из ответов (например, первый поступивший) и договориться с сервером о выделении адреса. Для этого клиент отсылает серверу сообщение *DHCPREQUEST* и входит в состояние *запроса* (*REQUEST*). Чтобы подтвердить получение запроса и выделить клиенту адрес, сервер отвечает на запрос, отсылая сообщение *DHCPSACK*. Получив подтверждение, клиент переходит в состояние *привязки* (*BIND*), после чего начинает использовать выделенный ему адрес. Подведем итог всему сказанному выше.

*При использовании протокола DHCP узел сети становится клиентом и отсылает в широковещательном режиме сообщение всем DHCP-серверам локальной сети. Затем узел сети собирает поступившие от серверов предложения, выбирает одно из них и отправляет подтверждение серверу.*

### 23.13. Досрочное прекращение использования адреса

Состояние привязки можно рассматривать как обычное рабочее состояние клиента. Клиент, как правило, продолжает находиться в этом состоянии до тех пор, пока он использует выделенный ему сервером IP-адрес. Если клиент имеет вспомогательное запоминающее устройство (например, локальный диск), он может сохранить присвоенный ему IP-адрес в файл, а затем при перезагрузке заново запросить тот же самый адрес. Однако в некоторых случаях находящийся в состоянии привязки клиент может обнаружить, что ему больше не нужен запрошенный у сервера IP-адрес. Предположим, пользователь подключил портативный компьютер к сети для работы с электронной почтой с помощью протоколов семейства TCP/IP. При этом для получения IP-адреса используется протокол DHCP. Пользователь заранее может не знать, сколько времени ему понадобится на чтение почты, поэтому в данном случае время использования адреса назначается сервером. В протоколе DHCP установлено минимальное время использования адреса — один час. Если, подключившись к сети и получив IP-адрес пользователь обнаружит, что в его почтовом ящике нет непрочитанных сообщений, он может выключить портативный компьютер и уехать в другое место. При этом полученный IP-адрес должен быть возвращен серверу.

Для досрочного прекращения использования адреса в протоколе DHCP предусмотрены специальные средства. Они применяются в тех случаях, когда ни клиент, ни сервер не могут определить продолжительность использования адреса в момент его выделения. С помощью этих средств сервер может установить приемлемое время использования адреса. Механизм досрочного прекращения использования адреса особенно важен, если количество доступных на сервере IP-адресов намного меньше, чем подключенных к сети компьютеров. Поэтому если

клиент сразу же уведомит сервер о том, что выделенный ему IP-адрес больше не нужен, сервер сможет назначить этот адрес другому клиенту.

Для досрочного прекращения использования адреса клиент отсылает серверу сообщение `DHCPRELEASE`. Отправка этого сообщения (т.е. освобождение адреса) — последнее действие, которое может выполнить клиент в сети с помощью назначенного ему IP-адреса. Таким образом, переслав серверу сообщение об освобождении адреса, клиент должен сразу же прекратить отсылку дейтаграмм, в которых используется данный адрес. Если рассматривать это явление с точки зрения изображенной на рис. 23.2 диаграммы перехода состояний, то после отсылки сообщения `DHCPRELEASE` узел сети выходит из состояния привязки и, прежде чем использовать протокол IP, он должен снова перейти в состояние инициализации.

## 23.14. Состояние ожидания продления срока использования адреса

Выше уже было сказано, что, получив адрес, DHCP-клиент переходит в состояние привязки. При этом он запускает три таймера: они управляют продлением срока использования адреса, изменением привязки и прекращением использования адреса. Назначая адрес клиенту, DHCP-сервер может определить явные значения для этих таймеров. Если сервер не определяет значения для таймеров, клиент использует значения, принятые по умолчанию. Значение по умолчанию для первого таймера составляет одну вторую от общего времени использования адреса. Когда время на первом таймере истекает, клиент должен попытаться продлить срок использования адреса. При этом он отсылает сообщение `DHCPREQUEST` серверу, которым был выделен данный адрес. Затем клиент переходит в состояние *продления* и ожидает ответа на запрос. В сообщении `DHCPREQUEST` указывается IP-адрес, используемый в текущий момент клиентом. С помощью этого сообщения клиент просит сервер продлить время использования указанного адреса. Как и при первоначальном запросе адреса, клиент может потребовать продлить время его использования на определенный период времени, однако решение о продлении, в конечном счете, принимает сервер. Сервер может ответить на посланный клиентом запрос двояко: приказать клиенту прекратить использование адреса либо одобрить дальнейшее его использование. В случае одобрения сервер отсылает сообщение `DHCPOFFER`, что заставляет клиента вернуться в состояние привязки и продолжить использование адреса. В сообщении `DHCPOFFER` также могут находиться новые значения для таймеров клиента. Если сервер не одобряет дальнейшее использование адреса, он отсылает сообщение `DCHPNAK` (отрицательное подтверждение), что заставляет клиента тотчас же прекратить использование адреса и возвратиться в состояние инициализации.

После отправки сообщения `DHCPREQUEST` с запросом на продление срока использования адреса, клиент продолжает находиться в прежнем состоянии (ожидания продления) до поступления ответа сервера. Если ответ не поступает, это означает, что сервер, выдавший данный адрес клиенту, находится либо в нерабочем состоянии, либо с ним прервалась связь. Чтобы обработать эту ситуацию, в протоколе DHCP используется второй таймер, который запускается при переходе клиента в состояние привязки. Значение второго таймера составляет 87,5% от установленного времени использования адреса. После срабатывания второго таймера, клиент переходит из состояния ожидания продления в состояние *изменения привязки* (*REBIND*). При этом клиент предполагает, что старый DHCP-сервер более недоступен, поэтому он начинает отсылать сообщение `DHCPREQUEST` в широковещательном режиме всем подключенным к локальной

**Печатай файл этой страницы отдельно**

# 24

## *Система доменных имен (DNS)*

### **24.1. Введение**

В предыдущих главах были описаны протоколы, в которых для идентификации машин используются 32-битовые целые числа, называемые адресами протокола IP, или IP-адресами. Они обеспечивают удобную и компактную форму идентификации отправителя и получателя пакетов, пересылаемых по объединенной сети. Тем не менее пользователи предпочитают присваивать машинам удобопроизносимые и легко запоминающиеся имена.

В этой главе рассматривается система назначения значащих высокоуровневых имен для большой совокупности машин и обсуждается механизм преобразования между высокоуровневыми именами и IP-адресами. Мы рассмотрим — преобразование высокоуровневых имен в IP-адреса и IP-адресов — в высокоуровневые имена машин. Назначения имен представляет интерес по двум причинам. Во-первых, она используется для идентификации машин по имени во всей глобальной сети Internet. Во-вторых, для преобразования имен в адреса используется набор распределенных серверов, которые находятся на значительном расстоянии друг от друга. Практическая реализация механизма преобразования имен представляет собой пример крупномасштабного взаимодействия по принципу клиент/сервер, который был описан в главе 21, “Модель взаимодействия клиент/сервер”.

### **24.2. Идея присвоения имен машинам**

Пользователи, работавшие с первыми компьютерными системами, были вынуждены мириться с тем, что для таких объектов, как системные таблицы и периферийные устройства, назначались цифровые адреса. Прогресс в деле обработки данных был достигнут благодаря системам, работающим в режиме разделения времени. Пользователи получили возможность назначать значащие символические имена как для физических объектов, (например, периферийных устройств), так и для абстрактных объектов (например, файлов). Подобная система идентификации сохранилась и при создании объединенной сети вычислительных машин. В первых системах взаимодействие между машинами обеспечивалось благодаря двухточечным соединениям, проложенным между компьютерами, а для идентификации машин использовался низкоуровневый физический адрес. Для обеспечения межсетевого обмена была придумана универсальная система адресации. В ней с помощью специальных программ выполнялось преобразование универсальных адресов в низкоуровневые физические адреса сетевых адаптеров. Поскольку в большинстве вычислительных сред содержится большое количество машин, для их идентификации пользователям требуются значащие символические имена.

Первые системы именования машин были ориентированы на работу в небольшом сетевом окружении. Имена машин в сетевых центрах с небольшим количеством машин обычно выбирались исходя из их назначения. Например, машины часто имели имена типа *research* (исследовательская лаборатория), *production* (производство), *accounting* (бухгалтерия) и *development* (разработчики). Именно такие имена предпочитают употреблять пользователи вместо громоздких и трудно запоминающихся физических адресов.

Хотя на интуитивном уровне различие между *адресом* и *именем* понятно, оно несколько искусственно, поскольку любое *имя* — это всего лишь идентификатор, который состоит из последовательности буквенно-цифровых символов, выбранных из алфавита ограниченного размера. Использование имен оправдано только в том случае, если система может эффективно соотнести их с объектом, который они обозначают. Поэтому IP-адрес является не чем иным, как *низкоуровневым именем*, а имя, присваиваемое пользователем своему компьютеру — *высокоуровневым*.

Выбор формы высокоуровневого имени очень важен поскольку именно она определяет способ преобразования этого имени в низкоуровневое (или способ привязки его к объекту), а также степень ответственности за выполнение этой операции. Выбор системы именования для небольшой сети, в которой взаимодействует несколько машин, не представляет особого труда, поскольку здесь подойдет любая форма. Однако в глобальной сети Internet, связывающей приблизительно сто миллионов машин, выбор символических имен становится непростой задачей. Например, когда в 1980 году главный ведомственный компьютер факультета информатики университета Пердью (Purdue University) был подсоединен к глобальной сети Internet, для его идентификации было выбрано имя *purdue*. При этом список зарезервированных имен едва превышал десяток. Однако уже к середине 1986 года список официально зарегистрированных имен узлов сети Internet содержал 3100 официально зарегистрированных имен и 6500 официальных псевдонимов<sup>1</sup>. Несмотря на то что этот список быстро пополнялся, в середине 80-х годов XX века в большинстве сетевых центров имелись машины (например, персональные компьютеры пользователей), которым не было назначено официальное имя.

### 24.3. Одноуровневая система имен

Одной из первых систем именования машин, использовавшейся долгое время в Internet, была *одноуровневая система имен* (*flat namespace*), т.е. каждое имя состояло из последовательности символов без какой либо структуризации. В первоначальной системе пространством имен управлял *сетевой информационный центр* (*Network Information Center*, или *NIC*). Он определял допустимость использования нового имени, то есть запрещалось использовать непристойные имена, а также новые имена, которые находились в противоречии с уже существующими именами.

Основным преимуществом одноуровневой системы именования были короткие имена, которыми очень удобно пользоваться. Главный же недостаток такой системы заключался в том, что ее нельзя было применить для больших наборов машин как в силу технических, так и административных причин. Во-первых, поскольку все имена были одноуровневыми и состояли из одного набора символов, вероятность конфликта возрастала с увеличением количества зарегистрированных имен. Во-вторых, поскольку право добавления новых имен должно принадлежать одному сетевому административному органу, с увеличением количе-

<sup>1</sup> К 1990 году более 137 000 сетевых узлов Internet имели имена, а к 2000 году их число превысило 60 миллионов.

ства зарегистрированных узлов сети возрастает административная нагрузка на этот орган. Чтобы понять серьезность проблемы, представьте себе быстро растущую объединенную сеть, к которой подключено несколько тысяч сетей предприятий. В каждой из них могут находиться сотни или тысячи персональных компьютеров и рабочих станций. Каждый раз, когда кто-либо приобретает и подключает к сети новый персональный компьютер, имя этого компьютера должно быть утверждено центральным административным органом. В-третьих, поскольку привязка имени к адресу часто меняется, стоимость поддержки достоверных копий полного списка имен на каждом узле сети высока и возрастает с увеличением количества узлов. С другой стороны, если база данных имен будет находиться на одном сервере, поток данных к этому серверу будет расти с увеличением количества узлов объединенной сети.

## 24.4. Иерархическая система имен

Как изменить систему назначения имен так, чтобы она легкоправлялась с большим, быстро расширяющимся пространством имен и при этом для ее управления не требовалось бы вмешательства центрального административного органа Internet? Решение этой проблемы состоит в децентрализации механизма назначения имен, когда полномочия по управлению частью пространства имен делегируются другим административным органам. При этом процесс преобразования имен в адреса и наоборот, должен выполняться распределенным набором серверов. Подобная система используется в объединенной сети на основе протокола TCP/IP. Перед изучением деталей этой системы, рассмотрим основные идеи, лежащие в основе ее работы.

Пространство имен должно быть разделено так, чтобы можно было выполнять эффективное преобразование имен в адреса и гарантировать автономность процесса назначения имен. Если оптимизацию проводить только с точки зрения эффективности процесса преобразования имен, то в конечном счете мы придем к выводу, что нужно сохранить старую одноуровневую систему именования. При этом для уменьшения количества запросов на преобразование, поступающих к одному серверу, все пространство имен следует разделить между несколькими физическими машинами. Если же проводить оптимизацию только с точки зрения облегчения административного управления, то это может привести к принятию решений, которые облегчают процесс делегирования полномочий, но делают процесс преобразования имен дорогим или сложным.

Чтобы понять, как следует разделить пространство имен, рассмотрим внутреннюю структуру больших организаций. Стоящий на вершине власти президент или генеральный директор имеет полную власть. Поскольку он не может следить за всем, что происходит внутри организации, ее разделяют на подразделения, которые возглавляют отдельные руководители. Обычно президент предоставляет каждому руководителю подразделения ограниченную автономию. Другими словами, руководитель каждого подразделения может нанимать или увольнять служащих, изменять штатное расписание и делегировать полномочия другим руководителям, не получая прямой санкции от президента.

Иерархическая структура большой организации, помимо облегчения делегирования полномочий, позволяет ввести также автономное управление. Например, когда работнику офиса нужна информация о номере телефона нового служащего, то он сначала спрашивает об этом у ближайших сотрудников, которые могут входить в контакт со служащими из других подразделений. Дело в том, что, хотя полномочие всегда передается через корпоративную иерархию сверху вниз, информация может передаваться через эту иерархию между подразделениями.

## 24.5. Делегирование полномочий в системе имен

Принципы управления иерархической системой назначения имен во многом напоминают принципы управления большой организацией. Пространство имен *разделяется* на зоны на высшем уровне, а полномочия для назначения имен в отдельных зонах предоставляют специальным агентам. Например, можно разделить пространство имен, взяв за основу *имя сетевого центра*, и делегировать каждому сетевому центру полномочия по поддержанию имен в пределах своей зоны. Деление на зоны и делегирование полномочий административным органам каждой зоны происходит на самом верхнем уровне иерархии. Дальнейшие изменения пространства имен в пределах зоны могут выполняться без участия административных органов верхнего уровня иерархии.

Синтаксис имен в иерархической системе зачастую отражает принцип делегирования полномочий того административного органа, который зарегистрировал эти имена. В качестве примера рассмотрим пространство имен, в котором имена имеют вид:

```
local.site
```

Здесь *site* — имя сетевого центра, зарегистрированное центральным административным органом, *local* — часть имени, назначаемая локальным сетевым центром, и точка (“.”) — разделитель, отделяющий две части имени. Когда центральный административный орган регистрирует новый сетевой центр *X*, он добавляет его имя в список зарегистрированных сетевых центров и делегирует сетевому центру *X* полномочия по управлению всеми именами, которые заканчиваются на “.*X*”.

## 24.6. Дальнейшее разделение полномочий

В иерархической системе пространство имен может подразделяться на более мелкие зоны на каждом уровне иерархии. Выше был описан пример разделения пространства имен между сетевыми центрами верхнего уровня. Однако в сам сетевой центр верхнего уровня может входить несколько административных групп (*group*). Администрация этого сетевого центра имеет полномочия разделить его пространство имен между мелкими группами, чтобы обеспечить легкость и удобство управления.

Синтаксически такое разделение пространства имен приводит к добавлению еще одного имени, отделенного точкой, в первоначальную иерархическую структуру. Например, разделение пространства имен, выделенного сетевому центру, между мелкими административными группами приведет к тому, что имена в пределах этих групп будут иметь следующий вид:

```
local.group.site
```

Поскольку центральный административный орган делегировал полномочия по управлению частью пространства имен сетевому центру верхнего уровня, центр не должен согласовывать имена мелких административных групп с другими сетевыми центрами верхнего уровня. Например, университетский сетевой центр может выбрать следующие имена для групп: *engineering* (инженерная группа), *science* (научная группа) и *arts* (группа художников). Корпоративный сетевой центр скорее всего назначит группам следующие имена: *production* (производственный отдел), *accounting* (бухгалтерия) и *personnel* (отдел кадров).

Еще одним примером иерархической системы именования является обычная телефонная сеть. Десять цифр телефонного номера разбиты на трехзначный *междугородный телефонный код*, трехзначный *номер АТС* (автоматической теле-

*фонной станции*) и четырехзначный номер абонента в пределах АТС. Каждая АТС имеет полномочия для назначения номеров абонентов в пределах своей части пространства имен (точнее, номеров). Хотя вполне возможно, сгруппировав абонентов разных АТС, назначить им один номер АТС, а АТС разных городов можно сгруппировать и назначить им один и тот же междугородный телефонный код. При назначении телефонных номеров обычно соблюдают определенный порядок. Это облегчает маршрутизацию звонков в телефонной сети.

Пример с телефонной сетью очень важен, поскольку с его помощью было показано основное различие между иерархической системой именования, используемой в объединенной сети TCP/IP, и других иерархиях. Суть состоит в том, что разделение адресного пространства по принципу принадлежности к одному административному органу верхнего уровня отнюдь не означает, что это разделение происходит по физическому положению узлов сети. Например, может случиться так, что в одном здании некоторого университета находятся и факультет математики, и факультет информатики. Может даже оказаться, что, хотя машины этих двух групп находятся в полностью отдельных административных доменах, они подключены к общей физической сети. Кроме того, машины, принадлежащие к одной административной группе, могут находиться в разных физических сетях. Поэтому принятая в протоколе TCP/IP система именования не привязана к физическим сетевым подключениям и позволяет делегировать полномочия по управлению частью пространства имен произвольным административным группам. Эта концепция может быть сформулирована так.

*Иерархические имена машин в объединенной сети TCP/IP присваиваются в соответствии со структурой организации, которой переданы полномочия по управлению частью пространства имен. Сам процесс назначения имени никак не связан со структурой физических сетевых подключений.*

Конечно, во многих сетевых центрах иерархическая структура организации соответствует структуре физических подключений к сети. Например, у большинства отделов в большом университете есть собственная локальная сеть. Если отдел уполномочен управлять частью иерархии пространства имен, то все машины, имена которых принадлежат к его части иерархии, будут также соединены в единую физическую сеть.

## 24.7. Имена доменов сети Internet

Механизм, с помощью которого реализована иерархическая система именования машин в объединенной сети TCP/IP, называется *системой доменных имен* (*Domain Name System*, или *DNS*). DNS имеет два концептуально независимых аспекта. Первый из них является абстрактным: он определяет синтаксис имен и правила делегирования полномочия по управлению частью пространства имен. Второй — конкретный: он определяет работу распределенной вычислительной системы, которая эффективно преобразует имена в адреса. В этом разделе рассмотрен синтаксис имен, а в следующих разделах — практическая реализация механизма преобразования имен.

В принятой системе именования используется иерархический метод присвоения имен, которые называются *доменными именами*. Как и в примерах, приведенных в начале главы, доменное имя состоит из набора алфавитно-цифровых символов (строк), разделенных точкой. В предыдущих примерах говорилось, что отдельные части имени могут соответствовать сетевым центрам или административным группам, но в системе доменных имен каждая часть называется *меткой* (*label*). Таким образом, имя домена

cs.purdue.edu

состоит из трех меток: *cs*, *purdue*, и *edu*. Любой суффикс метки в имени домена также называется *доменом*. В вышеупомянутом примере домен самого низкого уровня — *cs.purdue.edu* (имя домена факультета информатики университета Пердью), домен второго уровня — *purdue.edu* (имя домена университета Пердью), и домен верхнего уровня — *edu* (имя домена образовательных учреждений). Как видно из примеров, в начале имени домена указывается локальная метка, за которой следует имя домена высшего уровня. Как будет показано ниже, подобная форма записи позволяет сжимать сообщения, содержащие повторяющиеся имена доменов.

## 24.8. Официальные и неофициальные имена доменов в глобальной сети Internet

Теоретически стандарт доменных имен определяет абстрактное иерархическое пространство имен с произвольными значениями меток. Поскольку доменная система определяет только форму имен, а не их фактические значения, то любая административная группа может создать собственную независимую систему доменных имен и по своему усмотрению выбрать названия для меток всех частей иерархии. Например, частная компания может создать собственную иерархию доменов, в которой метки самого верхнего уровня будут определять филиалы корпорации, метки следующего уровня — подразделения в филиале, а метки нижнего уровня — отделы.

Однако большинство пользователей придерживается иерархии меток, официально принятых в системе доменных имен сети Internet. Это происходит по двум причинам. Во-первых, как станет ясно позже, структура сети Internet — универсальная и гибкая. Она может применяться в большом числе организаций и позволяет каждой административной группе выбирать свой метод разделения пространства имен: по территориальному и по организационному признаку. Во-вторых, структура большинства сетевых центров соответствует структуре, принятой в глобальной сети Internet. Это позволяет им подключать свои сети к Internet, не изменяя при этом имен отдельных машин и доменов. Поскольку принятая в Internet система назначения имен подходит практически для всех случаев, во всех примерах, приведенных далее, используются метки, взятые из иерархии глобальной сети Internet. Однако не стоит забывать, что, хотя эти метки встречаются чаще всего в Internet, система доменных имен позволяет при желании создать любую другую иерархию меток.

Центральным административным органом глобальной сети Internet утверждены несколько официальных доменов верхнего уровня, перечисленные в табл. 24.1. Кроме того, предложено для использования еще несколько дополнительных доменов верхнего уровня, но они не были формально приняты: *.firm*, *.store*, *.web*, *.arts*, *.rec*, *.info* и *.nom*. Следует отметить, что обработка меток имени домена выполняется без учета регистра их символов. Поэтому домен *.EDU* эквивалентен *.edu*.

**Таблица 24.1. Список официальных доменов верхнего уровня глобальной сети Internet и их описание**

Имя домена	Описание
<i>.com</i>	Коммерческие организации
<i>.edu</i>	Образовательные учреждения (4-летние)
<i>.gov</i>	Государственные учреждения
<i>.mil</i>	Военные группы

Окончание табл. 24.1

<b>Имя домена</b>	<b>Описание</b>
.net	Основные центры поддержки сети Internet
.org	Организации, не упомянутые выше
.arpa	Временный домен ARPANET (устаревший)
.int	Международные организации
Географический код страны	Организации, относящиеся к стране с указанным географическим кодом (территориальная схема разделения пространства имен)

Концептуально принятая система имен доменов верхнего уровня позволяет использовать две различные иерархии присвоения имен: территориальную и организационную. В территориальной системе машины разделяются по географическому положению. Например, компьютеры, подключенные к сети и территориально расположенные в Соединенных Штатах Америки, относятся к домену верхнего уровня .us. Если какая-либо страна захочет зарегистрировать собственный географический домен, то центральный административный орган Internet создаст для нее новый домен верхнего уровня и присвоит ему метку в соответствии со стандартным международным двухбуквенным классификатором страны. Например, администрация домена .us поделила его на домены второго уровня, соответствующие каждому штату США. Скажем, для штата Виргиния домен второго уровня выглядит так:

va.us

Кроме территориальной системы иерархии, принятая система именования доменов верхнего уровня позволяет сгруппировать организации по типу. Если организация хочет зарегистрировать собственный домен второго уровня, она должна сообщить центральному административному органу его имя, а также имя домена верхнего уровня, к которому он будет относиться. Рассмотрев заявку, центральный административный орган выделяет организации *поддомен*<sup>2</sup> в одном из существующих доменов верхнего уровня. Например, администрация университетской сети может зарегистрировать на себя домен второго уровня в домене .edu (обычная практика) либо в домене штата и страны, в которой расположен университет. Пока что немногие организации выбрали территориальную иерархию имен. Большинство предпочитает регистрироваться в доменах .com, .edu, .mil или .gov. Это происходит по двум причинам. Во-первых, географические имена длиннее и, соответственно, более трудны для запоминания и набора. Во-вторых, поскольку географические имена менее очевидны, их труднее соотнести с названием организации. Например, университет Пердью расположен в городе Вест Лафайетт (West Lafayette), штат Индиана. С точки зрения пользователя очень легко запомнить или угадать доменное имя организации, типа purdue.edu. Что касается территориальных имен, то конструкция типа laf.in.us не выдерживает никакой критики, хотя она и состоит из официально зарегистрированных региональных кодов.

С помощью следующего примера мы проиллюстрируем принцип передачи полномочий при регистрации иерархических имен. Одна из машин факультета информатики университета Пердью названа xinu; ее официальное доменное имя

xinu.cs.purdue.edu

<sup>2</sup> В стандарте нет определения термину *поддомен* (*subdomain*). Однако здесь мы использовали его по аналогии с термином *подмножество* (*subset*), чтобы помочь читателю прояснить существующие взаимосвязи между доменами.

Это имя было авторизовано и зарегистрировано администратором локальной сети факультета информатики, который получил полномочия на управление поддоменом `cs.purdue.edu` от администрации университетской сети, которая, в свою очередь, получила полномочия на управление поддоменом `purdue.edu` от центрального административного органа Internet. Последний сохраняет контроль над доменом `.edu`, поэтому новые домены второго уровня для учебных заведений могут быть зарегистрированы только с его разрешения. Точно так же администрация сети университета Пердью сохраняет свои полномочия по управлению доменом `purdue.edu`. Поэтому новые домены третьего уровня могут быть добавлены только с ее ведома.

На рис. 24.1 показана небольшая часть иерархии доменных имен глобальной сети Internet. Как видно из рисунка, фирма IBM зарегистрировала свой домен второго уровня `ibm.com` в домене `.com`, относящемся к коммерческим организациям. Университет Пердью зарегистрировал домен `purdue.edu`, а Национальный научный фонд США (National Science Foundation) — домен `nsf.gov`, поскольку он является государственной организацией. Корпорация национальных исследовательских инициатив (Corporation for National Research Initiatives, или CNRI), напротив, выбрала регистрацию в территориальном домене `cnri.reston.va.us`<sup>3</sup>.

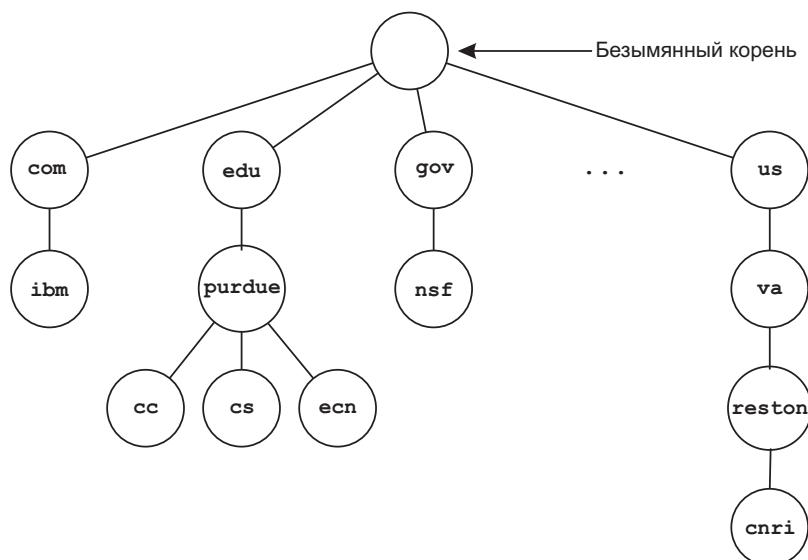


Рис. 24.1. Небольшая часть дерева иерархии имен доменов глобальной сети Internet. В действительности дерево гораздо разветвленнее. Обычно имена узлов сети располагаются на четвертом или пятом уровнях

## 24.9. Именованные элементы и синтаксис имен

Система доменных имен является универсальной, поскольку допускает вложенность нескольких иерархий присваивания имен в одной системе. Чтобы клиенты могли легко ориентироваться во множестве элементов, каждому поименованному элементу, хранящемуся в системе, присваивается *тип*. Он определяет, чем является данный элемент: адресом машины, именем почтового ящика, поль-

<sup>3</sup> Интересно, что CNRI также зарегистрировала доменное имя `nri.reston.va.us`.

зователя и т.д. Посылая серверу доменных имен запрос на преобразование имени в адрес, клиент должен определить тип предполагаемого ответа. Например, когда программа работы с электронной почтой использует систему доменных имен для преобразования строковых адресов электронной почты в IP-адреса машин, она должна указать в запросе, что в ответе ожидается адрес *почтового сервера*, выполняющего пересылку сообщений адресату (*mail exchanger*). Программа для регистрации на удаленной машине указывает в запросе, что ее интересует IP-адрес этой машины. Важно понять следующее.

*Имя может соответствовать нескольким элементам системы доменных имен. Отправляя запрос, клиент должен указать предполагаемый тип объекта, имя которого нужно преобразовать, а сервер возвращает объекты указанного типа.*

Кроме определения типа возвращаемого объекта, доменная система имен позволяет клиенту определить семейство протоколов, которое будет использоваться. В доменной системе полное пространство имен разделено на *классы*. Это позволяет хранить в одной базе данных информацию о привязках имен к сетевым адресам, относящимся к разным семействам протоколов<sup>4</sup>.

Проанализировав синтаксис имени нельзя определить, к какому типу объекта или классу семейства протоколов оно относится. В частности, количество меток в имени не определяет, относится ли оно к индивидуальному объекту (машине) или домену. Например, имя `gwen.purdue.edu` определяет физический компьютер в Internet. И это при том, что имя `cs.purdue.edu` относится к поддомену. Можно подытожить этот важный момент следующим образом.

*Невозможно отличить имена поддоменов от имен индивидуальных объектов или их типов, проанализировав только синтаксис доменных имен.*

## 24.10. Преобразование доменных имен в адреса

Кроме синтаксиса имен и правил делегирования полномочий, в систему доменных имен включен эффективный, надежный, распределенный и универсальный механизм, выполняющий преобразование имен в адреса. Его реализация является распределенной в техническом смысле этого слова. Это означает, что для решения проблемы преобразования имен в адреса во всех сетевых центрах запускают несколько серверов, которые тесно взаимодействуют друг с другом. Система эффективна в том смысле, что большинство имен может быть преобразовано локально. Поэтому общий трафик службы доменных имен в Internet небольшой. Она может считаться универсальной системой, поскольку ее элементы не обязательно должны соответствовать именам машин (хотя ниже будет использоваться именно этот пример). И наконец, система доменных имен надежна в том смысле, что она будет корректно функционировать, даже если произойдет сбой в работе одного или нескольких удаленных серверов.

Механизм преобразования доменных имен состоит из ряда независимых, но взаимодействующих между собой компьютерных систем, называемых *серверами имен* (*name servers*). Сервер имен — это программа, запущенная на компьютере, выполняющем роль сервера, которая преобразует имена доменов в IP-адреса. Очень часто эта программа запускается на специализированном компьютере. В этом случае сама машина также называется сервером имен. Клиентская часть программы называется *распознавателем имен* (*name resolver*). В процессе преобразования имени она обращается к одному или нескольким серверам имен.

---

<sup>4</sup> На практике такая возможность используется крайне редко.

Для того чтобы понять, как работают серверы доменных имен, нужно расположить их в виде древовидной структуры, которая соответствует иерархии присвоения имен, показанной на рис. 24.2. Корень дерева соответствует серверу, который распознает домены верхнего уровня и “знает”, какой сервер распознает каждый из доменов. Корневой сервер управляет процессом преобразования каждого имени и назначает для этого соответствующий сервер. На следующем уровне находятся серверы имен, каждый из которых отсылает ответ на запрос, относящийся к домену верхнего уровня (например, .edu). Сервер на этом уровне “знает”, какой сервер может распознать каждый из поддоменов в его домене. На третьем уровне дерева находятся серверы имен, которые отвечают на запросы, относящиеся к поддоменам (например, purdue в домене .edu). Можно сказать, что дерево домена начинается на самом верхнем уровне и продолжается вниз по уровням, на которых находятся серверы имен, относящиеся ко всем поддоменам этого домена.

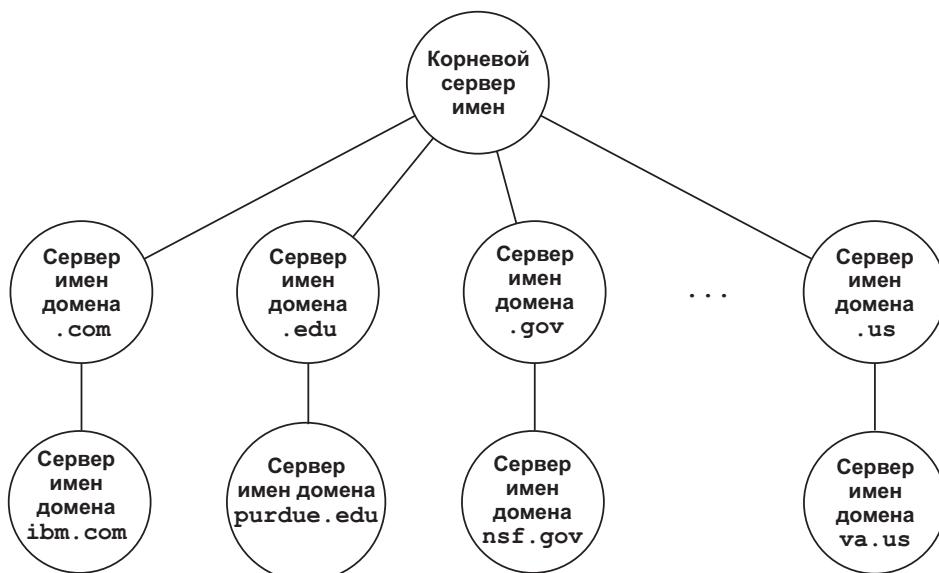


Рис. 24.2. Принципиальная древовидная схема взаимодействия серверов доменных имен, соответствующая иерархии присвоения имен. Теоретически каждый сервер в пределах домена, которым он управляет, “знает” адреса всех серверов низшего уровня, управляющих поддоменами

Связи на этой принципиальной древовидной схеме не соответствуют физическим связям в сети. Они просто показывают серверы имен, которые известны данному серверу, и с какими серверами он напрямую взаимодействует. Серверы сами по себе могут располагаться в произвольных местах объединенной сети. Таким образом, дерево серверов имен — это абстракция, в которой для взаимодействия серверов между собой используется объединенная сеть.

Если бы серверы в доменной системе имен работали так, как в описанной только что упрощенной модели, то организовать взаимодействие серверов, принадлежащих разным административным единицам, было бы очень легко. В самом деле, если организации переданы права на управление некоторым поддоменом, то она должна запустить у себя сервер доменных имен и связать его с описанным выше деревом.

На практике же взаимосвязь между иерархией присваивания имен и деревом серверов не столь проста, как в описанной модели. Обычно дерево серверов имеет

всего несколько уровней, поскольку на одном физическом сервере можно разместить информацию для больших частей иерархии пространства имен. В частности, организации часто выделяют один физический сервер имен для обслуживания всех своих поддоменов. На рис. 24.3 показана реальная схема взаимодействия серверов для иерархии присваивания имен, показанной на рис. 24.1.

На корневом сервере хранится информация, необходимая для поддержки системы доменных имен верхнего уровня. Каждая организация использует один сервер для управления своим пространством имен. Поскольку дерево серверов неглубокое, для распознавания имен типа `xinu.cs.purdue.edu` нужно, чтобы два сервера (корневой и сервер домена `purdue.edu`) связались между собой. Другими словами корневой сервер “знает”, какой из серверов имен управляет доменом `purdue.edu`, а вся информация об этом домене находится на одном сервере.



Рис. 24.3. Реальная схема взаимодействия серверов для иерархии присвоения имен, показанной на рис. 24.1. Поскольку дерево широкое и неглубокое, о для преобразования практически любого доменного имени необходимо, взаимодействие небольшого количества серверов имен

## 24.11. Распознавание доменных имен

Хотя принципиальная древовидная схема облегчает понимание существующих взаимосвязей между серверами доменных имен, на ней не показано несколько тонких деталей. Для пояснения рассмотрим алгоритм распознавания доменного имени. По идеи распознавание должно происходить сверху вниз — от корневого сервера вниз к серверам, расположенным на листьях дерева. Клиент может использовать систему доменных имен двумя способами. Во-первых, он может выполнять преобразование в процессе диалога с сервером, обращаясь к нему каждый раз для преобразования очередного уровня доменного имени. Во-вторых, он может запросить сервер преобразовать имя целиком. В любом случае клиентская программа должна сформировать запрос, поместив в него имя, которое должно быть преобразовано, информацию о его классе, желаемый тип ответа и код, указывающий, должен ли сервер имен преобразовать имя полностью. После этого клиентская программа отправляет запрос серверу имен.

Получив запрос, сервер доменных имен проверяет, не относится ли указанное клиентом имя к поддомуену, на который ему переданы полномочия по управлению. Если это так, то он с помощью своей базы данных преобразует имя в адрес, добавляет эту информацию в пользовательский пакет запроса и возвращает его клиенту. Если сервер не может распознать имя полностью, он проверяет, какой тип взаимо-

действия определен клиентом. Если клиент затребовал полное преобразование имени (в терминологии доменных имен такое преобразование называют *рекурсивным*), сервер обращается с запросом к тому серверу доменных имен, который может распознать имя, и возвращает ответ клиенту. Если клиент затребовал нерекурсивное (*итеративное*) преобразование, сервер имен не может отправить информацию клиенту. В этом случае он генерирует ответ, где указывает адрес другого сервера имен, с которым клиент должен связаться для распознания имени.

Как же клиент находит сервер имен, которому нужно послать первый запрос? А как один сервер имен находит адрес другого сервера имен, который может ответить на запрос клиента? Ответ прост. Клиент должен знать, как связаться по крайней мере с одним сервером имен. А для того чтобы сервер доменных имен мог связаться с другими серверами системы, каждый сервер должен знать адрес по крайней мере одного корневого сервера<sup>5</sup>. Кроме того, серверу имен может быть известен адрес другого сервера, находящегося выше его на один уровень иерархии (он называется *родительским*).

Для взаимодействия между собой серверы доменных имен используют стандартный номер порта протокола. Поэтому клиенты всегда могут связаться с сервером доменных имен при условии, что им известен только IP-адрес машины, на которой этот сервер запущен. Что касается способов поиска IP-адреса такой машины в локальном сетевом окружении, то стандартного решения пока не существует, — все зависит от разработчиков клиентского программного обеспечения<sup>6</sup>. В некоторых системах IP-адрес машины, на которой запущена служба доменных имен, встраивается в прикладные программы еще на этапе компиляции. В других системах он определяется при начальной загрузке компьютера. В третьих — администратор прописывает вручную этот адрес и сохраняет его в файле на вспомогательном запоминающем устройстве.

## 24.12. Эффективное преобразование имен

Несмотря на то что процесс распознавания имен, выполняемый сверху вниз по дереву серверов имен, кажется вполне естественным и понятным, он неэффективен по трем причинам. Во-первых, в большинстве случаев клиенты присылают запросы на преобразование локальных имен, т.е. имен, относящихся к той же части пространства имен, что и машина, от которой приходит запрос. Поэтому прохождение пути по всей иерархии серверов сверху вниз до осуществления контакта с сервером имен, которому переданы полномочия по управлению локальным доменом, неэффективно. Во-вторых, если бы каждый процесс преобразования доменного имени начинался с обращения к серверу имен, находящемуся на самом верхнем уровне иерархии, то это привело бы к его быстрой перегрузке. В-третьих, отказ в работе сервера, находящегося на верхних уровнях иерархии не позволил бы распознать имя, несмотря на то, что локальный сервер имен мог бы сделать это без особых проблем. Сказанное легко пояснить на уже использовавшемся примере иерархии телефонных номеров. Несмотря на то что телефонные номера назначаются в соответствии с принятой иерархической структурой, их распознавание происходит снизу вверх. Поскольку большинство телефонных звонков местные, они могут быть распознаны и обработаны местной АТС без установки связи с другими АТС. Кроме того, звонки, сделанные в пределах одного

<sup>5</sup> Чтобы повысить надежность работы системы, для каждого узла в дереве доменов создается несколько серверов имен. Кроме того, для снижения нагрузки на корневой сервер имен он тиражируется на несколько физических машин.

<sup>6</sup> Один из возможных подходов описан в главе 23, “Начальная загрузка и автоконфигурация (BOOTP, DHCP)”.

региона (т.е. внутри области с одним междугородным телефонным кодом) могут быть обработаны без привлечения телефонных станций, находящихся в зоне действия другого междугородного телефонного кода. Применив описанный алгоритм к системе доменных имен, получим двухэтапный механизм распознавания имен, при использовании которого сохраняется административная иерархия и эффективно выполняется преобразование доменных имен.

Выше уже было сказано, что на сервер имен часто приходят запросы на преобразование локальных имен. Процесс распознавания имен при использовании двухэтапного алгоритма начинается на локальном сервере имен. Если же локальный сервер не может распознать имя, то запрос перенаправляется другому серверу системы доменных имен.

### 24.13. Кэширование — залог эффективности

Стоимость преобразования нелокальных имен может быть чрезвычайно высока, если клиентские программы (распознаватели) будут каждый раз посыпать запрос на корневой сервер. Даже если бы запросы сразу поступали на нужный сервер, имеющий полномочия по управлению данным доменом, процесс преобразования имен представлял бы большую нагрузку на объединенную сеть. Таким образом, чтобы повысить общую производительность системы доменных имен, необходимо понизить стоимость преобразования нелокальных имен.

Для оптимизации процесса поиска имен серверы глобальной сети Internet используют механизм *кэширования имен*. То есть каждый сервер сохраняет в локальной кэш-памяти информацию о недавно использовавшихся именах, а также сведения о том, какой из серверов предоставил данные для преобразования этого имени. Получив от клиента запрос на распознание имени, сервер, согласно стандартной процедуре, сначала проверяет, имеет ли он полномочия на управление доменом, к которому принадлежит то имя. Если нет, то сервер выполняет поиск данного имени в своей кэш-памяти. В случае нахождения адресной привязки сервер имен извлекает ее из кэш-памяти и отправляет клиенту. При этом сервер уведомляет клиента, что привязка получена не из достоверного источника (*nonauthoritative*) и передает вместе с ней доменное имя сервера  $S$ , от которого он получил эту привязку. Локальный сервер посылает также клиенту дополнительную информацию, в которой указана привязка между именем сервера  $S$  и его IP-адресом. В результате клиенты получают ответы быстро, однако содержащаяся в них информация может быть устаревшей. Поэтому, если во главу угла ставится эффективность, клиент воспользуется ответом, полученным из недостоверного источника, и продолжит работу. Если же система должна работать без ошибок, клиент обязан направить запрос официальному серверу домена  $S$  и проверить достоверность полученной привязки между именем и адресом.

Кэширование хорошо работает в системе доменных имен, потому что привязки имен к адресам редко изменяются. Тем не менее рано или поздно это все-таки происходит. Поэтому если сервер поместит в кэш адресную привязку в тот момент, когда она была в первый раз затребована, и больше никогда не будет ее изменять, содержимое кэш-памяти со временем устареет и может стать неправильным. Для поддержания в кэш-памяти актуальной информации сервер имен контролирует время нахождения каждой привязки в кэше и удаляет из него элементы, время пребывания которых превысило установленное значение. Получив запрос после того, как соответствующий элемент удален из кэш-памяти, сервер должен обратиться к достоверному источнику и заново получить привязку. Важно то, что серверы имен не используют единое фиксированное время хранения для всех элементов кэш-памяти. Время хранения привязки в кэше определяется достоверным источником, от которого эта привязка получена. Всякий раз, когда официальный сервер имен отвечает на запрос, он помещает в отклик значение

*времени жизни* (*Time To Live*, или TTL). Это значение определяет, на какой срок привязка останется без изменений. Следовательно, для уменьшения нагрузки на сеть администратор официального сервера имен должен указать большие значения времени жизни для тех доменов, которые, как предполагается, не будут часто изменяться. Если же предполагается, что содержимое домена будет изменяться довольно часто, значение времени жизни должно быть достаточно малым, чтобы обеспечить достоверность информации.

Кэширование может осуществляться как на узлах сети, так и на локальных серверах доменных имен. Во многих системах, работающих в режиме разделения времени, используется сложная форма кода программы распознавателя, благодаря которой удается обеспечить высокую эффективность работы системы (иногда даже более высокую, чем на сервере имен). Работа таких систем основана на том, что при запуске программа распознавателя имен загружает с локального сервера доменных имен полную базу данных имен и адресов и поддерживает собственный кэш недавно использованных имен. Программа обращается к серверу только в том случае, когда не может найти имена в своей базе данных. Конечно, узел сети, на котором содержится копия базы данных локального сервера имен, должен периодически сверяться с сервером и получать от него новые привязки. Кроме того, по истечении времени жизни программы распознавателя должна удалить соответствующие привязки из своего кэша. Поскольку имена доменов изменяются нечасто, у узла сети, как правило, не возникает серьезных проблем с поддержанием согласованности этих баз данных.

Хранение копии базы данных локального сервера имен на каждом узле сети имеет несколько преимуществ. Очевидно, что благодаря этому существенно ускоряется процесс распознавания имен на локальных узлах сети, поскольку программа распознавателя может работать автономно, не обращаясь к сети. Кроме того, обеспечивается защита узла сети, если локальный сервер имен выйдет из строя. И наконец, уменьшается вычислительная нагрузка на сервер имен, что позволяет ему обслуживать большее количество машин.

## 24.14. Формат сообщений системы доменных имен

Подробное рассмотрение структуры сообщений, которыми обмениваются клиенты и серверы системы доменных имен, поможет объяснить принцип работы системы с точки зрения типичной прикладной программы. Предположим, пользователь запускает прикладную программу и задает имя машины, с которой эта программа должна взаимодействовать. Прежде чем прикладная программа сможет использовать протоколы типа TCP или UDP для обмена данными с указанной машиной, она должна определить IP-адрес этой машины. Для этого программа передает доменное имя машины локальному распознавателю и запрашивает у него ее IP-адрес. Локальный распознаватель проверяет свой кэш и возвращает ответ, если это имя в нем присутствует. Если локальный распознаватель не смог найти имя в кэше, он формирует сообщение системы доменных имен и посыпает его серверу (т.е. теперь он становится клиентом). Хотя в нашем примере было использовано только одно имя, формат сообщения позволяет клиенту делать несколько запросов в одном сообщении. Каждый запрос состоит из доменного имени, для которого клиент ищет IP-адрес, спецификации класса запроса (т.е., указывается, где нужно выполнять поиск, например в *объединенной сети*) и типа нужного объекта (например, *адреса*). В качестве ответа сервер возвращает сообщение, сформированное клиентом, в которое помещает имеющуюся у него информацию о привязках. Если же сервер не может ответить на все запросы, то в ответе будет указано, с какими серверами имен клиент должен связаться, чтобы получить ответы.

В ответ сервера имен включается также информация об официальных серверах имен для данного домена и их IP-адреса. Формат сообщения системы доменных имен показан на рис. 24.4. Каждое сообщение начинается с заголовка фиксированного формата. В заголовке предусмотрено специальное поле *идентификации*, куда клиент помещает уникальное значение для сопоставления запросов и ответов сервера. В поле *параметров* помещается тип запрашиваемого действия и код ответа. Значение битов этого поля приведено в табл. 24.2. Биты перечислены слева направо, начиная с нулевого.

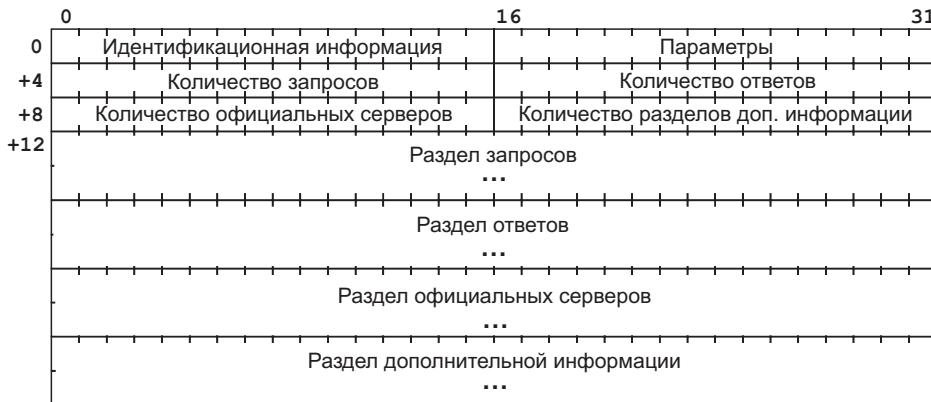


Рис 24.4. Формат сообщения системы доменных имен. Разделы запросов, ответов, официальных серверов и дополнительной информации имеют переменную длину

**Таблица 24.2. Значение битов поля параметров в сообщении системы доменных имен**

Номер бита	Описание
0	Тип операции: 0 — запрос 1 — ответ
1–4	Тип запроса: 0 — стандартный 1 — инверсный 2 — окончание 1 (устарел) 3 — окончание 2 (устарел)
5	Устанавливается, если ответ получен из ненадежного источника
6	Устанавливается, если сообщение усечено
7	Устанавливается, если желательна рекурсивная обработка
8	Устанавливается, если выполняется рекурсивная обработка
9–11	Зарезервированы
12–15	Тип ответа: 0 — операция выполнена без ошибок

Окончание табл. 24.2

Номер бита	Описание
1	неверный формат запроса
2	сбой в работе сервера
3	имя не существует

В четырех последующих полях указывается количество элементов в соответствующих разделах сообщения. Например, в поле *Количество запросов* указывается количество элементов, которые находятся в разделе запросов сообщения.

Клиент заполняет только раздел запросов, помещая сюда список всех своих вопросов к серверу. Сервер возвращает запросы клиента и ответы на них в одном сообщении. Каждый запрос состоит из *доменного имени*, за которым указываются *тип запроса* и его *класс*, как показано на рис. 24.5.

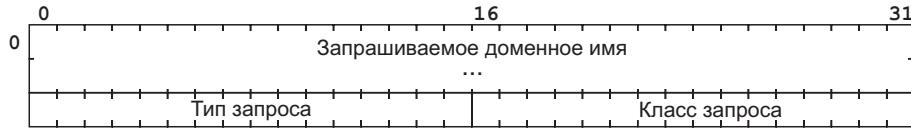


Рис. 24.5. Формат элемента в разделе запросов сообщения системы доменных имен. Доменное имя имеет переменную длину

Хотя поле *запрашиваемого доменного имени* имеет переменную длину, в следующем разделе мы увидим, как формат внутреннего представления доменных имен позволяет получателю узнать точную длину имени. Тип запроса (например, относится ли запрос к имени машины или к почтовому адресу) указывается в одноименном поле элемента в разделе запросов. Поле *класса запроса* позволяет использовать доменную систему имен для работы с объектами произвольного типа. Все дело в том, что при работе в глобальной сети Internet разрешен только один класс запросов, относящийся к официальным именам. Следует заметить, что, хотя на рис. 24.5 показан формат элемента запросов кратный 32-битам, поле *запрашиваемого доменного имени* может содержать произвольное количество октетов. Причем байты выравнивания на границу 32 битов не используются. Поэтому сообщения, посылаемые на серверы доменных имен или в обратном направлении, могут содержать нечетное количество октетов.

В сообщении системы доменных имен каждый из оставшихся разделов (ответов, официальных серверов и дополнительной информации) состоит из набора записей *ресурса*, которые описывают доменные имена и их соответствие адресам. В каждой записи ресурса описывается только одно имя. Формат записи показан на рис. 24.6.

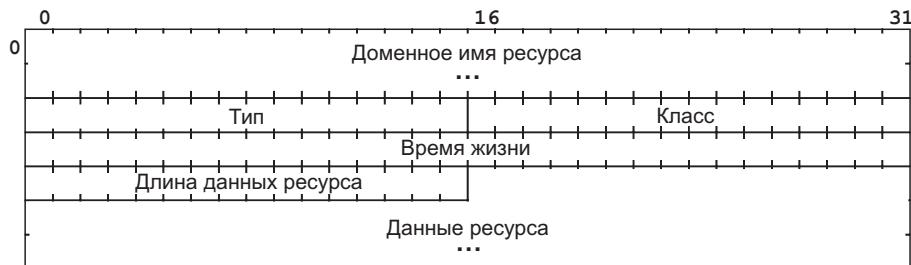


Рис. 24.6. Формат записи ресурса, используемой в трех последних разделах сообщения системы доменных имен, возвращаемых серверами имен

В поле *доменного имени ресурса* указывается имя домена, к которому относится запись ресурса. Имя может иметь произвольную длину. В поле *типа* кодируется тип данных, включенных в запись ресурса, а поле *класса* определяет класс этих данных. В поле *времени жизни* сервер помещает целое 32-битовое число, определяющее время в секундах, в течении которого информация об этой записи может сохраняться в локальной кэш-памяти. Значение этого поля используется клиентами, которые запросили привязку имени и, возможно, собираются поместить ее в кэш. Последние два поля содержат результаты поиска привязки: в поле *длины данных ресурса* указывается количество октетов, находящихся в поле *данных ресурса*.

## 24.15. Сжатый формат представления имен

Доменные имена представляются в сообщении в виде последовательности меток. Каждая метка начинается с октета, который определяет ее длину. Следовательно, для восстановления доменного имени получатель должен поочередно извлечь длину метки,  $n$ , из октета длины, а затем —  $n$  октетов самой метки, расположенной следом. Конец доменного имени обозначается октетом длины, содержащим нулевое значение.

Серверы доменных имен часто возвращают несколько ответов на запрос и во многих случаях в них указываются одинаковые суффиксы домена. Для того чтобы сэкономить место в ответном пакете, серверы сжимают имена, сохраняя только одну копию каждого доменного имени. При извлечении доменных имен из сообщения клиентская программа должна проверить каждый сегмент имени на предмет того, содержит ли оно строку символов (в формате: один октет счетчика длины, за которым следуют символы, составляющие имя) или указатель на строку символов. Если обнаружен указатель, клиентская программа должна перейти на новое место в сообщении и найти остаток имени.

Указатели всегда размещаются в начале сегментов и кодируются в байте счетчика. Если два старших бита 8-битового октета счетчика равны единице, то клиент должен использовать следующие 14 битов как целочисленный указатель. Если же два старших бита равны нулю, то следующие 6 битов определяют количество символов в метке, которая следует за октетом счетчика.

## 24.16. Сокращение доменных имен

На примере иерархии присвоения телефонных номеров можно увидеть другую полезную особенность локального распознавания имен — возможность *сокращения имен*. Сокращение позволяет укорачивать имена, если в процессе распознавания часть имени может быть получена автоматически. Звоня по местному телефонному номеру, абонент не набирает междугородный телефонный код. Предполагается, что цифры телефонного номера, вводимые абонентом и составляющие сокращенный номер, относятся к тому же междугородному коду, что и номер телефона абонента. Описанный принцип сокращения также хорошо работает и в случае имен машин. В процессе распознавания имени типа *хуз* можно предположить, что это имя и машина, на которой оно распознается, относятся к одному домену. Таким образом, распознаватель может автоматически восполнять отсутствующие части имени. Например, в пределах факультета информатики университета Пердью, сокращенное имя *xinu* будет эквивалентно полному доменному имени *xinu.cs.purdue.edu*.

В большинстве клиентских программ поддерживается возможность использования списка суффиксов домена, на основании которого реализуется процесс

сокращения имен. При формировании этого списка администратор локальной сети указывает в нем перечень возможных суффиксов домена, которые автоматически будут добавляться к концу сокращенного имени при выполнении поиска. Получив имя, распознаватель извлекает элементы этого списка и добавляет их по одному к имени, каждый раз проверяя полученное имя. Например, список суффиксов домена факультета информатики университета Пердью выглядит так:

```
.cs.purdue.edu  
.cc.purdue.edu  
.purdue.edu  
нуль
```

Таким образом, при поиске имени `xinu` локальный распознаватель сначала добавит к нему суффикс `cs.purdue.edu`. Если полученное в результате имя не будет найдено, то распознаватель добавит к имени `xinu` другой суффикс — `cc.purdue.edu` — и попытается распознать это имя и т.д. Последним элементом списка в нашем примере указана пустая строка. Она означает, что, если предыдущие операции поиска оказались неудачными, распознаватель попытается распознать имя без добавления суффикса. Список суффиксов домена может использоваться сетевым администратором для того, чтобы предоставить пользователям удобные и легко запоминающиеся короткие имена, а также для возможности использования локальных имен в прикладных программах.

Выше уже отмечалось, что при использовании сокращений их расширением занимается клиентская программа. При этом стоит подчеркнуть, что сами по себе сокращения не являются частью системы доменных имен. Доменная система имен позволяет искать только полностью определенные доменные имена. Вследствие этого программы, использующие сокращенные имена, могут работать неправильно за пределами того окружения, для которого они были созданы. Можно подытожить все сказанное выше так.

*Система доменных имен позволяет преобразовывать в адреса только полностью определенные доменные имена. Сокращения сами по себе не являются частью системы доменных имен, а используются в клиентских программах для удобства работы пользователей, которые привыкли к коротким локальным именам машин.*

## 24.17. Инверсные преобразования

Как было сказано выше, система доменных имен позволяет преобразовывать не только имена машин в IP-адреса, но и выполнять любые другие преобразования. В данном разделе речь пойдет о так называемых *инверсных преобразованиях*, которые позволяют клиентской программе запросить сервер выполнить преобразование имени в обратном направлении. То есть, на основании готового ответа сервера восстановить запрос клиента, при обработке которого был бы получен такой же ответ. Конечно, такое преобразование не всегда дает однозначный результат, т.е. одинаковый ответ сервера может быть получен при обработке нескольких разных запросов. Поэтому при выполнении инверсных преобразований сервер может построить неправильный запрос. Несмотря на то что механизм инверсных запросов с самого начала входил в систему доменных имен, он используется довольно редко. Причина состоит в том, что зачастую сервер не может обработать такой тип запроса без выполнения поиска по целому ряду серверов.

## 24.18. Запросы указателя

Несмотря на изложенные в предыдущем разделе недостатки инверсного преобразования, очевидно, что одна из его форм имеет право на существование, более того, просто необходима. Важность этой формы столь велика, что в системе доменных имен для ее реализации предусмотрен специальный домен и специальная форма запроса, называемая *запросом указателя* (*pointer query*). При использовании запроса указателя клиентская программа формирует обычное сообщение системы доменных имен, только в раздел запросов помещает не доменное имя, а IP-адрес машины, цифры которого представлены в виде текстовой строки и разделены точками. На запрос указателя сервер имен должен возвратить доменное имя машины, соответствующей указанному IP-адресу. Запросы указателя особенно полезны для бездисковых машин, поскольку с их помощью компьютер по IP-адресу может определить свое высокогородовое имя. О том, как бездисковая машина определяет свой IP-адрес, речь шла в главах 6, “Определение IP-адреса при начальной загрузке (RARP)” и 23, “Начальная загрузка и автоконфигурация (BOOTP, DHCP)”.

Запросы указателя нетрудно сгенерировать. Если IP-адрес записать в десятичной форме с точками в качестве разделителей, он будет иметь следующий вид:

aaa.bbb.ccc.ddd

Чтобы сформировать запрос указателя, клиент перестраивает десятичное представление адреса в строкуiformы:

ddd.ccc.bbb.aaa.in-addr.arpa

Новая форма адреса представляет собой имя, относящееся к специальному домену *in-addr.arpa*<sup>7</sup>. Поскольку локальный сервер имен может не быть официальным сервером для домена *arpa*, либо для домена *in-addr.arpa*, для обработки запроса ему может понадобиться связаться с другими серверами имен. Чтобы повысить эффективность обработки запросов указателя, корневые серверы системы доменных имен глобальной сети Internet поддерживают базу данных распределенных IP-адресов, а также информацию о серверах имен, которые могут распознать каждый адрес.

## 24.19. Типы объектов и содержимое записи ресурса

Как уже было сказано, система доменных имен может использоваться для преобразования имен машин в их IP-адреса, а также для преобразования адресов электронной почты в имена почтовых серверов, выполняющих обработку и пересылку электронной почты. Доменная система имен в целом является универсальной, поэтому ее можно применить для любых иерархий имен. Например, может понадобиться организовать справочную службу вычислительного центра, позволяющую по названию предоставляемой услуги определить номер телефона, по которому клиент может позвонить и получить дополнительную информацию о ней. Или, например, можно создать справочную службу производителей программного обеспечения, которая бы по типу используемого протокола находила названия фирм производителей и их адреса.

---

<sup>7</sup> При формировании доменного имени порядок октетов IP-адреса должен быть изменен на обратный. Дело в том, что в IP-адресах наиболее значимые октеты (старшие октеты адреса) находятся в начале, в то время как в доменных именах сначала указывается наименее значимая информация.

Напомним, что подобные преобразования возможны за счет включения поля *типа* в каждую запись ресурса. При отсылке запроса, клиент должен указать его тип<sup>8</sup>. Серверы имен указывают тип данных во всех возвращаемых записях ресурса. Тип определяет содержимое записи ресурса, как показано в табл. 24.3.

**Таблица 24.3. Типы записей ресурса системы доменных имен**

Тип	Значение	Содержимое
A	Host Address (адрес узла сети)	32-битовый IP-адрес
CNAME	Canonical Name (каноническое имя)	Каноническое имя домена для псевдонима
HINFO	Host Info (информация об узле сети)	Название фирмы — производителя центрального процессора компьютера и типа операционной системы
MINFO	Mailbox Info (информация о почтовом ящике)	Информация о почтовом ящике или списке почтовой рассылки
MX	Mail Exchanger (сервер пересылки почты)	16-битовое значение приоритета и имя узла сети, который принимает и пересыпает почту для указанного домена
NS	Name Server (сервер имен)	Имя официального сервера для указанного домена
PTR	Указатель	Имя домена (тип символьной связи)
SOA	Start of Authority (начало полномочия)	Несколько полей, которые определяют, за какую часть иерархии назначения имен отвечает сервер
TXT	Text (произвольный текст)	Неинтерпретируемая ASCII-строка текста

Большинство записей в базе данных имен относится к типу A. Они состоят из имени и соответствующего ему IP-адреса узла сети, подключенного к глобальной сети Internet. Второй часто используемый тип ресурса, MX, предназначен для указания серверов, принимающих и обрабатывающих электронную почту для этого домена. С его помощью сетевой администратор может указать несколько почтовых серверов, обрабатывающих электронную почту для данного домена. При отправке сообщения по электронной почте, пользователь указывает адрес получателя в виде *имя-пользователя@домен*. Почтовая система использует систему доменных имен для преобразования доменной части адреса электронной почты получателя в адрес почтового сервера, указанного в записи типа MX. Доменная система возвращает набор записей ресурса, в каждой из которых содержится значение приоритета и доменное имя почтового сервера. При отправке сообщения почтовая система пытается сначала передать его серверу, которому назначен наивысший приоритет (меньшие числа соответствуют более высокому приоритету). Если передача не удалась, из списка выбирается сервер с меньшим приоритетом, и процесс отправки сообщения повторяется, и т.д. При этом программа доставки электронной почты извлекает из MX-записи ресурса доменное имя очередного почтового сервера и использует запрос типа A, чтобы преобразовать его в IP-адрес. После этого она пытается связаться с узлом сети по указанному адресу и доставить почту. Если этот узел сети недоступен, программа доставки электронной почты попытается связаться с другим узлом, указанным в списке.

<sup>8</sup> В запросах можно указать несколько дополнительных типов (например, существует специальный тип запроса, который позволяет выбрать все записи ресурса).

Чтобы сделать поиск эффективным, сервер всегда возвращает дополнительные привязки, помещая их в раздел *дополнительной информации* сообщения системы доменных имен. В случае записей типа MX сервер имен может поместить в раздел *дополнительной информации* данные, полученные из записи ресурса типа A, соответствующие доменным именам, указанным в разделе *ответа*. В результате программа доставки электронной почты получит в одном сообщении всю нужную ей информацию, что существенно уменьшит количество запросов, которые она должна отправить своему серверу имен.

## 24.20. Получение полномочий для управления поддоменом

Прежде чем получить официальные полномочия на управление доменом второго уровня, организация должна запустить у себя сервер доменных имен, соответствующий стандартам глобальной сети Internet. Естественно, что сервер имен должен удовлетворять стандартам протокола, которые определяют форматы сообщений и правила ответов на запросы. Сервер должен также располагать адресами других серверов имен, которые отвечают за работу каждого поддомена (если такие существуют), а также адресом по крайней мере одного корневого сервера имен.

На практике структура доменной системы имен гораздо сложнее, чем описанная в этой главе. В большинстве случаев один физический сервер может обслуживать несколько иерархий назначения имен. Например, один сервер имен, запущенный в университете Пердью, обслуживает как домен второго уровня *purdue.edu*, так и географический домен *laf.in.us*. Таким образом, поддерево имен, обслуживаемых сервером, составляет его *зону полномочий*. Еще одно затруднение возникает из-за того, что серверы должны одновременно обрабатывать множество запросов, хотя для завершения некоторых из них требуется продолжительное время. Обычно серверы обрабатывают поступившие запросы параллельно. Это позволяет им переходить к обработке вновь поступивших запросов, в то время как запросы, поступившие ранее, еще обрабатываются. Параллельная обработка запросов особенно важна, когда сервер получает рекурсивный запрос, для завершения которого он вынужден обратиться за дополнительной информацией к другому серверу.

Реализация сервера имен осложняется еще и тем, что центральный административный орган глобальной сети Internet требует, чтобы информация на каждом сервере доменных имен дублировалась и автоматически реплицировалась на резервных серверах. Причем количество этих серверов должно быть не меньше двух, и работать они должны на разных физических машинах. Практические требования еще более строгие: выход из строя одного сервера не должен повлиять на работу другого сервера, т.е. у резервных серверов не должно быть общих предпосылок для сбоя. Устранение общих предпосылок для сбоя означает, что резервные серверы имен не могут быть подключены к одной и той же сети и получать электрическое питание из одного источника. Чтобы удовлетворить эти требования, администрация сетевого центра должна договориться с администрацией по крайней мере еще одного сетевого центра и расположить резервный сервер имен в их центре. Естественно, сервер, расположенный в любой точке дерева, должен "уметь" находить как основной, так и резервные серверы имен для данного поддомена. Если основной сервер по какой-либо причине оказывается недоступным, все запросы перенаправляются одному из резервных серверов имен.

## 24.21. Резюме

В иерархической системе назначения имен предусмотрена возможность передачи полномочий по управлению частью пространства имен другим административным единицам. Это позволяет создать общее пространство имен произвольно большого размера и снять нагрузку на центральный административный орган. Хотя процесс распознавания имен не связан с процессом делегирования полномочий, это не мешает создать иерархическую систему назначения имен с эффективным распознаванием имен. Все дело в том, что эффективный процесс распознавания имени должен начинаться на локальном сервере, а делегирование полномочий всегда направлено от вершины иерархии вниз.

В этой главе была рассмотрена система доменных имен (DNS) глобальной сети Internet и было показано, что она представляет собой пример иерархии назначения имен. В DNS используются методы распределенного поиска IP-адресов или имен почтовых адресов по доменному имени. Процесс распознавания имен начинается с отправки клиентской программой запроса локальному серверу имен. Если локальный сервер не может распознать имя, клиент должен обратиться к другому серверу имен, расположенному в другой части дерева, либо запросить у локального сервера имен рекурсивное распознавание. И наконец, было показано, что на основе системы доменных имен можно создать разнообразные привязки, например привязать IP-адреса машин к их высокоуровневым именам.

## Материал для дальнейшего изучения

Основные принципы системы доменных имен сети Internet в общих чертах описаны Мокапетрисом (Mokapetris) в [RFC 1034]. Стандарт протокола системы доменных имен представлен Мокапетрисом в [RFC 1035]. В [RFC 1101] Мокапетрис рассматривает использование системы доменных имен для кодирования сетевых имен, а также предлагает полезные дополнения для осуществления других видов преобразования. Требования, которым должен удовлетворять сервер доменных имен глобальной сети Internet, сформулированы Постелом (Postel) и Рейнольдсом (Reynolds) в [RFC 920]. Руководство администратора по созданию домена приведены Сталем (Stahl) в [RFC 1032], а руководство по запуску сервера доменных имен описано Лоттором (Lottor) в [RFC 1033]. Расширения системы безопасности описаны Истлейком (Eastlake) в [RFC 2535]. Связь между доменными именами и системой адресации электронной почты рассмотрена Партриджем (Partridge) в [RFC 974]. И наконец, интересный анализ роста глобальной сети Internet, полученный за счет обхода дерева доменных имен, приведен Лоттором в [RFC 1296].

## Упражнения

- 24.1.** Объясните, почему не стоит встраивать в операционную систему имена машин при ее компиляции.
- 24.2.** Какое имя бездисковой машины предпочтительней использовать: полученное из удаленного файла конфигурации или от сервера имен? Почему?
- 24.3.** Почему каждый сервер имен должен “знать” IP-адрес родительского сервера имен, а не его доменное имя?
- 24.4.** Разработайте систему имен, допускающую внесение изменений в иерархию назначения имен. В качестве примера рассмотрите случай слияния двух больших компаний, каждая из которых имела

независимую иерархию назначения имен. Можно ли сделать так, чтобы существующие имена остались работоспособными?

- 24.5. Прочитайте стандарт протокола системы доменных имен и выясните, для чего в ней используются записи типа SOA.
- 24.6. Выясните, как в системе доменных имен глобальной сети Internet указывается информация об именах почтовых ящиков.
- 24.7. В стандарте сказано, что если программе нужно найти доменное имя, связанное с определенным IP-адресом, то сначала она должна послать инверсный запрос локальному серверу. И только если сервер не сможет найти имя, следует использовать домен in-addr.arpa. Почему?
- 24.8. Как бы вы организовали систему сокращений в схеме назначения имен доменов? В качестве примера рассмотрите регистрацию двумя сетевыми центрами, домена второго уровня в домене верхнего уровня .edu. Объясните, как каждый тип сокращений может интерпретироваться в обоих сетевых центрах.
- 24.9. Ознакомьтесь с официальным описанием системы доменных имен и создайте клиентскую программу. Попытайтесь распознать имя merlin.cs.purdue.edu.
- 24.10. Расширьте предыдущее упражнение, включив в него запрос указателя. Пробуйте найти доменное имя для адреса 128.10.2.3.
- 24.11. Установите на своем компьютере программу nslookup и используйте ее для распознавания имен и адресов, использовавшихся в двух предыдущих упражнениях.
- 24.12. Если расширить синтаксис доменных имен и добавить в него точку после домена верхнего уровня, то можно было бы избежать путаницы между полными доменными именами и сокращениями. Каковы преимущества и недостатки этого метода?
- 24.13. Прочтите документы RFC, относящиеся к системе доменных имен. Каковы максимальные и минимальные возможные значения, которые могут указываться в поле *времени жизни* записи ресурса на DNS-сервере?
- 24.14. Должна ли система доменных имен распознавать запросы, содержащие только часть имени (т.е. запросы, в которых используются символы подстановки вместо части имени)? Почему?
- 24.15. Администрация сети факультета информатики университета Пердью поместила на своем сервере доменных имен следующую запись ресурса типа A:  
localhost.cs.purdue.edu 127.0.0.1  
Что произойдет, если с удаленного компьютера кто-то попытается с помощью программы ping проверить работоспособность машины с доменным именем localhost.cs.purdue.edu.



# 25

## Приложения: удаленный вход в систему (*TELNET, rlogin*)

### 25.1. Введение

В этой и следующих пяти главах продолжается изучение межсетевого обмена, рассматриваются высокогорневные службы объединенной сети и поддерживающие их протоколы. Эти службы составляют неотъемлемую часть семейства протоколов TCP/IP. Они определяют восприятие объединенной сети пользователями и позволяют продемонстрировать возможности этой технологии.

Мы покажем, что высокогорневые службы расширяют функциональные возможности сетевой связи и дают возможность пользователям и программам взаимодействовать как с автоматизированными службами, запущенными на удаленных машинах, так и с удаленными пользователями. Кроме того, здесь будет показано, что высокогорневые протоколы чаще всего реализуются в виде прикладных программ, а также, что они зависят от служб сетевого уровня, описанных в предыдущих главах. В этой главе мы рассмотрим процесс удаленной регистрации в системе.

### 25.2. Дистанционные интерактивные вычисления

В предыдущих главах мы уже рассматривали, как на основе модели взаимодействия типа клиент/сервер можно организовать вычислительную службу наподобие службы истинного времени, к которой может обращаться большое количество клиентов. При использовании надежных потоковых протоколов типа TCP можно работать с удаленными компьютерами в интерактивном режиме. Представьте себе, например, сервер, на котором запущена служба дистанционного редактирования текста. Чтобы реализовать эту службу, необходима серверная программа, принимающая запросы на редактирование определенного файла, и клиентская программа, формирующая такие запросы. Для обращения к дистанционной службе редактирования текста пользователь должен запустить клиентскую программу. Эта программа устанавливает TCP-соединение с сервером, после чего пересыпает ему сигналы нажатия клавиш и отображает на экране выводимые сервером данные.

Как можно обобщить функции вымышленной нами службы дистанционного интерактивного редактирования? Проблема состоит в том, что использование отдельной серверной программы для реализации каждой вычислительной службы неминуемо приведет к перегрузке машины серверными процессами. Поэтому логично не создавать большое количество специализированных серверов, а реализовать одну универсальную службу, которая бы позволяла пользователю

устанавливать сеанс связи с удаленной машиной и запускать на ней команды. После *удаленного входа в систему* (*remote login*) пользователи получают доступ ко всем командам удаленной системы. Поэтому системным инженерам не нужно создавать специализированные серверы для запуска специфических служб.

Однако обеспечить удаленный вход в систему не так просто. Многие операционные системы компьютеров были разработаны без учета возможности работы в сети. Поэтому они могли поддерживать сеансы связи только с непосредственно подключенными к ним устройствами, например терминалом и клавиатурой. Следовательно, для реализации на таком компьютере сервера удаленной регистрации необходимо внести изменения в программный код операционной системы. Реализация клиентской части интерактивного сервера также может оказаться непростой задачей. В качестве примера рассмотрим операционную систему, в которой при нажатии заданной комбинации клавиш должны выполняться определенные действия. Например, если в локальной системе нажатие комбинации клавиш <CONTROL+C> интерпретируется как аварийное завершение работы выполняемого в настоящее время процесса командной оболочки, то передать код комбинации клавиш <CONTROL+C> на удаленную машину невозможно. Если же клиент передаст на удаленную машину код <CONTROL+C>, то невозможно будет прервать локальный процесс клиента.

Несмотря на технические трудности системные программисты сумели создать программу сервера удаленной регистрации в системе для большинства операционных систем и написать клиентские программы для них. Чаще всего в клиентских программах отменяется локальная интерпретация всех нажатых клавиш, кроме некоторых. В результате пользователь может взаимодействовать с удаленной машиной так, как если бы он находился перед экраном подключенного к ней локального терминала. Резервирование одной или нескольких клавиш дает пользователю возможность выйти в локальное окружение для управления клиентом (например, прервать его работу). Кроме того, в некоторых протоколах удаленной регистрации используется понятие *надежного узла* (*trusted host*). При этом разрешается удаленный вход в систему со стороны таких узлов без проверки паролей. В других протоколах защита обеспечивается за счет шифрования передаваемых данных.

### 25.3. Протокол TELNET

В семейство протоколов TCP/IP входит простой протокол эмуляции удаленного терминала, называемый *TELNET*. Он позволяет пользователю регистрироваться (*log on*) на удаленном компьютере, подключенном к объединенной сети. Клиентская программа протокола TELNET устанавливает TCP-соединение с удаленным компьютером и передает ему коды нажатых клавиш клавиатуры так, будто пользователь нажимает их на клавиатуре, подключенной к удаленной машине. Сервер протокола TELNET возвращает клиентской программе данные, которые выводятся на экран удаленной машины в ответ на переданные коды клавиш. Подобная служба называется *прозрачной*, поскольку при ее использовании для пользователя создается иллюзия того, что его клавиатура и терминал непосредственно подключены к удаленной машине.

Хотя протокол TELNET по сравнению с другими протоколами эмуляции терминала не является сложным, тем не менее он широко используется. Обычно клиентская программа протокола TELNET дает пользователю возможность определять удаленный компьютер по доменному имени либо по IP-адресу. Поскольку клиентская программа протокола TELNET позволяет задать IP-адрес удаленного компьютера, она может использоваться для связи с теми узлами сети, для которых в DNS не задана привязка имени к адресу (например, в процессе отладки программы для работы с системой доменных имен).

Протокол TELNET предоставляет три основных службы. Во-первых, в нем определено понятие *сетевого виртуального терминала* (*network virtual terminal*), обеспечивающего стандартный интерфейс для удаленных систем. Клиентским программам не нужно учитывать детали работы всевозможных удаленных систем — они разрабатываются с учетом использования стандартного интерфейса. Во-вторых, в протокол TELNET включен механизм, позволяющий клиенту и серверу согласовывать параметры каждого сеанса, а также набор стандартных параметров. Например, один из параметров управляет типом кодировки символов (7- или 8-битовые), передаваемых через установленное соединение. И наконец, в протоколе TELNET оба конечных пункта соединения считаются симметричными. В частности, в протоколе TELNET не требуется, чтобы данные, при сланные клиентом, поступали обязательно с клавиатуры. Кроме того, не требуется также, чтобы клиент отображал результаты на экране. Таким образом, протокол TELNET позволяет любой программе стать клиентом. При этом любая из точек соединения может запросить согласование параметров сеанса.

На рис. 25.1 показано, как в прикладных программах реализуются функции клиента и сервера протокола TELNET.



Рис. 25.1. Путь, который проходят данные от клавиатуры пользователя к удаленной операционной системе во время сеанса связи TELNET. Реализация функций сервера TELNET в операционной системе, работающей в режиме разделения времени, обычно требует изменения исходного кода операционной системы

Как следует из рис. 25.1, в момент запуска программы TELNET на машине пользователя она становится клиентом. Клиент устанавливает TCP-соединение с сервером, через которое они будут связываться. Когда соединение установлено, клиент считывает коды нажатых клавиш с клавиатуры пользователя и пересыпает их серверу. При этом он одновременно принимает символы, которые сервер посылает в ответ, и отображает их на экране пользователя. Сервер должен установить TCP-соединение с клиентом и передавать по нему ответные данные, которые должна обрабатывать локальная операционная система.

В действительности работа сервера гораздо сложнее, чем показано на рис. 25.1, поскольку он должен параллельно обрабатывать одновременно поступающие по нескольким соединениям запросы. Обычно главная программа сервера ожидает запросов на соединение со стороны клиентов и запускает новую подчиненную программу для обработки каждого соединения. Таким образом, “Сервер TELNET”, изображенный на рис. 25.1, соответствует подчиненной программе сервера, которая обрабатывает одно соединение. На рис. 25.1 не показана как главная программа сервера, которая ожидает поступления новых запросов на соединение, так и подчиненные программы, обрабатывающие другие соединения.

Для описания точки входа в операционную систему, через которую запущенные прикладные программы, например сервер TELNET, смогут передавать данные (коды нажатия клавиш) для обработки так, будто они приходят с клавиатуры, мы будем использовать термин *псевдотерминал*<sup>1</sup>. Если в операционной системе такая точка отсутствует, создать сервер TELNET невозможно. Если же в операционной системе поддерживается интерфейс псевдотерминала, то сервер TELNET можно реализовать с помощью прикладной программы. Тогда каждая подчиненная программа сервера, обрабатывающая поток данных, поступающий от одного клиента, сможет подключить TCP-соединение к определенному псевдотерминалу.

Реализация сервера TELNET в виде программы уровня приложения имеет как преимущества, так и недостатки. Наиболее очевидное преимущество состоит в том, что при этом сильно облегчаются модификация и управление сервером, по сравнению с использованием кода, встроенного в операционную систему. Очевидный недостаток такой реализации сервера — его малая эффективность. Код каждого нажатия клавиши поступает с клавиатуры пользователя в клиентскую программу через локальную операционную систему, от клиентской программы — снова в локальную операционную систему и затем через объединенную сеть — на машину сервера. По достижении компьютера, на котором запущен сервер, данные должны пройти через операционную систему сервера и поступить на обработку в прикладную программу сервера. От программы сервера данные должны проследовать обратно в операционную систему сервера и поступить на псевдотерминал. И наконец, удаленная операционная система должна доставить выводимые сервером на экран данные локальной клиентской программе, которую запустил пользователь. При этом результат (включая генерацию эхосимвола удаленным компьютером, если этот параметр был установлен) возвращается от сервера к клиенту по тому же пути.

Читатели, которые представляют себе внутреннюю структуру операционных систем, могут оценить, что при выполнении последовательности действий, показанной на рис. 25.1, для обработки каждого нажатия на клавишу потребуется несколько раз переключить контекст процесса. В большинстве операционных систем потребуется дополнительное переключение контекста, поскольку операционная система сервера должна будет передавать символы от псевдотерминала в другую прикладную программу (например, интерпретатор команд). Хотя переключение контекста — операция довольно громоздкая, предложенная схема работы вполне пригодна для применения, поскольку скорость нажатия клавиш пользователями не очень высока.

## 25.4. Учет особенностей оборудования и программного обеспечения

Чтобы программы протокола TELNET могли работать в разных операционных системах и на разных аппаратных платформах, в них должны быть учтены особенности этих систем. Например, в некоторых операционных системах признаком конца строки является ASCII-символ возврата каретки (*carriage return*, или CR), в других — ASCII-символ перевода строки (*linefeed*, или LF), а в третьих — оба символа, CR и LF. Кроме того, в большинстве интерактивных систем пользователь может прервать выполнение текущей программы путем нажатия определенной комбинации клавиш. Однако в разных операционных системах

<sup>1</sup> В системах UNIX эта точка входа называется *псевдо tty* (*pseudo tty*), поскольку аналогичное символьное устройство называется *tty*.

комбинация этих клавиш будет различной. В одних — <CONTROL+C>, в других — <ESCAPE>.

Чтобы учесть особенности всех систем в протоколе TELNET определяется, как данные и управляющие последовательности команд должны передаваться через глобальную сеть Internet. Совокупность этих стандартов образует систему команд так называемого *сетевого виртуального терминала* (*network virtual terminal*, или *NVT*). Как показано на рис. 25.2, клиентская программа конвертирует коды нажатия клавиши и управляющих последовательностей команд пользователяского терминала в формат сетевого виртуального терминала и посыпает их серверу. Серверная программа преобразует поступившие данные и команды из формата NVT в формат, специфичный для удаленной системы. Возвращая данные, удаленный сервер переводит их из формата удаленной машины в формат NVT, а локальный клиент переводит их из формата NVT в формат локальной машины.

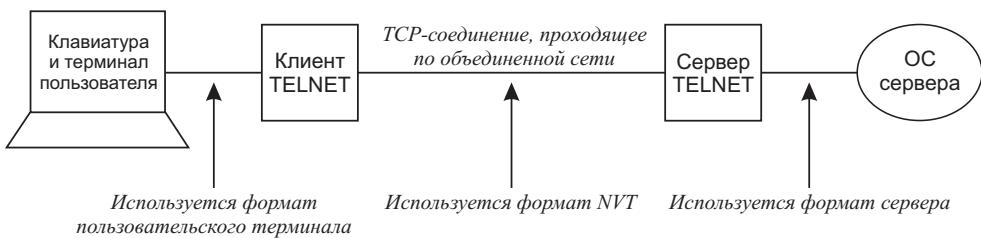


Рис. 25.2. Использование формата сетевого виртуального терминала (NVT) в протоколе TELNET

Формат NVT довольно прост. Во всех соединениях используется 8-битовый формат символов. Сразу после инициализации в системе NVT используется стандартное 7-битовое представление данных, как принято в кодировке USASCII. При этом байты, у которых старший бит равен единице, зарезервированы для передачи управляющих последовательностей команд. В набор символов USASCII включено 95 символов, соответствующих печатным знакам (буквы, цифры и знаки препинания), и 33 управляющих кода. Коды всех печатных знаков соответствуют кодам символов стандарта USASCII. Описание управляющих символов стандарта NVT приведено в табл. 25.1<sup>2</sup>. В стандарте TELNET не оговаривается положение символов табуляции.

**Таблица 25.1. Управляющие коды USASCII, используемые в стандарте NVT протокола TELNET**

Управляющий ASCII-код	Десятичное значение	Описание
NUL	0	Пустая операция (никак не влияет на вывод)
BEL	7	Звуковой/визуальный сигнал (курсор не перемещается)
BS	8	Перемещение курсора влево на одну символьную позицию
HT	9	Перемещение курсора вправо до следующей позиции горизонтального символа табуляции
LF	10	Перемещение курсора вниз (вертикально) на одну строку
VT	11	Перемещение курсора вниз до следующей позиции вертикального символа табуляции

<sup>2</sup> Коды управляющих символов в формате NVT соответствуют управляющим ASCII-кодам.

<b>Управляющий ASCII-код</b>	<b>Десятичное значение</b>	<b>Описание</b>
FF	12	Перемещение курсора в первую позицию первой строки следующей страницы
CR	13	Перемещение курсора к левому краю текущей строки
<i>Другой управляющий символ</i>	—	Пустая операция (никак не влияет на вывод)

Кроме интерпретации управляющих символов, приведенной в табл. 25.1, в стандарте NVT определено, что признаком конца строки является двухсимвольная последовательность управляющих кодов **<CR+LF>**. Когда пользователь нажимает на локальном терминале клавишу, соответствующую концу строки (например, **<ENTER>** или **<RETURN>**), вместо ее локального кода клиент TELNET должен передать по сети последовательность управляющих кодов **<CR+LF>**. Сервер TELNET, в свою очередь, преобразует последовательность управляющих кодов **<CR+LF>** в код конца строки, принятый на удаленной машине.

## 25.5. Передача команд управления удаленным устройством

Выше было сказано, что в большинстве операционных систем предусмотрен механизм, благодаря которому пользователи могут завершать выполнение программ. Обычно в локальной операционной системе такой механизм связан с нажатием специальной клавиши или комбинации клавиш. Например, во многих системах UNIX код, получаемый в результате нажатия клавиш **<CONTROL+C>**, зарезервирован для команды прерывания (если, конечно, пользователь не определяет иную комбинацию клавиш). Нажатие клавиш **<CONTROL+C>** служит для системы UNIX сигналом к принудительному завершению выполнения текущей программы. При этом самой прикладной программе управляющий код **<CONTROL+C>** не передается в качестве входных данных. Для выполнения других управляющих функций в операционной системе может быть зарезервировано еще несколько кодов и соответствующих им комбинаций клавиш.

В формате NVT протокола TELNET функции управления определяются в виде специальных кодов, которые передаются от клиента к серверу. Концептуально формат NVT можно считать обычным терминалом, клавиатура которого может генерировать более 128 возможных символов. Представить это можно так, будто на клавиатуре пользователя имеется ряд дополнительных клавиш, соответствующих функциям, которые обычно используются для управления обработкой данных. Например, в стандарте NVT определена абстрактная клавиша прерывания (interrupt), при нажатии которой операционной системе посыпается сигнал для завершения работы программы. Список кодов управляющих функций терминала NVT приведен в табл. 25.2. Клиентская программа получает их вместе с другими данными, введенными пользователем, и передает программе сервера, где они должны интерпретироваться.

В большинстве клавиатур не предусмотрены дополнительные клавиши для ввода команд. Поэтому управляющие коды вводятся разными способами в зависимости от типа операционной системы или интерпретатора команд. Выше уже упоминалось о наиболее общей методике — сопоставлении отдельного ASCII-кода

с определенной управляющей функцией. Поэтому при поступлении одного из таких кодов операционная система выполнит соответствующее действие, не интерпретируя этот символ как вводимые данные. Разработчики стандарта NVT разнесли коды управляющих функций и обычного набора ASCII-символов по двум причинам. Во-первых, отдельное определение кодов управляющих функций обеспечивает протоколу TELNET большую гибкость. При этом становится возможным передавать между клиентом и сервером в одном потоке все возможные последовательности ASCII-символов, включая управляющие коды. Во-вторых, отделение управляющих кодов от обычных данных, позволяет клиентской программе однозначно реагировать на поступающие сигналы, поскольку теперь не имеет значения, какой режим передачи (обычных символов или управляющих кодов) был установлен ранее для канала связи.

**Таблица 25.2. Список кодов управляющих функций терминала NVT протокола TELNET**

Сигнал	Описание
IP (Interrupt Process)	Прервать процесс (завершает выполнение программы)
AO (Abort Output)	Прекратить вывод (очистить данные, находящиеся в выходном буфере)
AYT (Are You There)	Есть кто живой? (проверка работоспособности сервера)
EC (Erase Character)	Стереть символ (удалить предыдущий символ)
EL (Erase Line)	Стереть строку (удалить текущую строку)
SYNCH (Synchronize)	Синхронизация (в ответ на получение сигнала срочных данных очистить тракт прохождения данных вплоть до обнаружения метки данных; команды при этом интерпретируются)
BRK (Break)	Принудительный останов (в результате нажатия клавиши <BREAK> или получения сигнала "Внимание")

При передаче управляющих кодов через TCP-соединение по протоколу TELNET они кодируются с помощью специальных *управляющих последовательностей* (*escape sequence*). Управляющая последовательность представляет собой специальный октет, содержащий соответствующее зарезервированное значение, за которым следует октет управляющего кода. В протоколе TELNET этот зарезервированный октет, с которого начинается управляющая последовательность, называют признаком *интерпретации команды* (*interpret as command*, или IAC). Список возможных команд и соответствующие им управляющие коды приведены в табл. 25.3. Коды интерпретируются как управляющие только в том случае, если перед ними находится символ IAC. Если нужно передать сам символ IAC как данные, то его следует поместить после другого символа IAC (т.е. два символа IAC, следующие друг за другом, интерпретируются как один символ IAC данных).

Как следует из табл. 25.3, каждому управляющему коду, генерируемому командными клавишами клавиатуры терминала NVT, поставлена в соответствие отдельная команда. Например, чтобы сервер прервал выполнение программы, клиент должен послать двухоктетную последовательность IAC+IP (коды 255 и 244). Несколько дополнительных команд позволяют клиенту и серверу согласовать используемые параметры, а также выполнить синхронизацию потока данных.

---

**Таблица 25.3. Управляющие коды протокола TELNET и их описание**

---

<b>Команда</b>	<b>Десятичный код</b>	<b>Описание</b>
IAC	255	Признак интерпретации следующего октета как команды. Если нужно передать символ IAC как данные, его следует поместить после еще одного символа IAC, т.е. передать двухоктетную последовательность IAC-IAC
DON'T	254	Отклонение запроса на выполнение указанной опции
DO	253	Подтверждение разрешения на выполнение указанной опции
WON'T	252	Отказ от выполнения указанной опции
WILL	251	Соглашение на выполнение указанной опции
SB	250	Начало процесса согласования параметров
GA	249	Сигнал "Продолжить работу"
EL	248	Сигнал "Очистить строку"
EC	247	Сигнал "Удалить символ"
AYT	246	Сигнал "Есть кто живой?"
AO	245	Сигнал "Прекращение вывода"
IP	244	Сигнал "Прерывание процесса"
BRK	243	Сигнал "Принудительный останов"
DMARK	242	Часть потока данных команды SYNCH (всегда сопровождается уведомлением с помощью признака срочных данных протокола TCP)
NOP	241	Пустая операция
SE	240	Конец процедуры согласования параметров
EOR	239	Конец записи

---

## 25.6. Привлечение внимания сервера к управляющим кодам

При отсылке управляющих кодов в одном потоке с обычными данными не всегда можно гарантировать, что сервер вовремя реагирует на них и что желаемые результаты будут достигнуты. Чтобы разобраться почему так происходит, рассмотрим ситуацию, когда пользователь посыпает на сервер управляющий код, соответствующий функции *прерывания процесса*. Необходимость использования этой функции возникает, как правило, только в том случае, если запущенная на удаленной машине программа некорректно работает (например, вследствие зацикливания перестала читать входные данные и ничего не выводит на экран) и пользователь хочет, чтобы сервер завершил ее выполнение. К сожалению, если пользовательская программа, запущенная на удаленном сервере, перестает читать входные данные, то буфер операционной системы в конечном счете переполняется и сервер не выводит данные на псевдотерминал. Если это произойдет, сервер прекратит считывание данных, поступающих по TCP-соединению, в результате чего переполнятся входные буфера протокола TCP. Как следствие, программа протокола TCP, запущенная на машине сервера, начнет анонсировать окна с нулевым размером, и поток данных через TCP-соединение прекратится.

В подобной ситуации сигнал прерывания, посланный пользователем серверному приложению, никогда не достигнет цели. Технически клиент может сформировать управляющую последовательность IAC+IP и поставить ее в очередь на отправку по этому TCP-соединению. Однако сервер не сможет ее прочитать, поскольку из-за переполнения входных буферов программа протокола TCP машины сервера запретила пересылку данных по этому TCP-соединению (анонсировала окно нулевого размера). Суть сказанного выше можно сформулировать так.

*Для работы протокола TELNET нельзя использовать только одно стандартное потоковое соединение, по которому передаются данные и команды управления между клиентом и сервером. Все дело в том, что некорректно работающая прикладная программа, запущенная на удаленном сервере, может непреднамеренно заблокировать поток данных. В результате пользователь локального компьютера потеряет над ней контроль.*

Чтобы решить эту проблему, в протоколе TELNET используется механизм уведомления сервера о наличии срочных данных. Он реализован с помощью механизма отправки *срочных данных* (*urgent data*) протокола TCP. Помещая управляющий код в поток данных, программа протокола TELNET посыпает также команду *синхронизации* (*SYNCH*). После этого клиентская программа TELNET помещает в поток специальный зарезервированный октет, называемый *меткой данных* (*data mark*, или DMARK), и заставляет программу протокола TCP послать сигнал серверу, путем отсылки сегмента, в заголовке которого установлен бит *срочных данных*. Такие сегменты не подчиняются правилам управления потоком данных, поэтому сразу достигают сервера. Получив сигнал о наличии срочных данных, сервер должен прочитать данные из потока и аннулировать их, пока не будет достигнута метка данных. Как только сервер столкнется с меткой данных, он возвращается к нормальной работе.

## 25.7. Параметры протокола TELNET

В упрощенном описании работы протокола TELNET, приведенном в этой главе, был опущен один из наиболее сложных моментов — согласование параметров, позволяющее клиенту и серверу динамически переконфигурировать параметры соединения. Например, выше было сказано, что обычно через поток данных передаются 7-битовые символы данных. Восьмой бит устанавливается только в случае пересылки управляющей информации, например команды *прерывания процесса*. Однако в протоколе TELNET предусмотрен параметр, позволяющий клиенту и серверу передавать 8-битовые данные. Если в потоке таких данных встретится зарезервированный октет IAC, то его нужно продублировать (т.е. управляющая последовательность IAC+IAC интерпретируется как один символ данных, значение которого равно IAC). Прежде чем передавать 8-битовые символы, клиент и сервер должны договориться между собой и одобрить режим 8-битовой передачи.

Диапазон параметров протокола TELNET очень широк. Изменение некоторых из них приводит к глобальным последствиям; изменение значений большинства параметров влияет лишь на отдельные детали работы клиентской и серверной программы. Например, первоначальная версия протокола TELNET была создана для работы в полудуплексном режиме. При этом, чтобы удаленный компьютер выводил на экран данные, клиентская программа должна была отослать ему специальную команду на продолжение выполнения (GA). Поэтому один из параметров протокола TELNET определяет, в каком из режимов (дуплексном или

полудуплексном) будет функционировать соединение. Еще один параметр позволяет серверу, запущенному на удаленной машине, определить тип клиентского терминала. Тип терминала важен для тех программ, которые выполняют позиционирование курсора с помощью специальных управляющих последовательностей, например для полноэкранного текстового редактора, запущенного на удаленной машине.

Список некоторых часто используемых параметров протокола TELNET приведен в табл. 25.4.

**Таблица 25.4. Часто используемые параметры протокола TELNET**

Название	Код	Документ RFC	Описание
Transmit Binary (бинарная передача)	0	856	Переход на 8-битовый режим передачи бинарных данных
Echo (эхо)	1	857	Предписывает удаленной программе отправлять обратно получаемые ею данные
Suppress-GA (подавление сигнала GA)	3	858	Подавляет отправку сигнала на продолжение работы после блока данных
Status (состояние)	5	859	Запрос значения TELNET-параметра на удаленном сервере
Timing-Mark (временная метка)	6	860	Запрос на помещение временной метки в возвращаемый поток данных (используется для синхронизации приемного и передающего конца соединения)
Terminal-Type (тип терминала)	24	884	Обмен информацией о типе и модели используемого терминала (позволяет прикладным программам использовать управляющие последовательности для выполнения определенных действий с терминалом пользователя, например позиционировать курсор в заданную точку экрана)
End-of-Record (конец записи)	25	885	В конец блока данных помещается специальный код конца записи — EOR
Linemode (строковый режим)	34	1116	Переход в режим локального редактирования. При этом на сервер отправляются целые строки, а не отдельные символы

## 25.8. Согласование параметров протокола TELNET

В протоколе TELNET используется необычный способ согласования параметров. Поскольку иногда серверу нужно задать значение некоторого параметра, то протокол разработан так, чтобы запросы можно было делать с любого конца соединения. Поэтому протокол считается *симметричным* по отношению к обработке параметров. Приемная сторона отвечает на запрос либо положительно (принимает его), либо отрицательно (отклоняет). В терминологии протокола TELNET запрос WILL X означает следующее: “Можно ли использовать параметр X”. Ответ на этот запрос может быть либо DO X либо DON’T X, что означает: “Разрешаю использовать параметр X” или “Не разрешаю использовать параметр X”. В данном случае о симметрии можно говорить, поскольку ответ DO X предписывает получателю начать использование параметра X, а запрос WILL X

или  $WON'T X$  свидетельствует о том, что отправитель запроса собирается или не собирается использовать параметр  $X$ <sup>3</sup>.

Еще один интересный момент согласования параметров состоит в том, что от обеих сторон соединения требуется, чтобы они вначале запустили у себя программу реализации простого терминала NVT (то есть без использования параметров). Если одна из сторон попытается согласовать значение параметра, который не поддерживается другой стороной, принимающая сторона может просто отклонить этот запрос. В результате между собой могут взаимодействовать как новые, более сложные версии клиентов и серверов TELNET (то есть программы, которые поддерживают большее количество параметров), так и старые, более простые. Если и клиент и сервер поддерживают новые параметры, они могут повысить эффективность взаимодействия. Если же новые параметры ими не поддерживаются, процесс взаимодействия между ними будет менее эффективным, но все-таки возможным. Все сказанное выше можно подытожить так.

*В протоколе TELNET используется симметричный механизм согласования параметров. Он позволяет клиентам и серверам на ходу изменять значения параметров, управляющих процессом их взаимодействия. Поскольку все программы поддержки протокола TELNET могут работать с простым терминалом NVT, клиенты и серверы могут взаимодействовать даже в том случае, если один из них поддерживает некоторые параметры, а другой — нет.*

## 25.9. Служба rlogin (BSD UNIX)

В поставку операционных систем, относящихся к клону BSD UNIX, включена служба удаленного входа в систему, `rlogin`, поддерживающая надежные узлы сети, с которыми установлены доверительные отношения. Она позволяет системным администраторам назначать группу машин, имеющих общие имена пользователей и средства защиты от несанкционированного доступа к файлам, а также устанавливать соответствия между регистрационными именами пользователей. Пользователи могут управлять доступом к своим учетным записям путем авторизации процедуры удаленного входа в систему, основанной на имени удаленного узла и имени учетной записи удаленного пользователя. Таким образом, пользователь может иметь учетную запись  $X$  на одной машине и  $Y$  — на другой, и при этом он может удаленно заходить с одной машины на другую, не вводя пароль каждый раз.

Наличие системы автоматической авторизации позволяет применить возможности удаленной регистрации как в программах общего назначения, так и для интерактивного общения между пользователями. Один из вариантов команды `rlogin` — `rsh` вызывает интерпретатор команд на удаленной машине UNIX и передает ему параметры командной строки, пропуская при этом этап входа в систему. Формат вызова команды `rsh` такой:

```
rsh имя-машины команда
```

Таким образом, после ввода команды

```
rsh merlin ps
```

на любой из машин факультета информатики университета Пердью, на машине `merlin` будет выполнена команда `ps`. При этом устройства стандартного ввода

<sup>3</sup> В данном случае система может зациклиться, если каждая из сторон “подумает”, что сигнал подтверждения от другой стороны является запросом. Поэтому в протоколе определено, что на запрос параметра, который уже используется, не требуется подтверждения.

и вывода системы UNIX связываются через сеть с клавиатурой и дисплеем пользователя, запустившего команду `rsh`. Пользователь увидит результат работы команды `ps` так, будто он ввел ее непосредственно на машине `merlin`, пройдя стандартную процедуру регистрации. Поскольку можно сделать так, чтобы команда `rsh` запускала команды или программы на удаленной машине без ввода пароля, она может использоваться и в прикладных программах, а не только вводиться с клавиатуры.

Службы типа `rlogin`, работающие как в локальных, так и в удаленных вычислительных средах, взаимодействуют между собой на более высоком уровне, чем стандартные службы удаленного входа в систему типа TELNET. Например, в команде `rlogin` поддерживаются три стандартных устройства системы UNIX: *стандартного ввода*, *стандартного вывода*, и *стандартной ошибки*, которые посредством протокола TCP подключены к удаленной машине. Таким образом, после ввода команды

```
rsh merlin ps > имя-файла
```

выходные данные, полученные в результате выполнения команды `ps` на машине `merlin`, будут перенаправлены<sup>4</sup> в локальный файл, заданный параметром *имя-файла*. Команда `rlogin` также поддерживает функции управления терминалом — обрабатывает символы управления потоком данных (обычно `<CONTROL+S>` и `<CONTROL+Q>`). Ввод этих символов вызывает немедленную остановку или возобновление процесса вывода на терминал, без задержки, требуемой для отсылки символов управления по сети на удаленный сетевой узел. И наконец, команда `rlogin` экспортирует часть переменных среды пользователя на удаленную машину, в том числе такую информацию, как тип терминала пользователя (т.е. значение переменной окружения `TERM`). В результате сеанс работы с удаленной системой почти ничем не отличается от сеанса работы с локальной машиной.

## 25.10. Резюме

Большая часть функциональных возможностей семейства протоколов TCP/IP связана с разнообразием высокоуровневых служб, реализованных в виде прикладных программ. В высокоуровневых протоколах удаленного входа в систему используются базовые службы семейства протоколов TCP/IP: служба негарантированной доставки дейтаграмм и надежная потоковая транспортная служба. Высокоуровневые службы обычно реализуются на основе модели типа клиент/сервер, в которой серверы прослушивают стандартные порты протокола. Поэтому клиентская программа всегда “знает”, как подключиться к нужному серверу.

В этой главе были рассмотрены две системы удаленного входа: TELNET — стандарт объединенной сети TCP/IP и `rlogin` — популярный протокол, используемый в системах UNIX, являющихся клонами BSD UNIX. Протокол TELNET обеспечивает основной набор услуг. Он позволяет клиентской программе передавать на сервер как данные, так и команды, например команду *прерывания процесса*. Кроме того, он позволяет клиенту и серверу согласовывать значения многих параметров. В отличие от TELNET, протокол `rlogin` предоставляет системным администраторам и пользователям большую гибкость, поскольку позволяет устанавливать соответствия между регистрационными именами пользователей на нескольких машинах, однако он не так распространен, как TELNET.

---

<sup>4</sup> Символ `>` (“больше чем”) обычно используется в командных интерпретаторах системы UNIX для переключения выходных потоков в файл или на другое устройство.

## Материал для дальнейшего изучения

Для удаленного входа в систему было предложено довольно много высоковысоконивневых протоколов, но обычно используются только некоторые из них. В работе Эджа (Edge) [46] сравниваются протоколы сквозной передачи (end-to-end) данных с методом последовательной передачи от узла к узлу. Зальцер (Saltzer), Рид (Reed) и Кларк (Clark) доказали в работе [115] важность сквозного контроля и обнаружения ошибок при передаче данных, выполняемых в протоколах высокого уровня.

Спецификация протокола удаленного входа в систему TELNET приведена Постелом (Postel) в [RFC 854]. За ним последовало более трех десятков документов RFC, в которых описывались параметры протокола TELNET, его слабые места, результаты экспериментов и предлагались изменения. Более ранний стандарт протокола TELNET описан Постелом в [RFC 764]. Спецификация параметров протокола TELNET и методы их согласования описаны Постелом и Рейнольдсом (Reynolds) в [RFC 855]. Длинный список документов RFC с описанием различных параметров протокола TELNET можно найти в: [RFC 856, 857, 858, 859, 860, 861, 884, 885, 1041, 1091, 1096, 1097, 1184, 1372, 1416, и 1572]. Для доступа к компьютерам фирмы IBM, на которых работает операционная система VM/CMS, используется программа tn3270. Она является аналогом клиента TELNET для мейнфреймов, механизм работы которого описан в [RFC 1576], [RFC 1646] и [RFC 1647]. Параметр протокола TELNET, разрешающий связь с терминалом IBM 3270, описан Рехтером (Rekhter) в [RFC 1041].

## Упражнения

- 25.1. Поэкспериментируйте с программами TELNET и rlogin. Каковы основные различия между ними?
- 25.2. Несмотря на большой объем публикаций о протоколе TELNET, можно доказать, что этот протокол все еще не до конца определен. Поэкспериментируйте с программой TELNET: подключитесь к машине A и запустите на ней еще одну программу TELNET, с помощью которой подключитесь к машине B. Проверьте, корректно ли обрабатываются символы перевода строки и возврата каретки комбинацией двух программ.
- 25.3. Что такое *дистанционный вызов процедур* (*remote procedure call*, или RPC)?
- 25.4. Народная мудрость гласит, что операционные системы приходят и уходят, а протоколы остаются навсегда. Проверьте эту аксиому на примере вашего сетевого центра и посмотрите, что там изменилось чаще — операционные системы или протоколы связи.
- 25.5. Напишите клиентскую программу TELNET.
- 25.6. Попробуйте с помощью клиентской программы TELNET подключить клавиатуру и монитор к одному из портов echo или chargen вашего локального компьютера и посмотрите, что произойдет.
- 25.7. Прочитайте стандарт протокола TELNET и выясните, как работает команда синхронизации SYNCH.
- 25.8. В протоколе TELNET используется механизм пересылки *срочных данных* протокола TCP для того, чтобы привлечь внимание удаленной операционной системы к посланным ей управляющим кодам.

Прочитайте стандарт и выясните, какие команды обрабатывает удаленный сервер при чтении входного потока данных.

- 25.9.** Как можно зациклить систему согласования параметров симметричными запросами типа DO/DON'T и WILL/WON'T при условии, что другая сторона будет *всегда* присыпать ответ на запрос?
- 25.10.** В документе [RFC 854] (спецификация протокола TELNET) содержится ровно 854 строки текста. Как вы думаете, есть ли в этом какой-то магический смысл?

# 26

## *Приложения: передача файлов и удаленный доступ к ним (FTP, TFTP, NFS)*

### **26.1. Введение**

В этой главе продолжается изучение протоколов уровня приложений. В ней рассматриваются средства удаленного доступа к файлам и протоколы их передачи, которые являются частью семейства протоколов TCP/IP. Мы опишем их структуру, а также приведем пример типичного интерфейса пользователя. Будет показано, что в наиболее широко используемом протоколе передачи файлов (FTP) применяется надежный транспортный протокол TCP, описанный в главе 13, “Надежная потоковая транспортная служба (TCP)”, и протокол TELNET, описанный в предыдущей главе.

### **26.2. Удаленный доступ к файлам и их передача**

Многие сетевые операционные системы позволяют клиентским компьютерам получать доступ к файлам, расположенным на удаленных машинах. Проектировщики проанализировали несколько вариантов удаленного доступа к файлам. Оказалось, что каждый из вариантов оптимизирован для выполнения определенного набора задач. Например, в некоторых предложенных решениях удаленный доступ к файлам используется для снижения общей стоимости системы. В подобных случаях система состоит из недорогих компьютеров, не имеющих локальных дисковых накопителей. При этом в качестве запоминающего устройства для хранения информации используется централизованный *файловый сервер*. В качестве бездисковых машин могут применяться дешевые карманные устройства, используемые для выполнения несложных операций типа выписки счетов. Такие машины связываются с файловым сервером по быстродействующей беспроводной локальной сети.

Иногда удаленные запоминающие устройства применяются для архивирования данных. При этом пользователи работают на обычных персональных компьютерах, оснащенных локальными дисковыми накопителями. Копии отдельных файлов пользователя или целых дисков периодически посыпаются по сети на устройство архивирования, где они сохраняются на случай непредвиденной потери.

Часто особое внимание уделяется возможности коллективного доступа к данным со стороны нескольких программ, пользователей, или даже сетевых центров. Например, организация хочет создать единую интерактивную базу данных, в которой информация о просроченных счетах была бы доступна для всех подразделений этой организации.

### **26.3. Коллективный доступ в интерактивном режиме**

Различают две формы коллективного использования файлов: *интерактивный доступ* и *копирование целого файла*. Коллективный интерактивный доступ позволяет множеству программ одновременно обращаться к одному файлу. Изменения в файле немедленно вступают в силу и доступны всем программам, имеющим к нему доступ. Копирование целого файла означает, что всякий раз, когда программа хочет получить доступ к файлу, она получает его локальную копию. Копирование используется в основном для данных, предназначенных только для чтения, но если файл надо изменить, программа делает изменения в локальной копии и передает копию измененного файла на сервер.

Многие пользователи считают, что коллективное использование данных можно реализовать только в виде системы управления базой данных, работающей в виде сервера и позволяющей пользователям (клиентам) подключаться к ней из удаленных узлов сети. Тем не менее коллективное использование файла возможно, просто обычно его труднее реализовать, но использовать гораздо легче. Например, при работе с файловой системой, обеспечивающей возможность коллективного интерактивного доступа для удаленных пользователей, не нужно вызывать специальную клиентскую программу, как при работе с системой баз данных. Доступ к файлам удаленных пользователей осуществляется средствами операционной системы точно так же, как и для локальных пользователей. Пользователь может запустить любую прикладную программу, используя удаленный файл для ввода или вывода данных. При этом говорят, что удаленный файл *интегрирован* с локальными файлами, и что средства файловой системы обеспечивают *прозрачный доступ* для коллективного использования файлов.

Преимущество прозрачного доступа очевидно: обращение к удаленному файлу происходит без внесения каких-либо изменений в прикладные программы. При этом пользователи могут получать доступ как к локальным, так и удаленным файлам, что позволяет им обрабатывать общие данные. Недостатки прозрачного доступа менее очевидны. Один из них — неадекватность ожидаемых результатов. В качестве примера рассмотрим прикладную программу, использующую и локальные, и удаленные файлы. Если сеть или удаленная машина не функционируют, прикладная программа не сможет продолжить работу, даже если машина пользователя функционирует normally. Кроме того, удаленная машина может быть перегружена, либо в сети может образоваться затор. Эти факторы замедляют выполнение прикладной программы; интервал времени, отпущенный сетевыми протоколами на передачу данных, истекает. В результате во время выполнения прикладной программы возникают ошибки, обработка которых не была предусмотрена программистом при ее создании. Пользователи считают такие прикладные программы ненадежными.

Несмотря на все преимущества, практическая реализация интегрированного прозрачного доступа к файлам может быть затруднена. В разнородной вычислительной среде чаще всего возникает проблема сопоставления имен файлов одного компьютера с именами файлов другого компьютера. Механизм удаленного доступа к файлам операционной системы должен также учитывать атрибуты владельца файла, средства его авторизации и защиты от несанкционированного доступа, которые не должны противоречить принятым в данной системе соглашениям. И наконец, реализовать все виды файловых операций для всех типов компьютеров может оказаться затруднительно или вообще невозможно из-за особенностей конкретной файловой системы и способа хранения файлов.

## **26.4. Коллективный доступ к файлу посредством его передачи**

Альтернативой интегрированному прозрачному интерактивному доступу к файлам является *передача файла* клиенту на обработку. Доступ к удаленным данным посредством передачи файла представляет собой двухступенчатый процесс: сначала пользователь получает локальную копию файла, а затем работает с этой копией. В большинстве случаев механизмы передачи работают независимо от локальной файловой системы (то есть, они не интегрированы с ней). Для передачи файлов пользователь должен вызвать специальную клиентскую программу. Вызывая клиентскую программу, пользователь указывает адрес удаленного компьютера, на котором находится нужный файл и при необходимости — сведения для выполнения авторизации (например, имя пользователя и пароль). Клиентская программа подключается к удаленному серверу и запрашивает копию файла. Когда процесс передачи заканчивается, пользователь завершает работу клиентской программы и использует подручные программы для чтения или модификации локальной копии файла. Одно из преимуществ в копировании целого файла заключается в высокой эффективности этой операции. Полученную копию удаленного файла программа может обрабатывать с высокой скоростью. Поэтому при копировании файла многие вычисления выполняются быстрее, чем при удаленном доступе к нему.

Как и в случае интерактивного коллективного доступа к файлу, передача файла между разнородными машинами может быть затруднена. При этом клиент и сервер должны согласовать способ авторизации доступа, методы обработки атрибутов, связанных с владельцем файла, защиту доступа, а также форматы данных. Последний момент особенно важен, поскольку он может сделать обратные преобразования файла невозможными. Чтобы понять, почему так происходит, рассмотрим процесс копирования файла между двумя машинами — *A* и *B*, в которых используются различные представления чисел с плавающей запятой, а также разные способы кодирования текстовых файлов. Программисты знают, что невозможно преобразовать без потери точности данные из формата с плавающей запятой, используемого на одной машине, в формат, используемый на другой машине. То же самое может произойти и с текстовыми файлами. Предположим, что на машине *A* текстовые файлы состоят из строк переменной длины, а на машине *B* — из строк фиксированной длины. При передаче файла от машины *A* к машине *B* к каждой строке файла будет добавлено определенное количество пустых символов (обычно пробелов) для выравнивания общей длины строки до заданного размера. В результате переданная копия будет отличаться от оригинала. При обратной передаче файла от машины *B* к машине *A* автоматическое удаление пустых символов в конце каждой строки также приведет к тому, что полученная копия файла будет отличаться от оригинала, если в конце некоторых строк пустые символы должны находиться по смыслу.

Различия в представлении данных и способах их обработки зависят от конкретных взаимодействующих компьютерных систем. Кроме того, мы видели, что не все различия в представлении данных могут быть учтены, поэтому при их переводе из одного представления в другое часть информации может быть потеряна. Хотя нет необходимости вникать во все возможные различия в представлении, но все же для объяснения особенностей протоколов передачи файлов следует напомнить, что семейство протоколов TCP/IP предназначено для работы в разнородной среде.

## 26.5. FTP — основной протокол передачи файлов семейства TCP/IP

Передача файлов — одно из основных применений семейства протоколов TCP/IP, и на ее долю приходится львиная доля сетевого трафика. Стандартные протоколы передачи файлов существовали еще в сети ARPANET, т.е. до появления семейства протоколов TCP/IP. Эти ранние версии программного обеспечения для передачи файлов развились в текущий стандарт, называемый *протоколом передачи файлов* (*File Transfer Protocol*, или *FTP*).

## 26.6. Особенности протокола FTP

На первый взгляд может показаться, что реализация механизма передачи файлов на основе надежного сквозного (end-to-end) транспортного протокола типа TCP, является тривиальной задачей. Однако, как было показано в предыдущих разделах, дело усложняют существующие различия в системах авторизации, присваивания имен и представления данных среди разнородных машин. Протокол FTP, кроме собственно функции передачи файлов, обеспечивает ряд других возможностей, которые перечислены ниже.

- *Интерактивный доступ.* Хотя протокол FTP был разработан для использования программами, в большинстве его реализаций предусмотрен интерактивный интерфейс, позволяющий людям легко взаимодействовать с удаленными серверами. Например, пользователь может запросить список всех файлов, содержащихся в некотором каталоге на удаленной машине. Кроме того, клиентская программа обычно отвечает на введенную пользователем команду `help` и выводит краткую справку о командах, которые могут быть вызваны.
- *Спецификация формата (представления данных).* Протокол FTP позволяет клиенту определять тип и формат сохраняемых данных. Например, пользователь может указать, содержит ли файл текст или двоичные данные и какой набор символов используется в текстовом файле — ASCII или EBCDIC.
- *Контроль аутентификации.* В протоколе FTP предусмотрена аутентификация клиентов с помощью имени пользователя и пароля. Прежде чем послать запрос на передачу файла, клиентская программа должна сообщить серверу имя учетной записи пользователя и пароль. Клиентам, которые не могут сообщить правильное имя пользователя и пароль, сервер отказывает в доступе.

## 26.7. Модель FTP-процесса

Подобно другим серверам, большинство реализаций FTP-серверов поддерживает одновременную работу с несколькими клиентами. Для подключения к серверу клиенты используют протокол TCP. Как описано в главе 21, “Модель взаимодействия клиент/сервер”, главная программа FTP-сервера ожидает запросы на соединение от клиентов и создает подчиненный процесс для обработки каждого соединения. Однако, в отличие от большинства серверов, подчиненный процесс не выполняет самостоятельно все действия необходимые, для передачи файла. Он принимает от клиента и обрабатывает команды управления FTP-сеансом, а для передачи данных создается один или несколько дополнительных процессов. С помощью команд пользователь указывает серверу, какой файл надо передать. Передача данных осуществляется в отдельном процессе через отдельное соединение. При этом в качестве транспортного используется протокол TCP.

Обычно для передачи данных и в клиентской, и в серверной программе создаются отдельные процессы. Структура этих процессов зависит от используемых операционных систем, а на рис. 26.1 продемонстрирована идея.

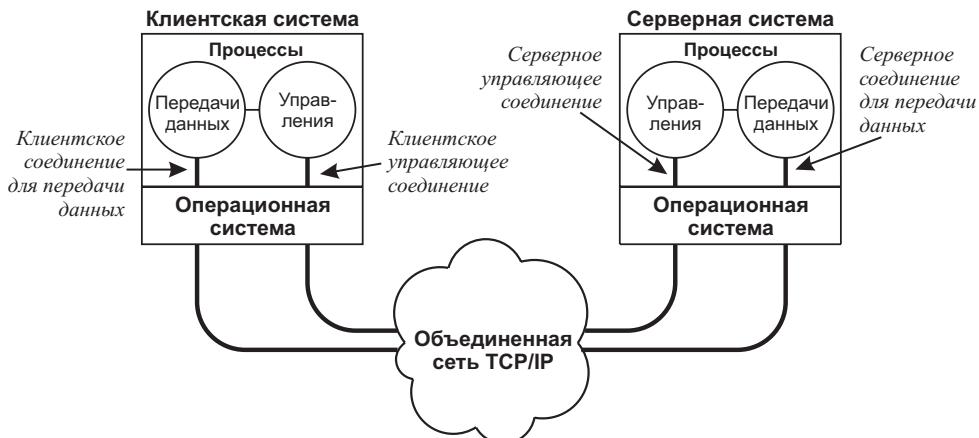


Рис. 26.1. Иллюстрация процесса обмена данными между клиентом и сервером FTP

Как показано на рис. 26.1, для подключения клиентского управляющего процесса к серверному управляющему процессу используется самостоятельное TCP-соединение, а обмен данными осуществляется через другое TCP-соединение. Чаще всего управляющие процессы и TCP-соединение между ними остаются в активном состоянии на протяжении всего сеанса работы с FTP. При этом для передачи каждого нового файла по протоколу FTP создается отдельное TCP-соединение. Во многих реализациях протокола FTP новая пара процессов передачи данных и новое TCP-соединение для них создается всякий раз, когда серверу надо отослать информацию клиенту. Эту идею можно подытожить так.

*Процессы передачи данных и используемые ими соединения могут создаваться динамически в процессе FTP-сеанса как только в них возникает необходимость. Однако управляющее соединение существует на протяжении всего сеанса. Когда управляющее соединение закрывается, сеанс работы с FTP завершается. При этом программы, запущенные на обоих концах соединения, завершают все процессы передачи данных.*

Конечно, если на компьютере установлена операционная система, не поддерживающая многозадачный режим работы, то структура клиента может быть менее сложной. В таких случаях часто жертвуют универсальностью и используют одну прикладную программу и для выполнения управляющих функций и для передачи данных. Однако в протоколе FTP все равно требуется, чтобы такие клиентские программы использовали несколько TCP-соединений: одно для управления, а другое (или другие) для передачи данных.

## 26.8. Назначение номеров портов протокола TCP

При установке начального соединения с FTP-сервером клиентская программа использует случайный, локально присвоенный номер порта протокола, однако обращение к серверу осуществляется по стандартному номеру порта 21. Как было показано в главе 21, “Модель взаимодействия клиент/сервер”, сервер, использующий только один порт протокола, может обрабатывать соединения от

многих клиентов, потому что для идентификации соединения в протоколе TCP используются обе его конечные точки. Возникает вопрос: какие номера портов протоколов используют управляющие процессы при создании нового TCP-соединения для передачи данных? Очевидно, что они не могут использовать ту же самую пару номеров портов, которая используется в управляющем соединении. Вместо этого, клиентской программе выделяется номер свободного порта, который и будет использоваться для создания TCP-соединения с FTP-сервером для передачи данных. Процесс передачи данных, запущенный на машине сервера, использует при этом стандартный порт, зарезервированный для передачи данных по протоколу FTP (20). Чтобы гарантировать подключение процесса передачи данных на сервере к соответствующему процессу передачи данных на машине клиента, сервер не должен принимать запросы на соединение от любого процесса. Поэтому, когда программа сервера посылает клиенту активный запрос на открытие нового TCP-соединения для передачи данных, она должна указать номера портов, которые будут использоваться на машине клиента и на машине сервера.

Теперь должно быть понятно, зачем в протоколе FTP используется два TCP-соединения. Управляющий процесс клиента получает от операционной системы номер свободного локального порта, который будет использоваться при передаче файла. После этого на машине клиента создается новый процесс, который переводится в состояние ожидания поступления данных по этому порту (говорят, что клиент начинает прослушивать локальный порт). Управляющий процесс клиента через свое TCP-соединение посылает запрос FTP-серверу на установку соединения по новому номеру порта и переходит в состояние ожидания, пока сервер не установит с этим портом TCP-соединение. Все сказанное выше можно подытожить так.

*Управляющее соединение используется в протоколе FTP не только для передачи пользовательских команд на сервер, но и позволяет управляющим процессам клиента и сервера скоординировать использование динамически выделяемых номеров портов протокола TCP и создание процессов передачи данных через эти порты.*

Какой формат данных должен использоваться в протоколе FTP при передаче данных через управляющее соединение? Хотя создатели протокола FTP могли бы изобрести для этого новую спецификацию, они не стали этого делать, а воспользовались спецификацией виртуального терминала протокола TELNET, описанной в главе 25, “Приложения: удаленный вход в систему (TELNET, rlogin)”. Однако они отказались от полной поддержки протокола TELNET. В протоколе FTP не предусмотрен режим согласования параметров, а используются только базовые команды терминала NVT. Таким образом, организация управляющего FTP-соединения гораздо проще организации стандартного TELNET-соединения. Несмотря на введенные ограничения использование стандартного протокола TELNET вместо задания новых управляющих последовательностей значительно упрощают реализацию протокола FTP.

## 26.9. Протокол FTP с точки зрения пользователя

С точки зрения пользователя сеанс работы с FTP-сервером происходит в интерактивном режиме. После запуска клиентская программа переходит в бесконечный цикл, в котором выполняются следующие действия. Сначала со стандартного устройства ввода считывается строка текста, затем выполняется ее синтаксический анализ с целью извлечения команды и ее параметров. После этого клиентская программа выполняет запрошеннную пользователем команду с указанными

параметрами. Например, чтобы начать сеанс работы с FTP-сервером в системе UNIX, пользователь вызывает команду `ftp`:

```
% ftp
```

При этом запускается программа локального клиента FTP, которая выводит пользователю специальное приглашение. Получив приглашение, пользователь может вводить команды, например `help`.

```
ftp> help  
Commands may be abbreviated. Commands are:  
(Допускается сокращение команд при вводе. Набор команд:)
```

!	cr	macdef	proxy	send
\$	delete	mdelete	sendport	status
account	debug	mdir	put	struct
append	dir	mget	pwd	sunique
ascii	disconnect	mkdir	quit	tenex
bell	form	mls	quote	trace
binary	get	mode	recv	type
bye	glob	mput	remotehelp	user
case	hash	nmap	rename	verbose
cd	help	ntrans	reset	?
cdup	lcd	open	rmdir	
close	ls	prompt	runique	

Чтобы получить больше информации о команде, пользователь должен ввести ее после команды `help`, как показано в следующих примерах (формат вывода соответствует команде `ftp`):

```
ftp> help ls  
ls           list contents of remote directory  
ftp> help cdup  
cdup        change remote working directory to parent directory  
ftp> help glob  
glob        toggle metacharacter expansion of local file names  
ftp> help bell  
bell        beep when command completed
```

Для выполнения команды пользователь должен ввести ее имя:

```
ftp> bell  
Bell mode on.
```

## 26.10. Пример анонимного FTP-сеанса

Средства безопасности протокола FTP позволяют закрыть доступ к серверу со стороны неавторизованных пользователей. Не зная имени и пароля, пользователь не сможет получить доступ к файлам. Однако иногда требуется, чтобы доступ к файлам, хранящимся на сервере, мог получить любой подключившийся к нему пользователь. Для обеспечения доступа к общим файлам в большинстве FTP-серверов реализован режим *анонимного* доступа. Это означает, что для доступа к файлам на сервере клиенту не нужны учетная запись и пароль. Вместо этого, при подключении к серверу клиент должен ввести в качестве имени пользователя строку `anonymous`, а в качестве пароля — `guest`. В результате анонимный клиент получает доступ только к общедоступным файлам<sup>1</sup>.

<sup>1</sup> Во многих системах UNIX ограничение доступа для анонимных клиентов происходит путем замены корневого каталога файловой системы на другой небольшой по размеру каталог (например, `/usr/ftp`).

Обычно для установки соединения с FTP-сервером и получения файла пользователь должен ввести несколько команд. Другие команды используются очень редко. Предположим, что кто-то поместил в подкаталог pub/comer FTP-сервера, запущенного на машине ftp.cs.purdue.edu, электронную копию этой книги и назвал архивный файл tcpbook.tar. Пользователь, зарегистрировавшийся на другом компьютере под именем usera, может получить копию этого файла, введя следующие команды.

```
% ftp ftp.cs.purdue.edu
Connected to lucan.cs.purdue.edu.
220 lucan.cs.purdue.edu FTP server (Version wu-2.4.2-VR16(1) ready.
Name (ftp.cs.purdue.edu:usera): anonymous
331 Guest login ok, send e-mail address as password.
Password: guest
230 Guest login ok, access restrictions apply.
ftp> get pub/comer/tcpbook.tar bookfile
200 PORT command okay.
150 Opening ASCII mode data connection for tcpbook.tar (9895469 bytes).
226 Transfer complete.
9895469 bytes received in 22.76 seconds (4.3e+02 Kbytes/s)
ftp> close
221 Goodbye.
ftp> quit
```

В этом примере пользователь вводит имя машины ftp.cs.purdue.edu в качестве параметра команды FTP. Клиентская программа автоматически открывает соединение и выводит приглашение на ввод имени пользователя. Чтобы начать анонимный сеанс работы, клиент должен ввести в качестве имени пользователя строку anonymous, а в качестве пароля — guest<sup>2</sup> (хотя в нашем примере показано, что FTP-программа выводит на экран пароль, который ввел пользователь, на самом деле пароль на экране не отображается).

После ввода имени пользователя и пароля клиент запрашивает копию файла, используя команду get. В нашем примере в команде get указано два параметра, которые определяют имя удаленного файла и имя его локальной копии. Удаленное имя файла — pub/comer/tcpbook.tar, а локальная копия будет помещена в файл bookfile. По окончании передачи, пользователь вводит команду close для закрытия соединения с сервером и команду quit, чтобы завершить работу клиентской программы.

Сообщения, которые выводит клиентская FTP-программа в ответ на ввод пользовательских команд, являются информационными. Они всегда начинаются с трехзначного кода, за которым следует текст. Большинство сообщений поступает непосредственно от сервера, а остальные генерируются клиентской программой. Например, сообщение, которое начинается с кода 220, пришло от сервера и содержит доменное имя машины, на которой запущен FTP-сервер. Статистические данные о количестве полученных байтов и средней скорости передачи выводит клиентская программа. Все сказанное выше можно подытожить так.

*Информационные сообщения и сообщения об ошибках, выводимые клиентской FTP-программой, начинаются с трехзначного кода, за которым следует текст. Этот код интерпретируется программой, а текстовое сообщение предназначено для пользователя.*

---

<sup>2</sup> На практике сервер выводит дополнительную информацию, в которой предлагает пользователю вместо пароля guest ввести свой адрес электронной почты.

В приведенном примере также показана особенность протокола FTP, описанная выше. Речь идет о создании новых TCP-соединений для передачи данных. Обратите внимание на сообщение о команде `PORT` в тексте листинга, выведенное клиентской программой. Оно информирует пользователя о том, что для создания нового TCP-соединения для передачи данных локальной операционной системой был выделен свободный номер порта TCP. Клиентская программа посыпает информацию о номере порта на FTP-сервер по управляющему соединению. Сервер использует этот номер порта для установки соединения с клиентом для передачи данных. После завершения передачи данных клиентский и серверный процессы закрывают соединение на каждой из сторон.

## 26.11. Протокол TFTP

Хотя FTP является универсальным протоколом для передачи файлов в семействе TCP/IP, он — наиболее сложный и трудный для программирования. Во многих приложениях не требуется поддержка полных функциональных возможностей протокола FTP; более того, такая сложность часто является для них непозволительной роскошью. Например, в протоколе FTP требуется, чтобы для передачи данных и клиенты, и серверы открывали несколько параллельных TCP-соединений. Однако, может оказаться, что некоторые операции трудно или вовсе невозможно выполнить на персональных компьютерах, не оснащенных современными развитыми операционными системами.

Поэтому в семействе протоколов TCP/IP предусмотрен второй протокол для передачи файлов, который позволяет реализовать простую и недорогую службу. Он называется *простейший протокол передачи файлов* (*Trivial File Transfer Protocol*, или *TFTP*) и предназначен для приложений, где не требуется сложное взаимодействие клиента и сервера. В протоколе TFTP предусмотрены только простые операции по пересылке файла без использования средств аутентификации. Поскольку протокол TFTP намного проще, чем FTP, для его программной реализации требуется небольшое количество кода.

Малый размер кода имеет огромное значение для реализации многих приложений. Например, изготовители бездисковых устройств обычно помещают программу поддержки протокола TFTP в постоянное запоминающее устройство (ПЗУ). Она вызывается при включении питания машины и позволяет получить файл с образом памяти для выполнения начальной загрузки компьютера. Эта программа, находящаяся в ПЗУ, называется *программой начальной загрузки* системы<sup>3</sup>. Преимущество протокола TFTP состоит в том, что он позволяет при написании программы начальной загрузки использовать тот же набор основных протоколов TCP/IP, которые будут использоваться операционной системой в процессе ее работы. Таким образом, компьютер может выполнить начальную загрузку с сервера, находящегося в другой физической сети.

В отличие от FTP, протоколу TFTP не нужна надежная потоковая транспортная служба. Для доставки пакетов он может использовать протокол UDP либо любой другой аналогичный протокол негарантированной доставки. При этом для гарантирования доставки данных в протоколе TFTP используется механизм повторной передачи по истечении заданного интервала времени (тайм-аута). Отправитель передает файл в виде блоков фиксированного размера (обычно 512 байт). При этом перед отправкой следующего блока он ждет сигнала подтверждения приема предыдущего блока. После успешного получения блока данных получатель посыпает отправителю уведомление.

<sup>3</sup> Подробно процесс начальной загрузки с помощью протокола DHCP рассмотрен в главе 23, “Начальная загрузка и автоконфигурация (BOOTP, DHCP)”.

**Печатай файл этой страницы отдельно**

данных, отправляющая сторона передает его повторно. Если после отправки сигнала подтверждения приема получающая сторона не получит в течение заданного времени следующий блок данных, она повторно передает сигнал подтверждения приема. Вовлечение обеих сторон в процесс повторной передачи данных гарантирует, что потеря одного пакета не приведет к сбою при передаче всех данных.

Хотя метод двухсторонней повторной передачи гарантирует высокую устойчивость к ошибкам, его использование может привести к большому количеству повторных передач. Эта проблема называется *ошибкой начинающего волшебника* (*Sorcerer's Apprentice Bug*). Она возникает, когда сигнал подтверждения приема для  $k$ -го пакета данных приходит после тайм-аута. Сначала отправитель повторяет передачу  $k$ -го пакета данных, на который получатель отправляет еще один сигнал подтверждения приема. В конечном счете оба сигнала подтверждения приема достигают сервера и каждый из них вызывает передачу  $(k+1)$ -го пакета данных. Получатель подтвердит получение обеих копий  $(k+1)$ -го пакета данных, а каждое из этих двух подтверждений заставит отправителя передать  $(k+2)$ -й пакет данных. Описанный эффект может также возникнуть, если по какой-либо причине в сети передачи данных возникнут дубли пакетов. Если это произойдет, каждый пакет данных будет передаваться дважды на протяжении всего времени взаимодействия клиента с сервером.

Несмотря на то что в протоколе TFTP предусмотрен минимальный набор средств для передачи файла, с их помощью можно пересыпать файлы разных типов. Эта интересная особенность протокола TFTP позволяет использовать его в программе чтения электронной почты<sup>4</sup>. При отправке запроса на сервер клиент может указать в поле *имени файла* полностью определенное имя почтового ящика, с которого сервер должен пересыпать информацию.

## 26.12. Система NFS

*Сетевая файловая система* (*Network File System*, или *NFS*), разработанная компанией Sun Microsystems Incorporated, обеспечивает коллективный интерактивный доступ к файлу. С точки зрения пользователя этот доступ является прозрачным, поскольку он интегрирован с файловой системой компьютера. К средствам NFS прибегают многие пользователи для объединения файловых систем своих компьютеров с помощью протокола TCP/IP. Для пользователя система NFS является практически незаметной. Он может запустить любую прикладную программу и использовать любые файлы для ввода или вывода, которые находятся как на локальных, так и на сетевых устройствах. По имени файла невозможно определить, является файл локальным или удаленным.

## 26.13. Реализация системы NFS

На рис. 26.3 показано, как система NFS встраивается в операционную систему. В процессе выполнения прикладная программа вызывает функции операционной системы, чтобы *открыть* файл, *сохранить* или *прочитать* данные из файла. Эти функции вначале обрабатываются специальной программой доступа к файлу, которая автоматически пересыпает запрос или локальной файловой системе, или клиенту NFS, в зависимости от того, где находится файл: на локальном диске или на удаленной машине. Получив запрос, клиентская программа с помощью протокола NFS подключается к соответствующему серверу, запущенному на удаленной машине, и выполняет требуемую операцию. Результаты ответа удаленного сервера клиентская программа пересыпает прикладной программе.

<sup>4</sup> На практике, использование протокола TFTP для передачи электронной почты не приветствуется. Для знакомства с тонкостями работы системы электронной почты обратитесь к главе 27, “Приложения: система электронной почты (SMTP, POP, IMAP, MIME)”.

**Печатай файл этой страницы отдельно**

не во всех компьютерах 32-битовые двоичные целые числа представлены в одинаковом формате. В некоторых процессорах старший байт числа хранится в памяти по меньшему адресу, а в других — по большему. Таким образом, если программист перешлет целое число по сети с одной машины на другую без изменения порядка следования байтов, значение этого числа может измениться. Механизм XDR позволяет решить эту проблему с помощью определения машинно-независимых форматов представления данных. Прежде чем передать данные на удаленный сервер, программа RPC вызывает соответствующие процедуры XDR и преобразует данные из локального физического представления в машинно-независимое представление. После получения данных сервером вновь вызываются процедуры XDR для преобразования данных из машинно-независимого представления в локальное физическое представление конкретной машины.

Главное преимущество использования механизма XDR состоит в том, что он автоматически выполняет преобразование большинства типов данных. Программисту нет необходимости вручную писать операторы вызова процедур XDR. Вместо этого, он указывает компилятору директивы XDR, в которых описываются необходимые преобразования, а компилятор автоматически генерирует программу с необходимыми вызовами библиотечных программ XDR.

## 26.15. Резюме

Существует две формы доступа к данным, хранящимся в удаленных файлах: копирование файла и коллективный интерактивный доступ. Протокол FTP является основным протоколом передачи файлов в семействе TCP/IP. В FTP используется метод копирования файла в любом направлении, а также предусмотрена возможность получения списка файлов и каталогов, хранящихся на удаленной машине. Простейший протокол передачи файлов TFTP является несложной альтернативой протокола FTP и используется в тех приложениях, для которых нужен только механизм передачи файлов. Поскольку программа реализации протокола TFTP имеет небольшой размер и свободно помещается в ПЗУ, она может использоваться для начальной загрузки бездисковых машин.

Сетевая файловая система (NFS), разработанная компанией Sun Microsystems Incorporated, предоставляет пользователям коллективный интерактивный доступ к файлам. Для транспортировки сообщений в ней используется протокол UDP, средства дистанционного вызова процедур (RPC) и независимый от платформы механизм представления внешних данных (XDR). Поскольку RPC и XDR определены независимо от NFS, программисты могут использовать их для создания распределенных прикладных программ.

## Материал для дальнейшего изучения

Стандарт протокола FTP описан Постелом (Postel) в [RFC 959]. Расширения его системы безопасности обсуждаются Горовицем (Horowitz) и Лантом (Lunt) в [RFC 2228], Оллманом (Allman) и Остерманном (Ostermann) в [RFC 2577] и Хауслеем (Housley) и Хоффманом (Hoffman) в [RFC 2585]. Комментарии по поводу протокола FTP даны более чем в трех десятках документов RFC: предложены его модификации или определены новые версии протокола. Среди них, Лоттором (Lottor) в [RFC 913] описан простейший протокол передачи файлов. Методы передачи файлов по FTP в фоновом режиме описаны де Шоном (DeSchon) и Браденом (Braden) в [RFC 1068]. Использование протокола FTP в среде протокола IPv6 и NAT рассматриваются Оллманом и Остерманном в [RFC 2428]. Простейший протокол передачи файлов, описанный в этой главе, придумал и описал Соллиnz (Sollins) в [RFC 783]. Использование протокола TFTP для выполнения начальной загрузки компьютер-

ных систем описано Финлейсоном (Finlayson) в [RFC 906], а его параметры обсуждаются Малкиным (Malkin) и Харкином (Harkin) в [RFC 2347] и [RFC 2348].

Компания Sun Microsystems опубликовала три документа RFC, в которых описывается сетевая файловая система и связанные с ней протоколы. Стандарт для NFS содержится в [RFC 1094], механизм RPC определяется в [RFC 1057], и, наконец, XDR определяется в [RFC 1014]. За более подробной информацией по RPC и NFS обратитесь к третьему тому этой книги.

## Упражнения

- 26.1. Почему даже при использовании надежного потокового транспортного протокола типа TCP протоколы передачи файлов должны вычислять контрольную сумму данных передаваемого файла?
- 26.2. Узнайте, вычисляется ли в протоколе FTP контрольная сумма для данных передаваемого файла.
- 26.3. Что произойдет в протоколе FTP, если при передаче данных вдруг разорвется TCP-соединение, но управляющее соединение при этом будет функционировать?
- 26.4. Какое главное преимущество использования отдельных TCP-соединений для передачи данных и команд управления в протоколе FTP? (*Подсказка. Рассмотрите аварийные ситуации.*)
- 26.5. Опишите в общих чертах метод, используемый программой TFTP в процессе начальной загрузки бездисковой машины. Будьте внимательны. Какие именно IP-адреса она использует на каждом этапе?
- 26.6. Напишите клиентскую часть программы протокола TFTP.
- 26.7. Поэкспериментируйте с протоколом FTP или эквивалентным ему протоколом и посмотрите, с какой скоростью вы сможете передать файл между двумя быстродействующими компьютерами по локальной сети. Попробуйте провести эксперимент, когда сеть загружена и когда она не загружена. Объясните результат.
- 26.8. Попробуйте использовать протокол FTP для передачи файлов внутри одной машины, а затем для передачи с одной машины на другую, находящуюся в той же локальной сети. Удивляют ли вас скорости передачи данных?
- 26.9. Сравните скорости передачи данных для протоколов FTP и NFS в локальной сети. Можете ли вы объяснить разницу?
- 26.10. Изучите документ RFC, в котором описан стандарт RPC. Обрабатываясь ли в RPC случаи потери, дублирования, задержки и искажения содержимого дейтаграмм?
- 26.11. Расширьте предыдущий вопрос и рассмотрите механизм NFS в котором используется RPC. Будет ли NFS хорошо работать в глобальной сети Internet? Почему?
- 26.12. При каких условиях механизм XDR неэффективен?
- 26.13. Рассмотрите процесс перевода чисел с плавающей запятой из внутренней формы во внешнюю и обратно во внутреннюю. Каково оптимальное решение при выборе размеров показателя степени и мантиссы для внешней формы?
- 26.14. В программе FTP по умолчанию установлен текстовый режим передачи файлов (ASCII). Разумно ли это? Докажите, что установка по умолчанию текстового режима передачи файлов может рассматриваться как “вредная”.

# 27

## Приложения: система электронной почты (SMTP, POP, IMAP, MIME)

### 27.1. Введение

В этой главе мы продолжим изучение принципов работы объединенной сети на примере службы электронной почты и поддерживающих ее протоколов. Здесь описана структура системы электронной почты, объяснена методика расширения псевдонимов, а также показано, как в программном обеспечении системы электронной почты используется принцип взаимодействия типа клиент/сервер для передачи сообщений.

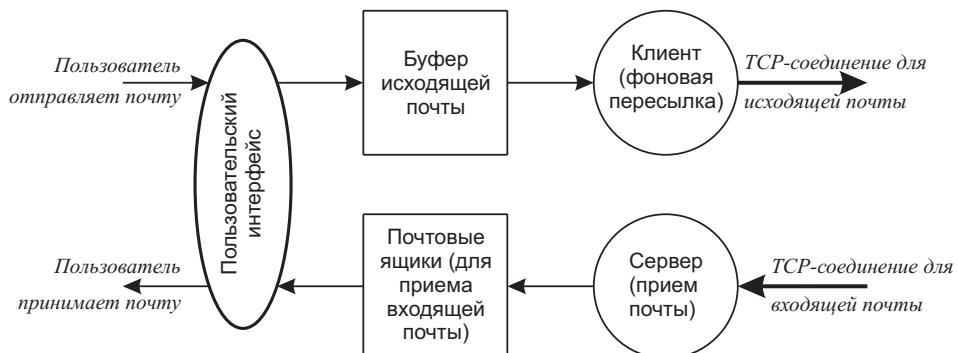
### 27.2. Служба электронной почты

Служба электронной почты (*e-mail*) позволяет пользователям отсыпать текстовые сообщения по объединенной сети. Электронная почта — это одна из широко используемых служб уровня приложений. Действительно, некоторые пользователи используют электронную почту для осуществления повседневной коммерческой деятельности.

Электронная почта приобрела популярность благодаря тому, что она является быстрым и удобным способом передачи информации. При этом не имеет значения какого размера сообщения передаются: короткие текстовые или длинные, содержащие файловые вложения. В обоих случаях используется один и тот же механизм пересылки. Вас, наверное, удивит тот факт, что большинство пользователей предпочитает обмениваться файлами по электронной почте, а не пересыпать их с помощью специализированных протоколов наподобие FTP.

Процесс доставки электронной почты в корне отличается от других уже рассмотренных нами способов использования объединенной сети и поэтому является новым понятием. Во всех приведенных нами примерах сетевые протоколы отсылают пакеты напрямую получателям. При этом отдельные сегменты передаются повторно, если в течение установленного интервала времени получатель не подтвердит прием. Однако при пересылке электронной почты должны быть учтены случаи, когда удаленная машина временно недоступна (например, по причине обрыва сетевого соединения). С точки зрения отправителя нежелательно приостанавливать работу до тех пор, пока не поступит ответ от удаленной машины. С другой стороны, неразумно аварийно завершать процесс передачи электронной почты только из-за того, что получатель временно недоступен.

Чтобы преодолеть проблему отсроченной доставки, в системах электронной почты используется метод *буферизации сообщений* (*spooling*). Когда пользователь отсылает сообщение по электронной почте, система помещает его копию во временный буфер (*spool*)<sup>1</sup>. Вместе с сообщением в буфер также помещается служебная информация: идентификационные параметры отправителя и получателя, адрес машины получателя, а также время, на протяжении которого копия должна храниться в буфере. Затем система начинает передачу данных удаленной машине в фоновом режиме, что позволяет отправителю выполнять другие вычислительные операции. Эта концепция схематически изображена на рис. 27.1.



*Рис. 27.1. Структурная схема системы электронной почты. Для отправки и чтения сообщений электронной почты пользователь вызывает специальную программу, в которой реализован пользовательский интерфейс; при этом пересылка данных происходит в фоновом режиме*

Фоновый процесс передачи электронной почты становится клиентом. Сначала он использует систему доменных имен, чтобы преобразовать имя машины получателя в IP-адрес, а затем пытается создать TCP-соединение с сервером электронной почты, запущенным на машине получателя. Если соединение удаётся установить, копия сообщения пересыпается удаленному серверу, который сохраняет ее в своем буфере. Получив от сервера подтверждение о том, что сообщение получено и сохранено в буфере, клиент удаляет локальную копию сообщения из своего буфера. Если в процессе передачи не удается создать TCP-соединение или соединение внезапно обрывается, фиксируется время доставки сообщения и работа клиента завершается. Периодически (как правило, каждые 30 минут) клиентский фоновый процесс просматривает область буферной памяти, проверяя, есть ли недоставленные сообщения. Обнаружив такое сообщение или помещенное пользователем новое исходящее сообщение, фоновый процесс пытается его доставить. Если по истечении установленного (достаточно длинного) интервала времени (например, трех дней) программа пересылки электронной почты не сможет доставить сообщение получателю, она возвращает его отправителю вместе с сообщением об ошибке.

### 27.3. Имена почтовых ящиков и псевдонимы

В приведенном выше упрощенном описании процесса доставки электронной почты было опущено три важных момента. Во-первых, пользователи определяют получателей сообщений с помощью двух строк символов: одна из них идентифи-

<sup>1</sup> Буферную область для размещения электронной почты иногда называют *почтовой очередью* (*mail queue*), хотя с формальной точки зрения этот термин неточен.

цирует *адрес машины получателя почты*, другая — *имя почтового ящика* на этой машине. Во-вторых, используемые при этом имена не зависят от других имен, назначенных машинам. Обычно имя почтового ящика совпадает с именем учетной записи пользователя, а адрес машины получателя — с доменным именем этой машины, но это не обязательно. Почтовый ящик можно назначить должностному лицу. Например, имя почтового ящика *departmenthead* (*заведующий отделом*) может относиться к любому лицу, возглавляющему отдел в данный момент. Кроме того, поскольку для получателей электронной почты в системе доменных имен предусмотрен специальный тип запроса, имена получателей электронной почты можно отделить от обычных доменных имен, назначенных машинам. Таким образом, электронная почта, отосланная пользователю машины *example.com*, может попасть на другую машину, хотя *telnet*-соединение по адресу *example.com* будет установлено именно с машиной *example.com*. В-третьих, в приведенной нами упрощенной схеме не учтены механизмы *обработки сообщений* (*mail processing*) и *пересылки сообщений* (*mail forwarding*). Под ними подразумевается передача электронной почты от одного пользователя к другому в пределах одной машины, а также пересылка почты, которая поступила на одну машину, но должна быть переслана другой машине.

## 27.4. Расширение псевдонимов и пересылка электронной почты

В большинстве систем используются программы *пересылки электронной почты*, в которых поддерживается механизм *расширения почтовых псевдонимов*. Суть его состоит в том, что идентификатор почтового ящика может преобразовываться в процессе доставки почты в один или несколько новых адресов электронной почты. Как правило, прежде чем переслать составленное пользователем сообщение системе доставки, интерфейсная программа обработки электронной почты обращается к списку локальных псевдонимов и заменяет псевдоним на реальный адрес электронной почты. Имена получателей, для которых не были определены псевдонимы, остаются неизмененными. Точно также система доставки электронной почты использует псевдонимы для преобразования адресов во входящих сообщениях в адреса получателей.

Применение псевдонимов в значительной степени повышает функциональность системы электронной почты и удобство ее использования. С математической точки зрения процесс преобразования псевдонимов можно описать тождеством “многие к одному” или “один ко многим”. Например, система псевдонимов позволяет одному пользователю иметь несколько идентификаторов электронной почты, которые могут быть мнемоническими именами или названием занимаемой должности. Причем все эти идентификаторы сопоставляются с одним субъектом. Система также позволяет серверу связать с одним идентификатором целую группу получателей. Используя псевдонимы, которые соответствуют списку идентификаторов, можно создать *систему массовой рассылки* электронной почты, когда одно входящее сообщение рассыпается большому количеству пользователей. Связанная с идентификатором совокупность получателей называется *списком рассылки электронной почты*. При этом в список могут включаться получатели, которые не относятся к локальному серверу. Хотя такие случаи встречаются редко, на произвольном сервере *Q* можно создать список рассылки, в котором ни один из получателей не принадлежит этому серверу. Расширение псевдонимов электронной почты в большую совокупность получателей является популярным и широко используемым приемом. На рис. 27.2. изображены компоненты системы электронной почты, поддерживающей псевдонимы и расширение элементов списка рассылки.

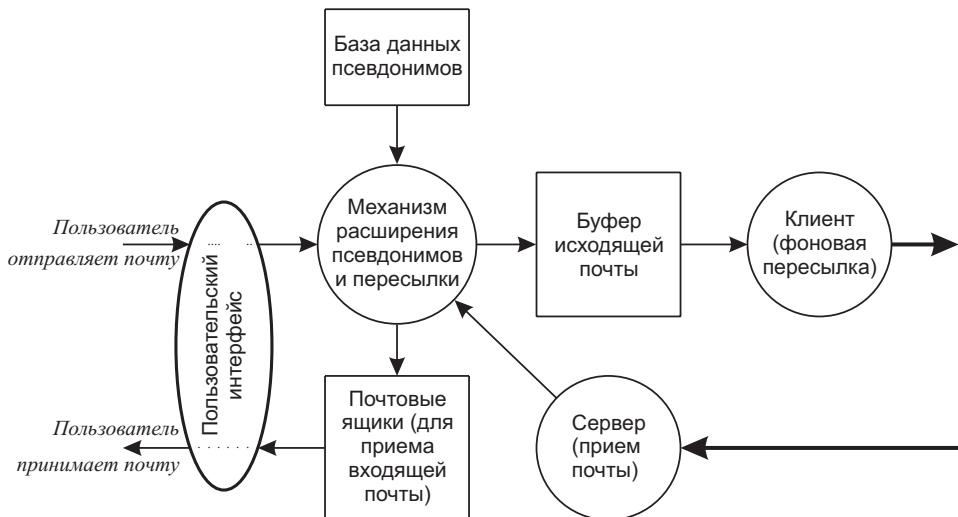


Рис. 27.2. Расширение изображенной на рис. 27.1 системы электронной почты, поддерживающей псевдонимы и пересылку сообщений. Механизм расширения псевдонимов используется как для входящих, так и исходящих сообщений

Как показано на рис. 27.2, при пересылке электронной почты к входящим и исходящим сообщениям применяется механизм расширения псевдонимов. Таким образом, если в базе данных псевдонимов электронный адрес  $x$  соответствует заменяющему его адресу  $y$ , механизм расширения псевдонимов изменит адрес получателя  $x$  на  $y$ . Затем программа расширения псевдонимов выясняет, соответствует ли  $y$  локальному или удаленному адресу. Таким образом, она определяет, в какую из двух очередей (входящую или исходящую) нужно поместить сообщение.

Процесс расширения псевдонимов электронной почты связан с потенциальными ошибками. Предположим, что на двух серверах созданы конфликтующие псевдонимы. Например, сервер  $A$  преобразует электронный адрес  $x$  в электронный адрес  $y$ , относящийся к серверу  $B$ . В свою очередь, сервер  $B$  преобразует электронный адрес  $y$  в электронный адрес  $x$ , относящийся к серверу  $A$ . В результате электронное сообщение, отосланное по адресу  $x$  сервером  $A$ , может бесконечно передаваться между двумя серверами<sup>2</sup>. Кроме того, администратор сервера  $A$  может случайно сопоставить локальному имени пользователя адрес электронной почты, относящийся к другому серверу. Тогда локальный пользователь не сможет получить свою почту, поскольку она будет переадресовываться другому пользователю. Если же псевдониму сопоставлен некорректный адрес электронной почты, отправители будут получать сообщения об ошибке.

## 27.5. Взаимосвязь между межсетевым обменом и электронной почтой

Существуют коммерческие службы, пересылающие электронную почту между компьютерами без помощи семейства протоколов TCP/IP; при этом от компьютеров не требуется подключение к глобальной сети Internet. Чем такие системы

<sup>2</sup> На практике большинство программ пересылки электронной почты завершает процесс доставки сообщений после того, как количество обменов достигло заранее определенной пороговой величины.

отличаются от описанной в этой главе системы электронной почты? Имеется два коренных отличия. Во-первых, объединенная сеть TCP/IP позволяет создать универсальную службу доставки. Во-вторых, системы электронной почты с использованием семейства протоколов TCP/IP на практике оказываются более надежны, чем их аналоги, работающие на основе других протоколов. Первое утверждение легко понять. Семейство протоколов TCP/IP позволяет создать универсальную службу доставки сообщений электронной почты, поскольку оно обеспечивает универсальный механизм взаимодействия между машинами. По сути все подключенные к объединенной сети машины функционируют так, будто они подключены к единой независимой от производителя оборудования сети. Очевидно, что при наличии базовых сетевых служб, процесс построения стандартного протокола обмена электронной почтой намного облегчается.

Второе утверждение о том, что использование семейства протоколов TCP/IP повышает надежность доставки электронной почты по сравнению с другими механизмами, требует пояснения. Основная идея заключается в том, что протокол TCP обеспечивает возможность создания сквозного соединения между отправителем и получателем. Это означает, что программы поддержки электронной почты на машине отправителя выполняют роль клиентов, связываясь с сервером на машине конечного получателя. Только успешно переслав сообщение электронной почты серверу, клиент удаляет его из локального буфера. Таким образом, прямая доставка сообщения между отправителем и конечным получателем подчиняется следующему принципу.

*В системах электронной почты, использующих принцип прямой доставки между отправителем и конечным получателем, гарантируется, что каждое электронное сообщение находится на машине отправителя до тех пор, пока его копия не будет успешно передана машине получателя. При использовании таких систем отправитель может всегда явно определять состояние каждого сообщения, проверив локальный буфер электронной почты.*

В альтернативных системах доставки электронной почты используются шлюзы уровня приложения, рассмотренные в главе 20, “Взаимодействие частных сетей (NAT, VPN)”. Сообщение пересыпается через ряд почтовых шлюзов<sup>3</sup> (*mail gateways*), иногда называемых почтовыми мостами (*mail bridges*), почтовыми ретрансляторами (*mail relays*) или промежуточными почтовыми станциями (*intermediate mail stops*). В таких системах машина отправителя не связывается напрямую с машиной конечного получателя. Вместо этого, целое сообщение электронной почты отсылается от исходного отправителя первому почтовому шлюзу. Затем сообщение пересыпается второму шлюзу и так далее.

Главный недостаток использования почтовых шлюзов заключается в том, что они снижают надежность системы доставки. Переслав сообщение первой промежуточной машине, компьютер отправителя удаляет локальную копию сообщения. Таким образом, пока сообщение находится в процессе доставки, его копии нет ни у отправителя, ни у конечного получателя. Сбои на промежуточных машинах могут привести к потере сообщения, и об этом не будет извещен ни отправитель, ни получатель. Потеря сообщения также может произойти, если почтовые шлюзы выполняют неправильную маршрутизацию электронной почты. Другой недостаток использования почтовых шлюзов заключается в том, что они вызывают задержку при передаче сообщений. В почтовом шлюзе сообщения могут задерживаться на минуты, часы или даже дни, если шлюзу не

<sup>3</sup> Читатели не должны путать термин *почтовый шлюз* с термином *IP-шлюз*, который был описан в главе 3, “Основы и структура межсетевого взаимодействия”.

удается переслать их другой машине. Ни отправитель, ни получатель не могут определить, в каком сегменте сети задержалось сообщение, почему оно до сих пор не доставлено и как долго продлится задержка. Важный момент заключается в том, что ни отправитель, ни конечный получатель никак не могут повлиять на работу промежуточных компьютеров.

Если система почтовых шлюзов менее надежна, чем прямая доставка, почему она все же используется? Главное преимущество почтовых шлюзов — возможность взаимодействия разнородных систем. Посредством почтовых шлюзов обеспечивается обмен информацией между стандартными системами электронной почты на основе протокола TCP/IP и другими системами электронной почты, а также между объединенными сетями TCP/IP и сетями, не поддерживающими протокол IP. Предположим, компания X имеет большую внутреннюю сеть, и ее сотрудники используют электронную почту. Однако сетевое программное обеспечение не поддерживает протокол TCP/IP. В описанном случае невозможно подключить сеть компании к глобальной сети Internet. Тем не менее между частной сетью компании и сетью Internet можно разместить почтовый шлюз и разработать программное обеспечение, принимающее сообщения из локальной сети и пересылающее их в сеть Internet, и наоборот.

Электронная почта превратилась в важное средство сетевых коммуникаций. Поэтому пользователи, не имеющие доступа к сети Internet, вынуждены работать через почтовый шлюз, хотя создание почтовых шлюзов — достаточно громоздкая операция. Таким образом, хотя служба почтовых шлюзов не такая надежная и удобная, как служба прямой доставки, она все же может оказаться полезной.

## 17.6. Протоколы семейства TCP/IP для службы электронной почты

Вспомним, что целью создания семейства протоколов TCP/IP была попытка обеспечить взаимодействие между максимально большим количеством компьютерных систем и сетей. Чтобы расширить возможность взаимодействия систем электронной почты, соответствующие стандарты семейства протоколов TCP/IP были разделены на два набора. Один набор протоколов определяет формат сообщений электронной почты<sup>4</sup>. В другом — определены детали обмена сообщениями электронной почты между двумя компьютерами. Разделение стандартов электронной почты на два независимых набора позволяет создать почтовые шлюзы, соединяющие системы доставки электронной почты сторонних производителей и стандартные почтовые системы объединенной сети TCP/IP. При этом в обеих системах используется один и тот же формат сообщений.

Как известно любому пользователю электронной почты, каждое текстовое сообщение состоит из двух частей: заголовка и тела, разделенных пустой строкой. В стандарте семейства протоколов TCP/IP для сообщений электронной почты определен строгий формат заголовков сообщений, а также семантическая интерпретация каждого поля заголовка. Формат тела сообщения определяется самим отправителем. В частности, в стандарте определено, что заголовки имеют текстовый формат и размещаются в отдельных строках. Стока заголовка состоит из ключевого слова, после которого следует двоеточие, а затем — значение. Некоторые ключевые слова являются обязательными, другие — необязательными, а остальные —необрабатываемыми. Например, в области заголовка должна находиться строка, определяющая получателя. Эта строка начинается из ключевого слова To: (*Кому:*), после которого указывается адрес электронной почты

<sup>4</sup> Специалисты говорят, что сообщения электронной почты соответствуют формату “822”, поскольку его стандарт определен в [RFC 822].

получателя сообщения. В строке, начинающейся со слова *From:* (*От:*), указывается адрес электронной почты отправителя. При желании отправитель может указать адрес, по которому нужно отсылать ответы (т.е. указать, что ответы необходимо посыпать по другому адресу, а не по адресу отправителя). Если в заголовке присутствует строка, начинающаяся со слов *Reply-to:* (*Обратный адрес:*), то в ней указан адрес, по которому следует пересыпать ответы. Если такая строка отсутствует, то в качестве адреса для пересылки ответов будет использоваться информация, содержащаяся в строке *From:*.

Стандартизация формата сообщений электронной почты облегчает их обработку и передачу через разнотипные почтовые серверы. Выбор простого текстового формата для заголовка сообщений электронной почты позволяет использовать его в различных системах. При этом исчезает проблема выбора стандарта для двоичного представления чисел, а также необходимость преобразования стандартного представления в представление, используемое в локальной системе.

## 27.7. Адреса электронной почты

Опытные пользователи знают, что в разных системах электронной почты используются разные форматы электронных адресов. Поэтому определить правильный формат адреса электронной почты или даже понять намерения отправителя бывает очень нелегко. В пределах глобальной сети Internet адреса имеют простой, легко запоминающийся вид:

локальная-часть@доменное-имя

Здесь *доменное-имя* определяет доменное имя получателя почты<sup>5</sup>, а *локальная-часть* — это имя почтового ящика, расположенного на машине получателя. Например, адрес электронной почты автора этой книги в сети Internet имеет следующий вид:

comer@purdue.edu

При использовании почтовых шлюзов адреса электронной почты усложняются. Если у пользователя нет прямого подключения к сети Internet, он должен или пересыпать электронную почту ближайшему почтовому шлюзу или использовать специальное программное обеспечение, которое сделает это автоматически. Например, когда в сети CSNET был запущен почтовый шлюз, который соединял ее внутренние сети с Internet, любой пользователь, имеющий доступ к шлюзу, мог послать автору этой книги сообщение по следующему адресу:

comer%purdue.edu@relay.cs.net

Когда электронная почта поступала на машину *relay.cs.net*, ее программное обеспечение извлекало локальную часть адреса, заменяло в ней знак процента (%) на знак @ и пересыпало почту на полученный в результате этого адрес получателя.

Если получатель или отправитель не имеют прямого подключения к Internet, адреса электронной почты усложняются, поскольку функция преобразования адресов электронной почты является локальной по отношению к каждому шлюзу. Таким образом, некоторые почтовые шлюзы требуют, чтобы локальная часть адреса имела следующий вид:

user%domain-name

В других шлюзах требуется, чтобы адреса были представлены в таком виде:

user:domain-name

---

<sup>5</sup> С формальной точки зрения доменное имя определяет адрес сервера, выполняющего пересылку электронной почты, а не имя машины.

Могут встречаться и другие формы адресов. Кроме того, в системах электронной почты обычно не устанавливается порядок обработки локальной части адреса. Поэтому нельзя гарантировать способ дальнейшей интерпретации адресов. Например, рассмотрим приведенный выше адрес электронной почты

`comer%purdue.edu@relay.cs.net`

Если в почтовом сервере для передачи электронной почты используется стандарт TCP/IP, то этот адрес будет обрабатываться следующим образом: сервер отошлет сообщение программе пересылки электронной почты, запущенной на машине `relay.cs.net`. При этом интерпретация локальной части адреса `comer%purdue.edu` будет зависеть от самой программы. По сути сервер пересылки электронной почты на основе протокола TCP/IP интерпретирует адрес так, будто его составляющие заключены в скобки:

`(comer%purdue.edu) @ (relay.cs.net)`

Если же в почтовом сервере для разделения имен пользователей и адресов машин получателей используется знак процента (%), то приведенный выше сложный адрес электронной почты будет интерпретироваться так: “отослать сообщение пользователю `comer`, который находится на машине, определяемой остальной частью адреса”. Это означает, что подобные серверы интерпретируют адрес так, будто он заключен в скобки:

`(comer) % (purdue.edu@relay.cs.net)`

Рассмотренную выше проблему можно обобщить так.

*Поскольку способ интерпретации и преобразования адресов электронной почты определяется программным обеспечением почтового шлюза, не существует единого стандарта для адресов сообщений, следующих через почтовые шлюзы во внешние относительно Internet сети.*

## 27.8. Псевдодоменные адреса

Чтобы решить проблему взаимодействия нескольких разнотипных систем электронной почты, в каждой из которых используется свой формат адресов, было предложено следующее решение. Почтовым серверам разрешено использовать имена, напоминающие доменные, для представления всех адресов электронной почты, даже если система доменных имен не используется на почтовом сервере получателя. Например, если на сервере для пересылки электронной почты используется протокол UUCP, администратор сервера может создать псевдодомен, `ииср`, чтобы пользователи могли пользоваться адресами электронной почты следующего формата:

`ииср-адрес-сообщения@ииср`

либо в связанном с ним формате:

`имя-пользователя@имя-ииср-сервера.ииср`

При этом локальная программа пересылки электронной почты распознает особый тип адресов и конвертирует их в адреса, используемые в программах поддержки протокола UUCP. С точки зрения пользователя преимущество такого метода понятно: все адреса электронной почты имеют одинаковый универсальный формат, не зависящий от используемых в сети протоколов связи, через которую сообщения доставляются адресату. Понятно, что такие адреса работают только на тех серверах, где есть локальная программа их преобразования, и только при условии поддержки соответствующих транспортных механизмов. Хотя

псевдодоменные адреса электронной почты имеют тот же формат, что и доменные имена, их можно использовать только для электронной почты. Преобразовать такие адреса в IP-адрес почтового сервера с помощью системы доменных имен невозможно.

## 27.9. Простой протокол передачи электронной почты (SMTP)

Помимо форматов сообщений, в семействе протоколов TCP/IP определен стандартный протокол для обмена сообщениями электронной почты между машинами. Это означает, что в стандарте определен точный перечень команд и форматов данных, которые должен использовать клиент для обмена почтой с сервером. Этот стандартный протокол называется *простым протоколом передачи электронной почты* (*Simple Mail Transfer Protocol* или *SMTP*). Как вы, наверное, уже догадались, протокол SMTP проще, чем использовавшийся ранее *протокол передачи электронной почты* (*Mail Transfer Protocol*, или *MTP*). При создании протокола SMTP основные усилия разработчиков были сосредоточены на способах доставки сообщений электронной почты по объединенной сети от одной машины до другой. В нем не определяется, каким образом система электронной почты принимает почту от пользователя или как программа пользовательского интерфейса обеспечивает пользователю доступ ко входящим сообщениям. В протоколе SMTP не определено, в каком виде должны храниться сообщения электронной почты в буфере и с какой частотой система электронной почты должна рассылать сообщения.

Протокол SMTP чрезвычайно прост и понятен. Взаимодействие между клиентом и сервером осуществляется с помощью команд, посылаемых в виде ASCII-строк. Хотя в протоколе SMTP строго определен формат команд, пользователи могут без труда просмотреть и понять процесс взаимодействия между клиентом и сервером. Сначала клиент устанавливает с сервером надежное потоковое соединение и ожидает, пока сервер не отправит сообщение 220 READY FOR MAIL, свидетельствующее о его готовности к приему электронной почты. (Если сервер перегружен, он может временно задержать отправку сообщения 220). Получив сообщение 220, клиент отсылает команду HELO<sup>6</sup>. Признаком конца команды является символ конца строки. В ответ сервер предоставляет свою идентификационную информацию. Как только взаимодействие установлено, отправитель может переслать одно или несколько сообщений электронной почты, закрыть соединение или послать серверу запрос на обмен ролями отправителя и получателя. Последняя команда необходима для того, чтобы можно было обмениваться сообщениями, следующими в обратном направлении. Получатель должен подтвердить прием каждого сообщения. Он также может досрочно закрыть соединение с сервером или оборвать текущую передачу сообщения.

Сеанс передачи электронной почты начинается с команды MAIL, с помощью которой клиент идентифицирует отправителя сообщения (пересыпает серверу информацию из поля FROM:), в чей адрес следует пересыпать возможные сообщения об ошибках. При этом сервер подготавливает внутренние структуры данных для получения нового сообщения электронной почты и в ответ на команду MAIL отсылает сообщение с кодом 250. Сообщение 250 означает, что процесс подготовки к получению почты прошел успешно. Полный ответ сервера состоит из текста 250 OK. Как и при использовании других протоколов уровня приложений, программы поддержки протокола SMTP “понимают” сокращения команд и

---

<sup>6</sup> HELO — это сокращение от слова “hello”.

обрабатывают трехзначный код ответа, расположенный в начале строки. Остальной текст ответа предназначен для того, чтобы пользователи могли отладить работу системы электронной почты.

После успешного выполнения команды MAIL отправитель с помощью нескольких команд RCPT идентифицирует адресатов сообщения электронной почты. Получатель должен подтвердить каждую команду RCPT путем отсылки ответного сообщения 250 OK или сообщения об ошибке 550, содержащего текст *No such user here (пользователь не найден)*.

После того как на все команды RCPT будут получены сигналы подтверждения, отправитель выдает команду DATA. Этим он сообщает получателю о своей готовности к отсылке полного сообщения электронной почты. На эту команду получатель отвечает сообщением 354 Start mail input (*начать прием почты*) и указывает последовательность символов, которые будут использоваться в качестве признака окончания сообщения электронной почты. Эта последовательность состоит из пяти символов: возврат каретки, перевод строки, точка, возврат каретки и перевод строки (<CR+LF+.+CR+LF>)<sup>7</sup>.

Приведенный ниже пример поможет прояснить процесс обмена сообщениями по протоколу SMTP. Предположим, что пользователь Smith компьютера Alpha.EDU отсылает сообщение пользователям Jones, Green и Brown компьютера Beta.GOV. Клиентское программное обеспечение протокола SMTP компьютера Alpha.EDU связывается с SMTP-сервером, запущенным на компьютере Beta.GOV, и начинает сеанс обмена данными, протокол которого приведен в листинге 27.1. Строки, начинающиеся с буквы “К”, являются командами клиента (Alpha.EDU), а строки, начинающиеся с буквы “С”, являются ответами сервера. В этом примере машина Beta.GOV не распознает пользователя Green, для которого предназначено сообщение.

#### Листинг 27.1. Пример сеанса связи по протоколу SMTP

```
C: 220 Beta.GOV Simple Mail Transfer Service Ready
K: HELO Alpha.EDU
C: 250 Beta.GOV
K: MAIL FROM:<Smith@Alpha.EDU>
C: 250 OK
K: RCPT TO:<Jones@Beta.GOV>
C: 250 OK
K: RCPT TO:<Green@Beta.GOV>
C: 550 No such user here
K: RCPT TO:<Brown@Beta.GOV>
C: 250 OK
K: DATA
C: 354 Start mail input; end with <CR><LF>.<CR><LF>
K: ...Пересыпает тело сообщения, которое может
K: ...состоять из произвольного количества строк
K: <CR><LF>.<CR><LF>
C: 250 OK
K: QUIT
C: 221 Beta.GOV Service closing transmission channel
```

<sup>7</sup> Признаком конца строки в протоколе SMTP служит последовательность управляющих символов: <CR+LF>. В теле сообщения электронной почты не должно быть строки, содержащей единственный символ — точку.

В приведенном выше примере сервер отказывается принять сообщение по адресу Green, поскольку он не смог распознать его как действительного получателя электронной почты (т.е. этот адрес не является ни именем пользователя, ни именем списка рассылки). В протоколе SMTP не оговаривается, как клиент должен обрабатывать подобные ошибки, — все зависит от реализации конкретной клиентской программы. Одно из возможных решений: при возникновении ошибки клиентская программа досрочно прекращает доставку сообщения и аварийно завершает работу. Однако обычно клиенты продолжают процесс доставки сообщений всем действительным адресатам, а затем сообщают о возникших проблемах отправителю сообщения. Как правило, клиент присыпает сообщение об ошибке по электронной почте. В нем содержится краткое изложение ошибки, а также заголовок сообщения электронной почты, при передаче которого возникла проблема.

После отправки всех сообщений, предназначенных для определенного сервера, клиент может выдать команду TURN<sup>8</sup>, чтобы изменить направление передачи сообщений по каналу связи. При подаче такой команды сервер отвечает текстом 250 OK и берет на себя управление каналом связи. После обмена ролями стороны соединения, которая раньше была сервером, отсылает клиенту сообщения электронной почты, находящиеся в очереди на отправку. Завершить сеанс связи может та из сторон, которая в настоящий момент управляет взаимодействием (т.е. клиент). Для этого используется команда QUIT. Другая сторона отвечает строкой 221, означающей согласие на завершения сеанса. Затем обе стороны по очереди закрывают TCP-соединение.

Протокол SMTP намного сложнее, чем можно судить из приведенного краткого описания. Например, при изменении местонахождения пользователя меняется его адрес электронной почты, который он может сообщить серверу. В протоколе SMTP предусмотрены средства, позволяющие серверу сообщить клиенту о новом адресе, чтобы клиент в дальнейшем его использовал. Сообщая новый адрес клиенту, сервер может либо сам переслать по нему сообщение, либо возложить это на клиента.

## 27.10. Получение электронной почты и протоколы управления почтовыми ящиками

В описании системы передачи электронной почты по протоколу SMTP предполагалось, что в любой момент времени сервер должен находиться в состоянии готовности к приему электронной почты. Клиент начинает процесс отправки сообщения, как только оно поступает от пользователя. Такой план действий хорошо срабатывает, когда сервер запущен на компьютере, имеющем постоянное подключение к объединенной сети. Если же компьютер периодически подключается к объединенной сети, описанный метод не работает. В частности, рассмотрим пример, в котором пользователь имеет доступ к сети Internet только посредством коммутируемого соединения. Для такого пользователя нет смысла запускать стандартный сервер электронной почты, поскольку доступ к нему можно получить только при установленном соединении с Internet. Все другие попытки связаться с сервером электронной почты потерпят неудачу, и отосланная пользователю электронная почта не будет доставлена. Возникает вопрос: каким образом пользователь, не имеющий постоянного подключения к объединенной сети, может получать электронную почту?

---

<sup>8</sup> На практике команда TURN используется в немногих серверах электронной почты.

Решение этой проблемы состоит в применении двухэтапного процесса доставки. На первом этапе каждому пользователю назначается почтовый ящик на компьютере, имеющем постоянное подключение к сети Internet. На этом компьютере запускается стандартный SMTP-сервер, который всегда находится в состоянии готовности к приему электронной почты. На втором этапе пользователь устанавливает с объединенной сетью коммутируемое соединение, а затем запускает программу поддержки протокола выборки сообщений из почтового ящика, находящегося на сервере. При этом сервер пересыпает полученные сообщения на компьютер пользователя, где их можно прочитать.

Существует два протокола, с помощью которых удаленный пользователь может извлекать электронную почту из почтового ящика на сервере. Эти протоколы выполняют одинаковые функции: помимо обеспечения доступа к почтовому ящику, они позволяют пользователю управлять содержимым этого ящика (например, удалить сообщение из ящика). Два упомянутых протокола будут рассмотрены в следующих разделах.

### 27.10.1. Почтовый протокол (POP3)

Самый распространенный протокол, используемый для передачи сообщений электронной почты из удаленного почтового ящика, находящегося на сервере, личному компьютеру, называется *почтовым протоколом версии 3* (*Post Office Protocol*, или *POP3*). Для получения почты пользователь вызывает программу клиента протокола POP3, которая устанавливает TCP-соединение с POP3-сервером, запущенным на компьютере, где находится удаленный почтовый ящик. Для аутентификации сеанса связи пользователь отсылает *регистрационное имя и пароль*. Когда процесс аутентификации успешно пройден, клиентская программа посыпает серверу команды на доставку копии одного или нескольких сообщений и удаления их из удаленного почтового ящика. Сообщения хранятся и передаются в виде текстовых файлов, представленных в стандартном формате “822”.

Обратите внимание, что на имеющем постоянное подключение к Internet компьютере должны быть запущены два сервера. Сервер SMTP принимает отосланную пользователю почту и помещает полученные сообщения в его почтовый ящик. Сервер POP3 позволяет пользователю извлекать сообщения из почтового ящика и удалять их. Чтобы обеспечить корректную работу системы, оба сервера должны согласованно использовать почтовый ящик. Поэтому если сообщение поступает через SMTP-сервер, в то время как пользователь извлекает сообщения через сервер протокола POP3, содержимое почтового ящика остается в достоверном состоянии.

### 27.10.2. Протокол доступа к сообщениям в сети Internet (IMAP4)

Версия 4 протокола доступа к сообщениям в сети Internet (*Internet Message Access Protocol*, или *IMAP4*) является альтернативой протоколу POP3, в основу работы которого положен такой же общий принцип. Подобно протоколу POP3, в протоколе IMAP4 определено абстрактное понятие, называемое *почтовым ящиком* (*mailbox*). Почтовые ящики размещаются на том же компьютере, что и сама программа сервера. Также подобно протоколу POP3, пользователь запускает программу клиента протокола IMAP4, которая связывается с сервером для выборки сообщений. Однако в отличие от POP3 протокол IMAP4 позволяет пользователю динамически создавать, удалять или переименовывать почтовые ящики.

В протоколе IMAP4 расширены функциональные возможности по выборке и обработке сообщений. Пользователь может получить информацию о сообщении или проанализировать поля его заголовка, не извлекая сообщение целиком. Кроме того, пользователь может выполнить поиск заданной строки и извлечь из

сообщения определенные части. Такая возможность особенно удобна при использовании низкоскоростных коммутируемых каналов связи, поскольку пользователю не нужно загружать с сервера лишние данные.

## 27.11. Расширение MIME для данных, представленных не в ASCII-формате

Стандарт *многоцелевых расширений электронной почты в сети Internet (Multipurpose Internet Mail Extensions, или MIME)* был разработан для обеспечения передачи по электронной почте данных, представленных не в ASCII-формате. Стандарт MIME не вносит изменений в протоколы SMTP или POP3 и не заменяет их. В протоколе MIME определены алгоритмы кодирования двоичных данных и представления их в формате ASCII. Это позволяет пересылать получателю файлы произвольных типов в обычном сообщении электронной почты. Для этого в каждое сообщение протокола MIME помещают служебную информацию, в которой указывается тип передаваемых данных и используемый алгоритм кодировки. Информация протокола MIME помещается в заголовок электронной почты формата “822”. В MIME-заголовках указывается используемая версия протокола MIME, тип передаваемых данных и метод кодирования, используемый для преобразования данных в формат ASCII. Например, в листинге 27.2 приведен пример сообщения, содержащего MIME-заголовки, которое используется для передачи графического файла в формате GIF<sup>9</sup>. Изображение в формате GIF было преобразовано в семибитовое представление в формате ASCII при использовании кодирования *base64*.

### Листинг 27.2. Пример сообщения, содержащего MIME-заголовки

```
From: bill@acollege.edu
To: john@example.com
MIME-Version: 1.0
Content-Type: image/gif
Content-Transfer-Encoding: base64
...Данные изображения...
```

Как видно из листинга, в строке заголовка *MIME-Version*: объявлено, что сообщение составлено с использованием версии 1.0 протокола MIME. В строке *Content-Type*: указывается, что в сообщении содержатся данные, представленные в формате GIF, а в заголовке *Content-Transfer-Encoding*: объявлено, что для преобразования изображения в формат ASCII использовано кодирование *base64*. Для просмотра изображения система электронной почты получателя должна сначала преобразовать его из кодировки *base64* обратно в формат GIF, а затем запустить программу, отображающую изображения в формате GIF на экране монитора.

В стандарте протокола MIME определено, что в строке *Content-Type*: должны указываться два идентификатора: *тип* содержимого (*content type*) и *подтип* (*subtype*), разделенные косой чертой. В приведенном выше примере *image* — тип содержимого, а *gif* — подтип.

В стандарте определено семь базовых типов содержимого, для каждого из которых установлены определенные подтипы, и типы кодирования передачи. Например, совместно с типом *image* может использоваться подтип *jpeг* или *gif*.

<sup>9</sup> GIF — аббревиатура английских слов Graphics Interchange Format (формат для обмена графикой).

Однако такие подтипы не могут использоваться при передаче текстовых сообщений (совместно с типом `text`). Помимо стандартных типов и подтипов, протокол MIME позволяет отправителю и получателю определять собственные типы содержимого<sup>10</sup>. В табл. 27.5 перечислены семь основных типов содержимого протокола MIME.

**Таблица 27.5. Типы содержимого протокола MIME, которые могут указываться в заголовке Content-Type**

<b>Заголовок Content-Type</b>	<b>Сообщение содержит...</b>
<code>text</code>	текстовые данные, например документ в формате Microsoft Word
<code>image</code>	статическое изображение (фотографию или рисунок, выполненный на компьютере)
<code>audio</code>	запись звуковых данных
<code>video</code>	движущееся изображение (например, видеозапись)
<code>application</code>	бинарные данные, составляющие исполняемый файл прикладной программы
<code>multipart</code>	несколько сообщений MIME, в каждом из которых указывается тип его содержимого и вид кодировки
<code>message</code>	полное сообщение электронной почты (например, при переадресации сообщения) или внешнюю ссылку на такое сообщение (например, адрес FTP-сервера и имя файла)

## 27.12. Комбинированные сообщения MIME

Тип содержимого протокола MIME `multipart` очень удобен в применении, поскольку он значительно повышает универсальность системы электронной почты. Для типа `multipart` в стандарте определено четыре возможных подтипа, причем каждый из подтипов обеспечивает важные функциональные возможности. Подтип `mixed` позволяет помещать в одно сообщение несколько независимых вложенных сообщений, каждое из которых может иметь свой тип и метод кодировки. В состоящее из нескольких частей сообщение типа `mixed` можно поместить фрагменты текстовых, графических или аудиоданных, целые сообщения электронной почты с вложенными дополнительными сегментами данных, которые являются аналогами *вложений*, прилагаемых к деловому письму. Подтип `alternative` позволяет включить в одно сообщение несколько представлений одних и тех же данных. Такие сообщения часто используются при массовой рассылке большому количеству адресатов, которые работают на разных аппаратных и программных платформах. Например, можно отослать документ как в виде простого ASCII-текста, так и в форматированном виде. Это позволяет адресатам, у которых на компьютерах установлены графические операционные системы, выбрать для просмотра документа форматированный вид. Подтип `parallel` позволяет включить в одно сообщение несколько частей, которые необходимо рассматривать как одно целое (например, фрагменты видео- или аудиоданных, которые необходимо воспроизвести одновременно). И наконец, подтип `digest` позволяет включить в одно сообщение набор других сообщений (например, совокупность сообщений электронной почты, составляющих дискуссию).

<sup>10</sup> Во избежание конфликта имен в стандарте требуется, чтобы каждое из имен, выбранных для частных типов содержимого, начиналось со строки X-.

В листинге 27.3 показан один из типичных примеров использования сообщения, состоящего из нескольких частей. В таком сообщении обычно содержится короткая текстовая пояснительная записка, а также другие элементы, в которых находится нетекстовая информация. В листинге 27.3 записка, содержащаяся в первой части сообщения, поясняет, что во второй его части находится фотографическое изображение.

### Листинг 27.3. Пример комбинированного сообщения, содержащего несколько независимых MIME-частей

```
From: bill@acollege.edu
To: john@example.com
MIME-Version: 1.0
Content-Type: Multipart/Mixed; Boundary=StartOfNextPart

--StartOfNextPart
Джон,
вот фотография нашей исследовательской лаборатории, которую я обещал тебе выслать. На ней – подаренное тобой оборудование.
Еще раз спасибо,
Билл

--StartOfNextPart
Content-Type: image/gif
Content-Transfer-Encoding: base64
...Данные изображения...
```

На примере листинга 27.3 также показано несколько особенностей протокола MIME. Например, в строке заголовка после основных объявлений могут следовать дополнительные параметры в виде *X=Y*. После объявления типа содержимого *Multipart/Mixed* через точку с запятой указано ключевое слово *Boundary*=, которое определяет строку, разделяющую части сообщения. В приведенном выше примере отправитель выбрал в качестве такой строки *"StartOfNextPart"*. Объявления типа содержимого и способа кодирования передачи для вложенного сообщения (если они присутствуют) указываются сразу же за разделяющей строкой. В рассматриваемом примере объявлено, что второе вложенное сообщение содержит изображение в формате GIF.

## 27.13. Резюме

Электронная почта является одной из широко распространенных служб уровня приложений объединенной сети. Подобно большинству служб семейства протоколов TCP/IP в ней используется модель взаимодействия типа клиент/сервер. Система электронной почты помещает в буфер исходящие и входящие сообщения, благодаря чему передача сообщения от клиента к серверу может выполняться в фоновом режиме.

В семействе протоколов TCP/IP предусмотрены отдельные стандарты, определяющие формат сообщений электронной почты и способ ее передачи. В формате сообщения электронной почты под названием “822” для отделения заголовка сообщения от его тела используется пустая строка. Простой протокол передачи электронной почты (SMTP) определяет, как система электронной почты на одной машине пересыпает электронную почту серверу на другой. В версии 3 почтового протокола (POP3) определено, каким образом пользователь может извлечь содержимое своего почтового ящика, расположенного на сервере, который имеет

постоянное подключение к сети Internet. Это позволяет доставлять электронную почту даже тому пользователю, который периодически подключается к сети Internet с помощью модема.

В стандарте многоцелевых расширений электронной почты в сети Internet (MIME) используется механизм, позволяющий пересыпать бинарные данные с помощью протокола SMTP. При использовании протокола MIME в заголовок сообщения электронной почты добавляются строки, определяющие тип данных и способы кодирования. Протокол MIME позволяет также создавать комбинированные сообщения электронной почты, состоящие из нескольких фрагментов данных разного типа.

## Материал для дальнейшего изучения

Детальное рассмотрение всех описанных в этой главе протоколов можно найти в документах RFC сети Internet. Простой протокол передачи электронной почты описан Постелом (Postel) в [RFC 821]; там же приведено большое количество примеров. Формат сообщений электронной почты описан Крокером (Crockier) в [RFC 822]. Многие документы RFC посвящены рассмотрению дополнений и изменений. Фрид (Freed) и Боренштейн (Borenstein) в [RFC 2045, 2046, 2047, 2048 и 2049] описывают стандарт протокола MIME, включая синтаксическую структуру объявлений заголовка, процесс создания новых типов содержимого, интерпретацию типов содержимого и упомянутый в этой главе алгоритм кодирования *base64*. Партидж (Partridge) в [RFC 974] обсуждает взаимосвязь между маршрутизацией электронной почты и системой доменных имен. Стандарт UUCP для системы электронной почты UNIX предложен Хортоном (Horton) в [RFC 976].

## Упражнения

- 27.1. При использовании некоторых систем электронной почты пользователь вынужден определять список машин, через которые должно пройти сообщение, чтобы достичь получателя. При этом программа поддержки протокола электронной почты, запущенная на каждой из машин, просто пересыпает сообщение следующей по порядку машине, указанной в списке. Перечислите три преимущества использования такой системы.
- 27.2. Выясните, позволяет ли операционная система вашего компьютера напрямую передать сообщение с помощью протокола SMTP.
- 27.3. Напишите программу клиента протокола SMTP и используйте ее для доставки сообщений электронной почты.
- 27.4. Выясните, сможете ли вы отослать электронную почту через почтовый шлюз обратно на свой компьютер.
- 27.5. Составьте список форм адресов электронной почты, которые обрабатываются системой электронной почты вашего компьютера, и напишите список правил для выполнения их синтаксического анализа.
- 27.6. Выясните, как использовать программу *sendmail* системы UNIX для реализации почтового шлюза.
- 27.7. Выясните, как часто ваша локальная система электронной почты пытается доставить сообщения, сколько времени длится попытка доставки, прежде чем программа решит ее прекратить.

- 27.8.** Многие системы электронной почты позволяют пользователям перенаправлять входящие сообщения электронной почты на обработку специальной программе, а не просто складывать их в почтовом ящике. Напишите программу, которая будет принимать вашу входящую почту, записывать сообщения в файл, а затем отсылать отправителю ответ, информируя его, что в данный момент вы находитесь в отпуске.
- 27.9.** Внимательно прочитайте стандарт протокола SMTP. Затем с помощью программы клиента TELNET попытайтесь подключиться к порту протокола SMTP удаленной машины и отправьте SMTP-серверу запрос на преобразование псевдонима электронной почты.
- 27.10.** Предположим, пользователь получил сообщение, в котором в поле To: указана строка `important-people` (*важные лица*). Известно также, что сообщение отправлено с компьютера, на котором псевдониму `important-people` не сопоставлен ни один реальный идентификатор почтового ящика. Внимательно прочитайте спецификацию протокола SMTP и поясните, как может возникнуть такая ситуация.
- 27.11.** В протоколе POP3 процессы выборки и удаления сообщений разделены. Другими словами пользователь может извлечь и просмотреть сообщение, не удаляя его из почтового ящика на сервере. В чем преимущества и недостатки такого разделения?
- 27.12.** Ознакомьтесь со спецификацией протокола POP3. Как работает команда TOP и в каких случаях она может пригодиться?
- 27.13.** Внимательно прочитайте стандарт протокола MIME. Адреса серверов какого типа могут быть указаны во внешней ссылке сообщения MIME?



# 28

## Приложения: *World Wide Web* (HTTP)

### 28.1. Введение

В этой главе мы продолжаем рассмотрение приложений, использующих технологию TCP/IP. Основное внимание в ней уделяется службе, оказавшей наибольшее влияние на развитие Internet, — службе *World Wide Web (WWW)*. После краткого обзора общих концепций будет описан основной протокол, используемый для передачи Web-страницы с сервера Web-браузеру. Кроме основного механизма передачи здесь будет рассмотрена также система кэширования.

### 28.2. Важность службы Web

На заре развития глобальной сети Internet приблизительно треть всего трафика, проходящего по объединенной сети, приходилась на долю приложений передачи данных по протоколу FTP. Этот трафик в несколько раз превышал объем трафика любого другого приложения. Однако служба Web, которая появилась в начале 90-х годов XX века, имела намного более высокие темпы роста. Уже к 1995 году трафик службы Web превысил трафик службы FTP. На его транспортировку затрачивается большая часть пропускной способности глобальной сети Internet. К 2000 году трафик службы Web полностью вытеснил трафик других протоколов.

Хотя не составляет большого труда измерить величину трафика и опубликовать эти данные, они не смогут отразить всей степени важности службы Web. Что такое Web, знают все пользователи объединенной сети, поскольку именно эту службу они используют чаще всего. Большинство компаний имеет собственные Web-серверы, на которых размещены интерактивные каталоги их продукции и предоставляемых услуг. Адреса Web-серверов сегодня можно увидеть практически в любой рекламе. Для многих пользователей глобальная сеть Internet и служба Web — неразделимые понятия.

### 28.3. Структурные компоненты

Концептуально служба Web состоит из большого набора документов, называемых *Web-страницами*, доступ к которым можно получить из любой точки глобальной сети Internet. Каждая Web-страница относится к виду так называемых *гипермедиа* (*hypermedia*) документов. Сuffix *медиа* (*media*) используется для указания, что документ может содержать не только текст, но и другие

элементы (например, графические изображения). Приставка *гипер* (*hyper*) означает, что документ может содержать *выбираемые ссылки*, которые ведут к другим связанным документам.

Для реализации службы Web в глобальной сети Internet используются два основных структурных элемента. Для получения доступа и отображения Web-страницы используется *Web-браузер*, представляющий собой пользовательское приложение. Браузер является клиентом, который связывается с соответствующим Web-сервером для получения копии указанной страницы. При создании запроса браузер должен указать точное положение страницы на сервере, поскольку обычно там содержится большое количество страниц.

Стандарты представления данных, используемые для Web-страницы, зависят от ее содержимого. Например, если страница содержит только графическое изображение, то оно может быть представлено в форматах *GIF* (*Graphics Interchange Format*, или *формат для обмена графикой*) или *JPEG* (*Joint Photography Experts Group*, или *формат, разработанный объединенной группой экспертов в области фотографии*). Web-страницы, содержащие как текст, так и другие элементы, оформляются на языке *гипертекстовой разметки* (*Hypertext Markup Language*, или *HTML*). HTML-документ является обычным текстовым файлом, который кроме собственно текста содержит внедренные в него команды. Эти команды называются *дескрипторами*, или *тэгами*. Они предназначены для обработки браузерами и управляют отображением страницы на экране монитора. Дескриптор заключается в угловые скобки, образованные из символов “меньше чем” и “больше чем” (“<” и “>”). Некоторые дескрипторы являются парными. При этом специфические атрибуты форматирования применяются ко всем элементам, заключенным между парой дескрипторов. Например, команды *<CENTER>* и *</CENTER>* заставляют браузер расположить элементы, находящиеся между ними, по центру окна.

## 28.4. Унифицированные указатели информационного ресурса

Каждой Web-странице присваивается уникальное имя, использующееся для ее идентификации в глобальной сети. Это имя называется *унифицированным указателем информационного ресурса* (*Uniform Resource Locator*, или *URL*)<sup>1</sup>. В начале URL указывается *спецификация* метода доступа к данному элементу. В действительности эта спецификация определяет тип протокола передачи. Формат остальной части URL зависит от указанной спецификации доступа. Например, URL, который следует за *спецификацией* протокола *http*, имеет следующий вид<sup>2</sup>:

`http://имя-узла-сети[:порт]/путь[;параметры][?запрос]`

Здесь в квадратных скобках указаны необязательные элементы. Пока достаточно понять, что строка *имени-узла-сети* определяет доменное имя или IP-адрес компьютера, на котором работает сервер для этого элемента. Стока *:порт* — необязательный номер порта протокола, необходимый только в тех случаях, когда сервер не использует стандартный порт (80). Стока */путь* идентифицирует документ на сервере. Необязательная строка *;параметры* определяет дополнительные параметры, заданные клиентом, а необязательная строка *?запрос* используется при отправке браузером запроса на сервер. Обычно поль-

<sup>1</sup> URL является отдельным типом более общего *универсального идентификатора ресурса* (*Uniform Resource Identifier*, или *URI*).

<sup>2</sup> В некоторой литературе начальную часть URL (в данном случае *http:*) называют *псевдо-комментарием* (*pragma*).

зователь не задает непосредственно необязательные части URL. Он вводит только имя-узла-сети и путь. Например, URL

`http://www.cs.purdue.edu/people/comer`

определяет Web-страницу автора этой книги. Из этого URL следует, что Web-сервер работает на компьютере `www.cs.purdue.edu`, а документ имеет название `/people/comer`.

В стандартах протокола различают *абсолютную* форму URL, показанную выше, и *относительную* форму URL. Относительная форма URL, с которой иногда может столкнуться пользователь, имеет смысл только в том случае, если сервер уже определен. Относительные формы URL применяются после установки связи с нужным сервером. Например, во время сеанса связи с сервером `www.cs.purdue.edu` для указания документа, абсолютный URL которого приведен выше, необходима только строка `/people/comer`. Все сказанное выше можно подытожить так.

*Каждой Web-странице в Internet присваивается уникальный идентификатор, называемый унифицированным указателем информационного ресурса (URL). Существует две формы URL: абсолютная и относительная. Абсолютная форма URL содержит полную спецификацию документа. Относительная форма URL (в ней адрес сервера не указывается) применяется только в том случае, когда сервер задан неявно.*

## 28.5. Пример HTML–документа

Получить доступ к службе Web очень просто. Все запросы в виде URL направляются серверу. Пользователь может ввести URL вручную с помощью клавиатуры либо выбрать его из списка, предложенного броузером. Броузер выполняет синтаксический анализ введенного URL и извлекает из него информацию, необходимую для получения копии требуемой страницы. Поскольку формат URL зависит от метода доступа, то броузер сначала извлекает спецификацию метода доступа, а затем, с ее помощью определяет способ выполнения синтаксического анализа остальной части URL.

Ниже приводится пример обработки URL, полученного из *выбираемой ссылки* в документе. Фактически в документе содержится два значения для каждой ссылки: элемент, который будет отображен на экране; и URL, по которому надо следовать, если пользователь выберет этот элемент. Для определения ссылки в документе HTML предусмотрена специальная пара дескрипторов `<A>` и `</A>`, называемых *привязкой (anchor)*. URL помещается в первый дескриптор привязки, а элементы, которые будут отображены на экране, помещаются между дескрипторами `<A>` и `</A>`. При выборе пользователем этой ссылки броузер следует по URL, хранящемуся внутри документа HTML. Например, в показанном ниже документе HTML содержится выбираемая ссылка.

```
<HTML>
Автор этой книги -
<A HREF="http://www.cs.purdue.edu/people/comer">
Дуглас Камер.</A>
</HTML>
```

При отображении этого документа в окне броузера появится одна строка текста:  
Автор этой книги – Дуглас Камер.

Броузер подчеркивает фразу Дуглас Камер, указывая тем самым пользователю, что она соответствует выбираемой ссылке. Если пользователь выберет эту ссылку, броузер загрузит в свое окно документ HTML, соответствующий URL, который хранится внутри дескриптора `<A>`.

## 28.6. Протокол передачи гипертекста

Протокол, используемый для связи между броузером и Web-сервером или между промежуточными машинами и Web-серверами называется *протоколом передачи гипертекста (HyperText Transfer Protocol, или HTTP)*. Протокол HTTP имеет следующие отличительные особенности.

- *Уровень приложения.* Протокол HTTP работает на уровне приложения. Для передачи данных в нем используется надежный потоковый транспортный протокол с установкой соединения между отправителем и получателем (типа TCP). Однако непосредственно в протоколе HTTP не предусмотрены средства обеспечения надежности передачи данных (например, их повторная передача).
- *Принцип запрос/ответ.* После установки сеанса связи с сервером одна из сторон (броузер) должна послать HTTP-запрос, в ответ на который другая сторона (сервер) присыпает данные.
- *Без фиксации состояния.* Каждый HTTP-запрос является независимым. Сервер не сохраняет список предыдущих запросов или предыдущих сеансов работы с клиентом.
- *Двунаправленная передача.* В большинстве случаев броузер запрашивает Web-страницу, а сервер передает ее копию броузеру. Однако в протоколе HTTP предусмотрена также передача данных от броузера серверу (например, когда пользователь отправляет на обработку серверу данные так называемой формы).
- *Возможность согласования параметров.* Протокол HTTP позволяет броузеру и серверу согласовывать параметры сеанса связи, например набор символов, который нужно использовать при передаче. При этом отправитель сообщает серверу набор параметров, который он поддерживает, а получатель возвращает те из них, которые он может принять.
- *Возможность кэширования.* Чтобы уменьшить время ответа, броузер кэширует копию каждой Web-страницы, которую он получает с сервера. При этом, если пользователь снова запросит копию той же страницы, протокол HTTP позволяет броузеру опросить сервер и определить, изменилось ли содержимое страницы с момента ее последней выборки.
- *Поддержка промежуточных серверов.* Протокол HTTP позволяет создавать промежуточные proxy-серверы (*proxy server*), расположенные между броузером и Web-сервером. Они кэшируют Web-страницы и на запрос броузера возвращают документы, находящиеся в локальном кэше.

## 28.7. HTTP-запрос GET

В самом простом случае для получения Web-страницы броузер связывается непосредственно с Web-сервером. Сначала броузер выполняет синтаксический анализ введенного пользователем URL, извлекает из него имя узла сети (*hostname*), использует DNS для преобразования этого имени в эквивалентный IP-адрес и использует полученный IP-адрес для установки TCP-соединения с сервером. Когда соединение установлено, броузер и Web-сервер используют для связи между собой протокол HTTP. При этом броузер отсылает запрос серверу на получение нужной ему Web-страницы. В ответ на запрос сервер присыпает копию этой страницы.

Для запроса Web-страницы у сервера броузер использует команду протокола HTTP `GET` (*получить*)<sup>3</sup>. Запрос состоит из одной строки текста, которая начинается с ключевого слова `GET`, за которым указываются URL и номер версии протокола HTTP. Например, чтобы получить копию Web-страницы, рассмотренную в предыдущем примере, с сервера `www.cs.purdue.edu`, броузер может отослать следующий запрос:

```
GET http://www.cs.purdue.edu/people/comer HTTP/1.0
```

После установки TCP-соединения с сервером исчезает необходимость в использовании абсолютного URL, поскольку ссылка приведенного ниже относительного URL приведет к выборке той же Web-страницы:

```
GET /people/comer HTTP/1.0
```

Можно подвести некоторые итоги.

*Протокол передачи гипертекста (HTTP) используется для обмена данными между броузером и Web-сервером. Для получения страницы с сервера броузер отсылает ему HTTP-запрос `GET`, в ответ на который сервер возвращает запрашиваемый элемент.*

## 28.8. Сообщения об ошибках

Какова должна быть реакция Web-сервера на некорректный запрос? В большинстве случаев запрос отсылается броузером, поэтому броузер отобразит на экране все, что вернет сервер. Следовательно, серверы могут формировать сообщения об ошибках на языке HTML, что и происходит в большинстве случаев. Вот пример сообщения об ошибке:

```
<HTML>
  <HEAD> <TITLE>400 Некорректный запрос</TITLE>
  </HEAD>
  <BODY>
    <H1>Некорректный запрос</H1>Броузер отоспал запрос,
    который сервер не смог распознать.
  </BODY>
</HTML>
```

Поле заголовка документа (т.е. элементы, расположенные между дескрипторами `<HEAD>` и `</HEAD>`) обрабатывается броузером, а пользователю отображается только тело документа (т.е. элементы, расположенные между дескрипторами `<BODY>` и `</BODY>`). Пара дескрипторов `<H1>` и `</H1>` заставляет броузер отображать текст Некорректный запрос в виде заголовка первого уровня (т.е. крупным полужирным шрифтом). Само же сообщение об ошибке появится на экране пользователя в виде двух строк:

### Некорректный запрос

Броузер отоспал запрос, который сервер не смог распознать.

## 28.9. Использование постоянных соединений

В первоначальных версиях протокола HTTP использовался тот же принцип, что и в протоколе FTP, — для каждого сеанса передачи данных открывалось новое TCP-соединение. Другими словами, клиент открывал TCP-соединение и отсыпал

---

<sup>3</sup> В спецификации протокола HTTP используется объектно-ориентированная терминология. Поэтому вместо термина “команда” используется термин “метод”.

серверу запрос GET. Передав клиенту копию требуемого документа, сервер закрывал TCP-соединение. Клиент считывал данные из открытого TCP-соединения до тех пор, пока он не получал признак конца файла. После этого клиент закрывал свою сторону соединения.

В июне 1999 года появился документ RFC, в котором описывалась версия 1.1 протокола HTTP. В нем был кардинально пересмотрен принцип взаимодействия клиента и сервера по протоколу HTTP. Вместо использования для каждой передачи данных отдельного TCP-соединения, в версии 1.1 по умолчанию используется одно *постоянное TCP-соединение*. То есть, открыв TCP-соединение с сервером, клиент использует его для передачи нескольких запросов и ответов. Когда одной из сторон соединения (клиенту или серверу) нужно закрыть соединение, она сообщает об этом другой стороне и закрывает соединение.

Основное преимущество использования постоянных соединений заключается в уменьшении накладных расходов при передаче служебной информации. Уменьшение количества TCP-соединений приводит к сокращению времени ответа сервера, уменьшению передачи служебной информации по объединенной сети, сокращению количества памяти, используемой для внутренних буферов, и уменьшению времени центрального процессора, затрачиваемого на обработку запросов. При использовании постоянного соединения работа броузера может быть оптимизирована за счет использования *конвейерных запросов* (pipelining requests) (т.е. броузер может отослать серверу подряд несколько запросов, не ожидая ответа на них). Конвейерные запросы особенно привлекательны в ситуациях, когда для отображения Web-страницы требуется загрузить с сервера несколько изображений, а объединенная сеть имеет высокую пропускную способность и большую задержку при передаче данных.

Основной недостаток использования постоянного соединения заключается в том, что необходимо как-то идентифицировать начало и конец каждого элемента, посылаемого по одному TCP-соединению. Существует два возможных решения этой проблемы: отсылать перед элементом его длину или после элемента специальную *сигнальную метку*, отмечающую его конец. В протоколе HTTP нельзя зарезервировать специальное значение для сигнальной метки, поскольку передаваемые элементы могут содержать графические изображения с произвольными последовательностями октетов. Таким образом, во избежание путаницы при выделении сигнальной метки из передаваемого потока данных в протоколе HTTP используется первый подход — отправка перед элементом его длины.

## 28.10. Длина данных, получаемых в результате работы программы

С точки зрения сервера подход, когда указывается длина элемента перед его отправкой, является не самым удачным. Причина в том, что иногда сервер не может узнать длину передаваемых данных до отправки. Чтобы понять, почему так происходит, давайте рассмотрим механизм *интерфейса общего шлюза* (Common Gateway Interface, или CGI), который позволяет запустить на сервере прикладную программу, создающую Web-страницу динамически. При получении запроса, соответствующего одной из динамически создаваемых Web-страниц, сервер запускает указанную в запросе CGI-программу. При этом данные, генерируемые программой, отсылаются клиенту в качестве ответа сервера. Процесс генерации динамических Web-страниц позволяет серверу передать клиенту текущую информацию (например, результаты игр спортивных команд). Однако чаще всего это означает, что сервер не может заранее определить точный размер отсылаемых данных. Сохранение данных в файле перед их отсылкой нежелательно

по двум причинам: затрачиваются ресурсы сервера и задерживается передача данных. Таким образом, для поддержки динамических Web-страниц в стандарте HTTP определено, что, если серверу не известна *заранее* длина передаваемого элемента, он должен сообщить броузеру, что по завершении передачи он закроет TCP-соединение. Вывод таков.

*Для того чтобы в протоколе HTTP по одному постоянному TCP-соединению можно было отсылать несколько запросов и ответов, сервер должен указывать броузеру перед каждым элементом его длину. Если сервер заранее не “знает” длины передаваемых данных, он должен сообщить об этом клиенту и после отправки данных закрыть TCP-соединение.*

## 28.11. Кодирование длины и заголовков

В каком виде сервер должен посылать броузеру информацию о длине передаваемых им данных? Интересно, что в протоколе HTTP для этой цели используется такой же формат, как и при передаче сообщений электронной почты (речь идет о формате “822” и расширениях MIME)<sup>4</sup>. Как и в сообщении электронной почты формата “822”, каждое передаваемое по протоколу HTTP сообщение содержит заголовок, который отделяется от передаваемых данных пустой строкой. Кроме того, каждая строка заголовка состоит из ключевого слова и значения, разделяемых двоеточием. В табл. 28.1 описаны часто используемые HTTP-заголовки. Тип и кодировка содержимого (заголовки Content-Type и Content-Encoding) соответствуют стандарту MIME.

**Таблица 28.1. Примеры заголовков HTTP-сообщения, посылаемых перед элементом данных**

Заголовок	Описание
Content-Length	Размер передаваемого элемента в октетах
Content-Type	Тип передаваемого элемента
Content-Encoding	Тип кодировки, используемой при передаче элемента
Content-Language	Язык, используемый в элементе

В качестве примера давайте рассмотрим листинг 28.1, в котором приведено несколько заголовков, использующихся при передаче HTML-документа через постоянное TCP-соединение. Обратите внимание, что строки заголовка отделяются от передаваемых данных (в данном случае обычного текстового сообщения) пустой строкой. Кроме того, поскольку передача выполняется через постоянное TCP-соединение, указывается заголовок Content-Length, содержащий длину передаваемых данных.

### Листинг 28.1. Пример передачи простого HTTP-сообщения

```
Content-Length: 34
Content-Language: en
Content-Encoding: ascii

<HTML> Простой пример передачи документа HTML. </HTML>
```

<sup>4</sup> Подробнее формат “822” и MIME-расширения описаны в главе 27, “Приложения: система электронной почты (SMTP, POP, IMAP, MIME)”.

Кроме заголовков, показанных в листинге 28.1, в стандарте протокола HTTP предусмотрено большое количество других заголовков, которые позволяют броузеру и серверу обмениваться служебной информацией. Например, выше было сказано, что, если серверу заранее не известна длина передаваемого элемента, то после передачи элемента он должен закрыть соединение. Однако сервер не может просто так закрыть соединение, не уведомив об этом броузер. Он сообщает броузеру, что следует ожидать закрытия соединения. Для этого перед отправкой элемента броузеру сервер помещает вместо заголовка Content-Length заголовок Connection:

```
Connection: close
```

Получив такой заголовок, броузер “понимает”, что сервер намеревается после передачи элемента закрыть соединение. При этом броузеру запрещается отсылать дальнейшие запросы по этому TCP-соединению. В следующих разделах будут описаны и другие HTTP-заголовки.

## 28.12. Согласование потенциальных возможностей программ

Кроме спецификации отсылаемого элемента, в протоколе HTTP предусматриваются специальные заголовки, позволяющие клиенту и серверу *согласовать* (*negotiate*) возможности используемых программ. В набор согласуемых элементов входит большое количество параметров, относящихся к особенностям работы как клиента, так и сервера. Среди них можно выделить следующие: характеристики соединения (например, необходимость авторизовать доступ клиента к серверу), параметры представления (например, способность клиента обрабатывать графические изображения в формате JPEG или используемые типы сжатия изображений), содержимого (например, язык представления текстовых файлов) и управления (например, на протяжении какого времени Web-страница может считаться актуальной).

Существует два основных типа согласования параметров: *управляемые сервером* и *управляемые агентом* (то есть броузером). Управляемый сервером процесс согласования параметров начинается с отправки запроса броузером. В запросе, кроме URL нужного элемента, указывается список предпочтаемых параметров. Получив запрос, сервер выбирает нужный элемент, представление которого удовлетворяет указанным параметрам броузера. Если запросу удовлетворяют несколько элементов, сервер делает выбор по принципу “наиболее подходящего” элемента. Например, если имеются копии документа на нескольких языках, а в запросе отдается предпочтение английскому языку, то сервер отоследит броузеру английскую версию.

Управляемое агентом согласование параметров означает, что процесс выбора происходит в два этапа. Сначала броузер отсылает запрос на сервер, чтобы узнать, какие возможности сервера он может использовать. Сервер возвращает броузеру список предоставляемых возможностей. Броузер выбирает одну из них и отсылает второй запрос для получения нужного элемента. Недостаток управляемого агентом процесса согласования параметров заключается в том, что для его выполнения необходимо обратиться к серверу два раза. Преимущество же состоит в том, что броузер сохраняет полный контроль над процессом выбора.

Для определения приемлемых форматов представления данных броузер использует HTTP-заголовок Accept. В нем перечисляются названия форматов и значение приоритета для каждого из них. Например, заголовок

```
Accept: text/html, text/plain; q=0.5, text/x-dvi; q=0.8
```

указывает, что для броузера предпочтительными являются текстовые документы в формате HTML. Однако в отсутствие таковых броузер примет документы в

формате `text/x-dvi`; в крайнем случае он примет обычный текстовый файл `text/plain`. Численные величины, связанные со вторым и третьим элементами, можно рассматривать как *уровень предпочтений*. Причем, если значение приоритета отсутствует, оно считается равным единице ( $q=1$ ). Значение  $q=0$  означает, что такой тип документа неприемлем. Для мультимедийных документов (например, аудиофайлов) параметр  $q$  (“quality”, или качество) имеет особое значение. Его можно интерпретировать как готовность броузера принять документ, качество которого ухудшено по сравнению с другими формами этого же документа на  $q$  процентов.

Существует несколько форм заголовка `Accept`, соответствующих описанным ранее заголовкам `Content`. Например, броузер может отослать серверу любой из следующих заголовков:

```
Accept-Encoding:  
Accept-Charset:  
Accept-Language:
```

Тем самым он определяет тип кодировок, наборы символов и языки документов, с которыми броузер может работать. Резюме такое.

*В протоколе HTTP для передачи служебной информации используются заголовки, подобные заголовкам MIME. Их передают и броузеры, и серверы. Это позволяет им согласовать параметры представления документа и тип кодировки, который нужно использовать.*

## 28.13. Условные запросы

В протоколе HTTP предусмотрена возможность для отправителя отсылать серверу *условный* запрос. Другими словами, посылая запрос на сервер, броузер включает в него специальный заголовок, в котором описываются условия, при выполнении которых запрос должен быть удовлетворен. Если указанное условие не выполняется, сервер игнорирует поступивший запрос. Условные запросы позволяют броузеру оптимизировать процесс выборки информации и избежать не нужных повторных передач документа. Запрос `If-Modified-Since` (*если документ изменился с указанного времени*) определяет одно из самых простых условий. Он позволяет броузеру избежать повторной передачи документа, если он не модифицировался после указанной даты. Например, броузер может включить в запрос `GET` следующий заголовок:

```
If-Modified-Since: Sat, 01 Jan 2000 05:00:00 GMT
```

Для сервера это означает, что не следует передавать броузеру документ, если он был создан или модифицирован до 5 утра по Гринвичу 1 января 2000 года.

## 28.14. Поддержка proxy-серверов

Proxy-серверы — важная часть службы Web. Они позволяют оптимизировать ее работу, сократить время ответа Web-сервера и уменьшить нагрузку на него. Однако для броузера работа proxy-сервера не является “прозрачной”. Это означает, что для работы с proxy-сервером броузер нужно настроить соответствующим образом. В результате все запросы на загрузку документов броузер будет посыпать не оригинальному Web-серверу, а своему локальному proxy-серверу. Обычно proxy-сервер выполняет роль большого кэша Web-страниц. Например, proxy-сервер можно запустить в локальной сети компании, сотрудники которой часто пользуются глобальной сетью Internet. При этом все броузеры компании должны

быть настроены так, чтобы запросы посыпались на корпоративный proxy-сервер. Когда пользователь компании в первый раз обращается к какой-либо Web-странице, proxy-сервер загружает ее копию с указанного пользователем сервера, размещает ее в локальном кэше и возвращает пользователю в качестве ответа на запрос. Если в следующий раз этот или другой пользователь обратится к той же Web-странице, proxy-сервер извлечет данные из локального кэша, не отсылая запрос по глобальной сети Internet. Следовательно, трафик между сетевым центром компании и глобальной сетью Internet значительно уменьшается.

Чтобы обеспечить корректную работу системы, в протокол HTTP были включены средства явной поддержки proxy-серверов. В нем точно определено, как proxy-сервер должен обрабатывать запросы, как должны интерпретироваться заголовки proxy-серверами, как браузер согласовывает параметры с proxy-сервером и как proxy-сервер согласовывает параметры с Web-сервером. Кроме того, несколько HTTP-заголовков было создано специально для использования proxy-серверами. Например, один из заголовков предназначен для аутентификации proxy-сервера на Web-сервере; другой — для фиксации адресов промежуточных proxy-серверов, через которые прошел запрошенный пользователем документ. Это дает возможность конечному получателю проанализировать список всех промежуточных proxy-серверов. Наконец, протокол HTTP позволяет Web-серверу управлять обработкой Web-страниц proxy-сервером. Например, Web-сервер может включить в ответное сообщение заголовок Max-Forwards, чтобы ограничить количество proxy-серверов, которые будут обрабатывать документ перед доставкой браузеру. Если Web-сервер задаст значение этого заголовка равным единице (как показано ниже),

```
Max-Forwards: 1
```

то обработать документ по пути от сервера к браузеру может только один proxy-сервер. Если же значение заголовка равно нулю, значит, сервер запрещает proxy-серверам обрабатывать документ.

## 28.15. Кэширование

Кэширование применяется для повышения эффективности работы системы Web. Оно позволяет уменьшить время ожидания ответа пользователем и сократить сетевой трафик за счет устранения ненужных передач. Одно из самых очевидных преимуществ кэширования заключается в промежуточном хранении Web-страниц. При первом обращении к Web-странице ее копию сохраняют на своих локальных дисках либо браузер, либо proxy-сервер, либо тот и другой. При последующих запросах той же Web-страницы время ее поиска существенно сокращается, поскольку браузер может получить ее копию из локального кэша, а не с Web-сервера.

Основная проблема во всех системах кэширования заключается во временных соотношениях. То есть речь идет о том, как долго документ должен сохраняться в кэше. С одной стороны, сохранение в кэше копии документа длительное время приводит к тому, что копия *устаревает*, то есть изменения в оригинале не будут отражены в кэшированной копии. С другой стороны, если кэшированную копию не сохранять достаточно долго, то это снижает эффективность кэширования, поскольку после удаления документа из кэш-памяти следующий запрос на выборку этого документа должен быть снова направлен на Web-сервер.

Протокол HTTP позволяет серверу управлять кэшированием двумя способами. Во-первых, когда сервер отвечает на запрос клиента и возвращает копию затребованной Web-страницы, он может определить параметры кэширования этой страницы: возможно ли кэширование страницы; может ли proxy-сервер кэшировать

эту страницу; идентификаторы групп, в которых кэшированная копия будет совместно использоваться; время устаревания кэшированной копии и ряд ограничений на преобразование копии Web-страницы. Во-вторых, протокол HTTP позволяет броузеру инициировать процесс *проверки актуальности* страницы. Для этого броузер отсылает запрос на выборку страницы и в специальном заголовке указывает, что максимальный “возраст” страницы (т.е. время, прошедшее с момента сохранения копии Web-страницы в кэше) не может быть больше нуля. Для ответа на подобный запрос не будут использоваться копии Web-страницы, сохраненные в одном из кэшей, поскольку “возраст” любой копии отличен от нуля. Таким образом, на запрос ответит только Web-сервер, где находится оригинал страницы. При этом промежуточные proxy-серверы, расположенные по пути передачи Web-страницы, и броузер, от которого исходил запрос, обновят ее копию в своем кэше. Можно сделать следующие выводы.

*Кэширование является залогом эффективной работы службы Web. Протокол HTTP позволяет Web-серверам определять, может ли страница кэшироваться и каким образом, а также устанавливать ее время жизни. Броузер может выдать запрос на загрузку Web-страницы в обход кэша и получить ее обновленную копию с сервера, на котором находится оригинал.*

## 28.16. Резюме

Система World Wide Web состоит из совокупности гипермедиа документов, хранящихся на множестве Web-серверов. Для доступа к этим документам используются специальные программы просмотра, называемые броузерами. Каждому Web-документу присваивается уникальный URL, который идентифицирует его в глобальной сети Internet. В URL закодирован тип протокола, использующийся для получения доступа к документу, адрес сервера в Internet и путь к документу на этом сервере.

Язык гипертекстовой разметки HTML предназначен для форматирования документа. Он позволяет встраивать в текст команды, управляющие процессом отображения документа в окне броузера. В языке HTML также допускается помещать внутрь документа ссылки на другие документы.

Для взаимодействия броузера и сервера используется специальный протокол передачи гипертекста — HTTP. HTTP — это протокол, относящийся к уровню приложений, в котором реализована поддержка согласования параметров между броузером и сервером, proxy-серверов, кэширования и постоянных соединений.

## Материал для дальнейшего изучения

Унифицированные указатели информационного ресурса (URL) описаны Бернерсом-Ли (Berners-Lee) и др. в [RFC 1768]. В других документах RFC содержатся различные дополнения и расширения. Вопросы, касающиеся места URL в доменной системе имен (DNS), рассматриваются Дэниелом (Daniel) и Миллингом (Mealling) в [RFC 2168].

Стандарт языка HTML версии 2 описан Бернерсом-Ли и Коннолли (Connolly) в [RFC 1866]. Процесс передачи HTML-формы на Web-сервер описан Небелом (Nebel) и Масинтером (Masinter) в [RFC 1867], а стандарт для оформления таблиц на языке HTML приведен Рэггетом (Raggett) в [RFC 1942].

В протокол HTTP версии 1.1 добавлено много новых возможностей, включая дополнительную поддержку постоянных соединений и кэширования. Эта версия описана Филдингом (Fielding) и др. в [RFC 2616]. Процесс аутентификации в протоколе HTTP рассмотрен Франксом (Franks) и др. в [RFC 2617].

## Упражнения

- 28.1. Прочитайте стандарт, в котором приведена спецификация URL. Что означает запись, состоящая из символа “#” и следующей за ней текстовой строки в конце URL?
- 28.2. Усложните предыдущее упражнение. Допустимо ли отправить на Web-сервер URL, в конце которого будет находиться знак “#”? Поясните свой ответ.
- 28.3. Как броузер отличает документ, содержащий HTML-разметку, от документа, содержащего произвольный текст? Чтобы выяснить это, попытайтесь с помощью броузера прочитать файлы разных типов. При интерпретации файла полагается ли броузер только на его имя или на содержимое?
- 28.4. Каково назначение команды протокола HTTP TRACE?
- 28.5. Каково различие между командами протокола HTTP PUT и POST? При каких условиях каждая из них может использоваться?
- 28.6. Когда используется заголовок протокола HTTP Keep-Alive?
- 28.7. Может ли любой Web-сервер выполнять функции прокси-сервера? Чтобы выяснить это, выберите произвольный Web-сервер и сконфигурируйте ваш броузер, чтобы он использовал этот сервер как прокси-сервер. Удивляют ли вас полученные результаты?
- 28.8. Прочитайте про директиву управления кэшем протокола HTTP must-revalidate. Приведите пример Web-страницы, на которой могла бы использоваться такая директива.
- 28.9. Как ответит Web-сервер, если броузер перед отправкой запроса не поместит заголовок протокола HTTP Content-Length?

# 29

## Приложения: передача голосовых и видеоданных по IP-сети (RTP)

### 29.1. Введение

В этой главе рассматривается передача данных (голосовых и видео) в реальном масштабе времени по IP-сети. Кроме протоколов, используемых для передачи таких данных, здесь будут описаны две более широких проблемы. Во-первых, мы рассмотрим использование IP-сети для организации коммерческой телефонной службы. Во-вторых, мы увидим, как маршрутизаторы в IP-сети могут обеспечить уровень обслуживания, достаточный для высококачественного воспроизведения видео- и аудиопотока.

Несмотря на то что протокол IP был создан и оптимизирован для передачи пакетов данных, он с самого начала успешно использовался для передачи аудио- и видеинформации. Фактически исследователи начали заниматься передачей голосовых данных еще по сети ARPANET, т.е. задолго до появления глобальной сети Internet. Уже к середине 90-х годов XX века многие коммерческие радиостанции транслировали свои передачи по глобальной сети Internet. Кроме того было создано специальное программное обеспечение, позволявшее отдельным пользователям пересыпать потоки аудиоданных по сети Internet или по обычной телефонной сети. Технология IP для передачи голоса также уже давно начала использоваться внутри коммерческих телефонных компаний.

### 29.2. Аудиоклипы и стандарты кодировки

Самый простой способ передачи голосовых данных по IP-сети состоит в том, что сначала аналоговый аудиосигнал переводится в цифровой вид и из него формируется файл данных. Затем этот файл передается по сети с помощью обычных протоколов получателю, после чего цифровой файл декодируется и из него воспроизводится оригинальный аналоговый сигнал. Естественно, что эта методика плохо работает в диалоговом режиме общения пользователей, поскольку кодирование и запись цифровых аудиоданных в файл и передача файла по сети связаны с большой задержкой. Поэтому описанный нами метод используется обычно для передачи коротких аудиозаписей, называемых *аудиоклипами*.

Для получения высококачественного цифрового звука используется специальное оборудование. Такое устройство называется *кодером-декодером* (*кодек*). Оно может выполнять преобразование аналогового аудиосигнала в цифровую форму, и наоборот. Чаще всего используют так называемый *кодер формы сигналов* (*waveform coder*), или *аналого-цифровой преобразователь* (АЦП). Он изменяет амплитуду входного аналогового сигнала через равные промежутки времени

и преобразует результат каждого замера в цифровое значение (то есть, целое число)<sup>1</sup>. При раскодировании кодек считывает последовательность целых чисел и по ней воссоздает непрерывный аналоговый сигнал, соответствующий цифровым значениям.

Существует несколько стандартов цифровой кодировки, выбранных по оптимальному соотношению между качеством воспроизведения аналогового сигнала и размером цифрового представления. Например, в цифровой телефонии используется стандарт *импульсно-кодовой модуляции* (*Pulse Code Modulation*, или *PCM*), в котором 8-битовые значения выборок считаются каждые 125 микросекунд (то есть, 8000 раз в секунду). Таким образом, в цифровой телефонии данные передаются со скоростью 64 Кбит/сек. При этом при использовании кодировки *PCM* размер получаемых цифровых данных неоптimalен — хранение 128 секундного аудиоклипа требует порядка одного мегабайта памяти.

Уменьшить количество данных, получаемых при цифровом кодировании, можно тремя способами: уменьшив частоту выборки, используя меньшее количество бит для кодирования каждой выборки либо сжимая цифровые данные. На практике применяется один или несколько способов одновременно. Поэтому существуют программы, которые кодируют аудиоданные со скоростью всего 2,2 Кбит/сек. Однако у каждой методики есть преимущества и недостатки. Главный недостаток при снижении частоты выборки или уменьшении разрядности кодирования выборки состоит в более низком качестве воспроизведения аудиосигнала. Другими словами, при этом сужается диапазон воспроизводимых частот. Основной недостаток при использовании систем сжатия состоит в задержке воспроизведения сигнала, поскольку для сжатия выходного цифрового потока требуется определенное время. Кроме того, для сокращения объема цифровых данных требуется выполнить больше математических вычислений. Поэтому для повышения степени сжатия нужно либо увеличить быстродействие центрального процессора, либо мириться с длительной задержкой. Поэтому высокая степень сжатия данных оправдана, если задержка не имеет особого значения (например, при сохранении цифровых данных, полученных на выходе кодека в файле).

### 29.3. Передача и воспроизведение аудио- и видеопотока

Большинство аудио- и видеоприложений должны работать в так называемом *реальном масштабе времени*. Это означает, что генерируемый ими поток цифровых данных должен быть своевременно отправлен и доставлен получателю<sup>2</sup>. Например, двухсторонняя цифровая телефонная связь является примером обмена данными в реальном масштабе времени, поскольку аудиоданные должны быть доставлены получателю без существенной задержки, иначе пользователи просто не смогут работать с этой системой. Своевременная передача данных в этом случае играет более важную роль, чем малое время задержки. Все дело в том, что на приемном конце невозможно воспроизвести результирующий сигнал, если порядок следования выборок или его временные характеристики будут нарушены. Таким образом, если отправитель делает выборку каждые 125 микросекунд, то получатель должен преобразовывать цифровые данные в аналоговый сигнал с такой же частотой.

<sup>1</sup> Альтернативой этому устройству является *речевой кодер/декодер* (*vocoder*), который распознает и кодирует человеческую речь, а не аналоговый сигнал произвольной формы.

<sup>2</sup> В данном случае своевременность доставки данных более важна, чем надежность, поскольку в подобных случаях потерянные данные могут быть просто пропущены.

Как же обеспечить передачу потока данных по сети с заданной отправителем скоростью? Решение было предложено разработчиками цифровой телефонной системы — речь идет об использовании так называемого *изохронного* подхода. Это означает, что вся система, включая ее цифровые схемы, должна проектироваться так, чтобы доставка цифрового потока данных выполнялась с той же тактовой частотой, которая была использована для его генерации (т.е. синхронно с отправителем). Таким образом, если в изохронной системе существует несколько путей между любыми двумя конечными точками, то система должна обеспечивать для всех путей одинаковое время задержки передачи данных.

Как известно, объединенная сеть на основе протокола IP не является изохронной. В предыдущих главах мы уже говорили о том, что в процессе доставки дейтаграммы могут дублироваться, задерживаться, или прибывать не в том порядке. Изменение времени задержки в процессе передачи данных называется *дрожанием (jitter)*. Это явление довольно распространено в IP-сетях. Следовательно, чтобы обеспечить разборчивую передачу и воспроизведение цифровых сигналов по IP-сети, в ней должны поддерживаться дополнительные протоколы. Для устранения дублирования дейтаграмм и нарушения порядка их доставки необходимо, чтобы в каждой передаваемой дейтаграмме содержался ее порядковый номер. Для устранения эффекта дрожания передаваемые дейтаграммы должны включать *временную метку*, сообщающую получателю, в какой момент времени должны воспроизводиться данные, находящиеся в пакете. Разделение информации о порядке следования пакетов и синхронизирующих данных позволяет получателю точно восстанавливать сигнал, независимо от того, как прибывают пакеты. Подобная информация о синхронизации особенно важна в случае потери дейтаграммы или при остановке отправителем передачи кодированных данных во время периодов тишины. Она позволяет получателю сделать паузу при воспроизведении на время, указанное во временных метках. Все сказанное выше можно подытожить так.

*Поскольку объединенная сеть на основе протокола IP не является изохронной, то для передачи цифровых данных в реальном масштабе времени требуется введение дополнительных протоколов. Кроме основной информации о порядковом номере пакета, позволяющей устраниить повторы пакетов или нарушение их порядка доставки, в каждом пакете должна находиться отдельная временная метка. Она сообщает получателю точное время воспроизведения находящихся в пакете данных.*

## 29.4. Дрожание и задержка воспроизведения

Как может получатель точно восстановить сигнал, если в сети возникает эффект дрожания? Для этого должен использоваться *буфер воспроизведения*<sup>3</sup> (рис. 29.1).

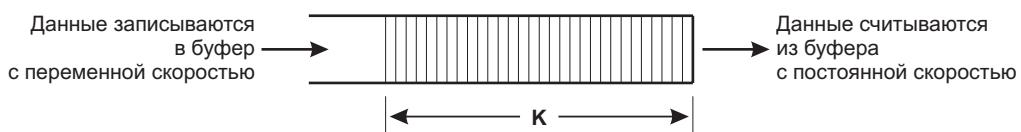


Рис. 29.1. Структурная схема работы буфера воспроизведения, компенсирующего эффект дрожания в сети. В буфере содержатся данные, длительность воспроизведения которых составляет  $K$  единиц времени

<sup>3</sup> Буфер воспроизведения еще называется *буфером дрожания (jitter buffer)*.

Перед началом сеанса работы получатель задерживает на некоторое время воспроизведение и помещает поступающие данные в буфер. Когда количество данных в буфере достигнет заранее заданного порогового значения, называемого *точкой воспроизведения* (*playback point*), начинается их считывание из буфера и воспроизведение. Точка воспроизведения, обозначенная рис. 29.1 как  $K$ , измеряется в единицах времени, которое будет затрачено на воспроизведение находящегося в буфере фрагмента данных. Таким образом, воспроизведение начнется после того, как получатель накопит такое количество данных, воспроизведение которых займет  $K$  единиц времени.

Дейтаграммы во время воспроизведения продолжают прибывать. Если дрожания в сети нет, то новые данные будут прибывать с такой же скоростью, с которой извлекаются и воспроизводятся находящиеся в буфере данные (т.е. в буфере будет всегда находиться точно  $K$  единиц времени невоспроизведенных данных). Если дейтаграмма поступит с небольшой задержкой, то на качество воспроизведения это никак не повлияет. По мере считывания данных из буфера их количество будет уменьшаться, а процесс воспроизведения будет непрерывно продолжаться в течение  $K$  единиц времени. Если за это время будут получены задержанные дейтаграммы, то количество данных в буфере снова восстановится.

Конечно, применение буфера воспроизведения не может компенсировать потерю дейтаграммы. В таких случаях воспроизведение в конечном счете достигает незаполненной позиции в буфере, и в выводе возникает пауза на период времени, соответствующий длительности отсутствующих данных. Очевидно, что выбор числа  $K$  обусловлен компромиссом между количеством потерянных дейтаграмм и временем их задержки<sup>4</sup>. Если значение числа  $K$  выбрать слишком маленьким, то даже небольшая величина дрожания приведет к быстрому исчерпанию данных в буфере воспроизведения (еще до прибытия следующей порции данных). Если число  $K$  окажется слишком большим, то система останется невосприимчивой к дрожанию, но дополнительная задержка, добавляемая к задержке передачи базовой сети, может оказаться для пользователей ощутимой. Несмотря на все недостатки в большинстве приложений, передающих данные в реальном масштабе времени по объединенной IP-сети, используется буферизация воспроизведения, поскольку она является единственным методом борьбы с дрожанием.

## 29.5. Протокол передачи данных в реальном масштабе времени (RTP)

Протокол, использующийся для передачи цифровых аудио- или видеосигналов по объединенной IP-сети, называется *протоколом передачи данных в реальном масштабе времени* (*Real-Time Transport Protocol*, или *RTP*). Интересно, что в протоколе RTP не предусмотрены средства, гарантирующие своевременную доставку дейтаграмм. Эта задача возложена на протоколы нижнего уровня. В протоколе RTP предусмотрена только регистрация порядкового номера в каждом пакете, что позволяет получателю обнаружить потерю дейтаграммы или факт нарушения порядка ее доставки, и регистрация временной метки, что дает возможность получателю управлять процессом воспроизведения.

Поскольку протокол RTP создавался для передачи разнообразных данных в реальном масштабе времени, включая аудио- и видеопотоки, формат его пакета не является фиксированным: каждый пакет начинается с заголовка фиксированного формата, а поля в заголовке определяют, как интерпретировать остальные поля

<sup>4</sup> Хотя на основании значения задержки в сети и величины дрожания можно динамически определить значение числа  $K$ , во многих системах с буферизацией при воспроизведении используют постоянное значение числа  $K$ .

заголовка и полезную информацию, находящуюся в пакете. Формат заголовка фиксированного формата протокола RTP показан на рис. 29.2.

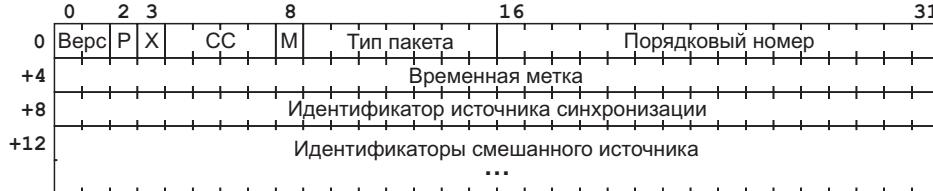


Рис. 29.2. Формат заголовка фиксированного формата протокола RTP. С этого заголовка начинается каждое сообщение. Точная интерпретация остальных полей пакета зависит от значения поля Тип пакета

Как показано на рис. 29.2, пакет начинается с 2-битового номера версии протокола RTP, помещаемого в поле *Верс*; текущий номер версии — 2. *Порядковый номер* пакета содержится в одноименном 16-битовом поле. Исходное значение порядкового номера в данном сеансе выбирается случайным образом. В некоторых приложениях используются дополнительные расширения заголовка, поля которого помещаются между фиксированным заголовком и полезной информацией пакета. Если в приложении используется расширение заголовка, то для того, чтобы определить, присутствует ли расширение в пакете, используется бит *X*. Интерпретация большинства остальных полей заголовка зависит от значения 7-битового поля *типа пакета*, которое определяет тип полезной информации, находящейся в пакете. Бит выравнивания (padding) *P* определяет, будет ли полезная информация пакета дополнена нулями. Эта информация нужна для программ шифрования, алгоритмы работы которых требуют, чтобы данные были распределены по блокам фиксированного размера. Интерпретация бита *M* (“маркер”) также зависит от конкретного приложения. Он используется приложениями, которые отмечают точки в потоке данных (например, начало каждого кадра при передаче видеопотока).

Значение поля *типа пакета* также влияет на интерпретацию поля *временной метки*. Она представляет собой 32-битовую величину, которая определяет время оцифровки первого октета цифровых данных. Начальное значение временной метки для текущего сеанса выбирается случайным образом. В стандарте определяется, что значение временной метки должно непрерывно увеличиваться даже в течение тех периодов времени, когда не регистрируется сигнал и не посылаются данные. Однако в стандарте не оговаривается значение величины приращения, или периода квантования — период квантования определяется типом данных, передаваемых в пакете. Другими словами, каждое приложение может выбрать частоту квантования, которая позволяет получателю воспроизвести данные с заданным качеством, выбранным исходным приложением. Например, при передаче аудиопотока по протоколу RTP, целесообразно выбирать величину логического кванта времени (или приращения временной метки) так, чтобы за один период тактового генератора делалась одна выборка<sup>5</sup>. Однако при передаче видеоизображения для достижения плавности воспроизведения частота квантования временной метки должна быть больше, чем один период тактового генератора на кадр. В стандарте допускается, чтобы значения временных меток в двух пакетах были одинаковыми, если данные, находящиеся в этих пакетах, были оцифрованы в одно и то же время.

<sup>5</sup> Временную метку иногда называют информационной временной меткой (*media timestamp*), чтобы подчеркнуть, что величина ее квантования зависит от типа передаваемого сигнала.

## 29.6. Потоки, микширование и многоадресатная передача

Одной из основных особенностей протокола RTP является поддержка процессов *трансляции потока* (т.е. изменения кодирования потока в промежуточном пункте) и *микширования потоков* (т.е. получения потоков данных из многих источников, объединения их в единый поток и отправки результата получателю). Чтобы понять необходимость микширования потоков, представьте, что несколько пользователей компьютеров хотят провести между собой сеанс конференц-связи по IP-сети. Для уменьшения количества аудиопотоков, пересылаемых по протоколу RTP, группа может использовать один из компьютеров в качестве *микшера* и потребовать, чтобы каждый из компьютеров установил с ним сеанс связи по протоколу RTP. В функции микшера входит объединение (микширование) аудиопотоков, и отправка результата в виде одного цифрового потока. Существует несколько методов микширования аудиопотоков: цифровой и аналоговый. В простейшем случае применяется аналоговый метод, т.е. получаемые цифровые потоки преобразуются в аналоговые сигналы, микшируются (т.е. смешиваются друг с другом), после чего результирующий сигнал снова переводится в цифровой вид и отправляется получателям.

Для поддержки процесса микширования и идентификации отправителя в пакете протокола RTP предусмотрены специальные поля. Поле, обозначенное на рис. 29.2 как *Идентификатор источника синхронизации*, определяет источник данных потока. Для каждого источника должен быть выбран уникальный 32-битовый идентификатор. В протокол RTP включен механизм для разрешения конфликтов значений идентификаторов, если они возникают. При объединении нескольких потоков в микшере он становится источником синхронизации для нового потока. Однако информация о первоначальных источниках, смешанных вместе, не теряется, потому что микшер запоминает их идентификаторы в поле переменной длины, которое обозначено на рис. 29.2 как *Идентификаторы смешанного источника*. Количество смешанных источников указывается в поле *CC (Contributing counter)*, размер которого составляет четыре бита. Таким образом, в процессе микширования может быть задействовано до 15 источников.

Протокол RTP поддерживает режим многоадресатной передачи протокола IP, поскольку наибольший эффект от процесса микширования можно получить при использовании именно этого режима передачи. Чтобы понять, почему так происходит, представьте себе телеконференцию с большим количеством участников. При одноадресной передаче требуется, чтобы сервер телеконференции посыпал по протоколу RTP копию каждого исходящего пакета каждому участнику конференции. Однако при использовании режима многоадресатной передачи сервер должен посыпать только одну копию пакета, который будет доставлен всем участникам телеконференции. Кроме того, при использовании микширования все источники могут передавать данные обычным образом микшеру, который объединит их в один поток и разошлет всем участникам телеконференции в многоадресатном режиме. Таким образом, применение микширования и многоадресатного режима передачи приводит к существенному уменьшению количества дейтаграмм, посылаемых по сети каждому участнику телеконференции.

## 29.7. Инкапсуляция RTP-пакетов

Название протокола RTP (протокол передачи данных в реальном масштабе времени) говорит о том, что он относится к протоколам транспортного уровня. Однако, если бы RTP работал как обычный транспортный протокол, каждое его сообщение должно было бы инкапсулироваться непосредственно в IP-дейтаграмму. В действительности протокол RTP является не совсем обычным

транспортным протоколом, поскольку в нем не используется прямая инкапсуляция в IP-дейтаграммы, хотя теоретически она возможна. Вместо этого в протоколе RTP в качестве протокола нижнего уровня используется протокол UDP. Другими словами, каждое RTP-сообщение инкапсулируется в UDP-дейтаграмму. Как известно, протоколы IP и UDP позволяют запускать на одном компьютере без взаимных помех несколько однотипных сетевых приложений. А это означает, что на одном компьютере может работать несколько приложений, использующих протокол RTP.

В отличие от большинства протоколов уровня приложений, рассмотренных выше, в протоколе RTP не используется зарезервированный номер порта протокола UDP. Вместо этого, номер порта выделяется перед каждым сеансом работы. Следовательно удаленное приложение должно быть проинформировано о выделенном номере порта. В соответствии с соглашением, в протоколе RTP выбирается порт UDP с четным номером. В следующем разделе будет показано, что в родственном протоколе RTCP используется следующий по порядку номер порта (т.е. нечетный).

## 29.8. Протокол управления в реальном масштабе времени (RTCP)

До этого момента описание процесса передачи данных в реальном масштабе времени было сосредоточено на возможностях самого протокола, позволяющих получателю воспроизводить содержимое. Однако есть другой, не менее важный аспект передачи данных в реальном масштабе времени: контроль работы базовой сети на протяжении всего сеанса и обеспечение механизма оперативной связи (*out of band*) между конечными точками. Такой механизм особенно важен, когда используются адаптивные методы передачи данных. Например, при перегрузке базовой сети приложение может уменьшить поток передаваемых данных за счет ухудшения качества воспроизведения сигнала (т.е. изменить кодирование аналогового сигнала). Другой пример: при изменении величины задержки или дрожания в сети получатель может соответствующим образом изменить размер своего буфера воспроизведения. Наконец, механизм оперативной связи может использоваться для передачи какой-либо информации параллельно с данными в реальном масштабе времени (например, субтитры, поясняющие передаваемый видеопоток).

Необходимые функции управления реализованы в виде родственного протокола, который является неотъемлемой частью протокола RTP. Он так и называется: *протокол управления в реальном масштабе времени (Real Time Control Protocol, или RTCP)*. Протокол RTCP позволяет отправителям и получателям передавать друг другу отчеты, содержащие дополнительную информацию о передаваемых данных и параметрах работы сети. Перед передачей RTCP-сообщения инкапсулируются в UDP-дейтаграммы<sup>6</sup> и отсылаются по номеру порта протокола, который на единицу больше, чем номер порта потока RTP, к которому они относятся.

## 29.9. Работа протокола RTCP

Для того чтобы отправители и получатели могли обмениваться информацией о сеансе, в протоколе RTCP используется пять типов сообщений. Они приведены в табл. 29.1. Каждое сообщение начинается с заголовка фиксированного формата, идентифицирующего его тип.

<sup>6</sup> Очень короткие RTCP-сообщения стандарт позволяет объединять для передачи в одной UDP-дейтаграмме.

---

**Таблица 29.1. Пять типов RTCP-сообщений**

---

<i>Тип</i>	<i>Описание</i>
200	Отчет отправителя
201	Отчет получателя
202	Сообщение, содержащее описание источника
203	Сообщение о завершении сеанса связи (bye)
204	Сообщение, определяемое самим приложением

---

Самыми простыми являются сообщения о завершении сеанса связи (bye) и сообщения, определяемые самим приложением. Отправитель передает сообщение bye при закрытии потока данных. Определенный приложением тип сообщения расширяет основные возможности протокола за счет введения новых типов сообщений. Например, для передачи субтитров, сопровождающих видеопоток, можно в приложении определить специальный тип сообщения RTCP.

Получатели периодически отправляют сообщения с *отчетами*, которые информируют источник передаваемых данных об условиях приема. Отчеты важны по двум причинам. Во-первых, они позволяют всем отправителям и получателям, участвующим в сеансе, узнавать об условиях приема у других получателей. Во-вторых, они позволяют получателям изменять частоту отправки отчетов, чтобы избежать перегрузки сети и источника данных. Использование адаптивного алгоритма работы гарантирует, что общий трафик управляющих сигналов будет меньше 5% от трафика передаваемых в реальном масштабе времени данных и что на отчеты получателя будет приходиться менее 75% трафика управляющих сигналов. Каждый отчет получателя относится к одному или нескольким источникам синхронизации и содержит отдельный раздел для каждого из них. В этом разделе указываются: самый больший порядковый номер пакета, полученный от источника; общее количество потерянных пакетов и их процентное соотношение к полученным пакетам; время, прошедшее с момента получения последнего RTCP-отчета от источника; величину дрожания, измеренную между получением двух последовательных отчетов.

Отправители периодически передают свои *отчеты*, содержащие абсолютную временную метку. Чтобы понять необходимость такой метки, напомним, что в протоколе RTP для каждого из потоков прикладная программа может самостоятельно выбрать величину приращения временной метки и что значение для первой метки выбирается случайным образом. Абсолютная временная метка, рассылаемая в отчетах отправителя необходима, поскольку она является единственным средством, позволяющим получателю *синхронизировать* несколько потоков. В частности, в протоколе RTP требуется, чтобы для передачи каждого типа информационного потока использовался отдельный поток данных. Это означает, что для передачи видеоизображения и его аудиосопровождения требуется два потока данных. Следовательно, информация об абсолютной временной метке позволяет получателю одновременно воспроизводить эти два потока.

Кроме периодической рассылки сообщений с отчетами, отправители также передают сообщения с *описанием источника*. В них указывается общая информация о пользователе, которому принадлежит источник или который им управляет. В каждом сообщении содержится один раздел для каждого исходящего RTP-потока. Его содержимое предназначено для чтения человеком. Например, в разделе описания источника требуется указать значение только одного обязательного

поля, содержащее *каноническое имя* владельца потока, представленного символьной строкой в форме:

*имя-пользователя*<sup>®</sup> *имя-узла*

Здесь вместо строки *имя-узла* подставляется доменное имя компьютера или его IP-адрес, представленный в точечной десятичной форме записи. Вместо строки *имя-пользователя* указывается регистрационное имя пользователя. Кроме обязательного поля, в сообщении с описанием источника предусмотрен ряд необязательных полей, содержащих дополнительную информацию, такую как адрес электронной почты пользователя (он может отличаться с канонического имени), номер телефона, почтовый адрес сетевого центра, название прикладной программы или утилиты, используемой для создания потока, а также другие текстовые заметки, относящиеся к источнику данных.

## 29.10. IP-телефония и система сигнализации

Из приведенного выше описания процесса передачи данных в реальном масштабе времени следует выделить один важный аспект. Речь идет об использовании протокола IP как основы для создания телефонной службы. Эта идея реализована многими телефонными компаниями в виде системы передачи голосовых данных по сети IP (*voice over IP*), называемой также *IP-телефонией* (IP telephony). При этом возникает вопрос: какие технологии необходимо дополнительно реализовать, чтобы можно было бы использовать сеть передачи данных на основе протокола IP вместо существующей изохронной телефонной системы? Хотя на этот вопрос не существует простого ответа, исследователи сосредоточили свои усилия в трех направлениях. Во-первых, было показано, что для корректной передачи цифрового сигнала через объединенную IP-сеть необходим специальный протокол наподобие RTP. Во-вторых, необходим механизм, с помощью которого можно было бы делать телефонные звонки (устанавливать и завершать связь с нужным абонентом). В-третьих, исследователи ищут средства превращения объединенной IP-сети в изохронную сеть.

В телефонной промышленности для описания процесса установки и завершения связи с абонентом (т.е. телефонного звонка) используется специальный термин — *система сигнализации* (*signaling system*). В частности, в обычной коммутируемой телефонной сети общего пользования (*Public Switched Telephone Network*, или *PSTN*) используется механизм передачи сигналов, называемый *системой общеканальной сигнализации №7*, или ОКС 7 (*Signaling System 7*, или *SS7*). Система SS7 предназначена для выполнения маршрутизации телефонного вызова и работает до того, как по проложенному каналу связи начнет передаваться какой-либо аудиосигнал. По заданному телефонному номеру она формирует в коммутируемой телефонной сети канал связи с указанным абонентом и передает ему сигнал вызова. После ответа абонента она соединяет его с помощью этого канала связи с абонентом, сделавшим вызов. Система SS7 также выполняет и другие функции, такие как *переадресация звонка* (*call forwarding*) и *обработка ошибок* (например, когда указанный телефонный номер занят).

Очевидно, уже понятно, что для выполнения телефонных звонков в IP-сети должны быть реализованы возможности сигнальной системы. Кроме того, чтобы система IP-телефонии была принята телефонными компаниями, она должна быть совместима с существующими телефонными стандартами. Другими словами, в системе IP-телефонии должны быть предусмотрены возможности взаимодействия с обычной телефонной системой на всех уровнях. Таким образом, нужно обеспечить возможность преобразования сигналов, используемых в системах IP-телефонии и в SS7, а также кодировку голосовых данных при потоковой передаче по IP-сети и в

стандарте РСМ. Вследствие этого у механизмов двух сигнальных систем будут реализованы эквивалентные функциональные возможности.

Самый общий подход к решению проблемы взаимодействия двух систем заключается в использовании *шлюза* (*gateway*) между системой IP-телефонии и обычной телефонной системой. При этом инициатор телефонного звонка может находиться по обе стороны шлюза. После получения запроса от сигнальной системы на телефонный звонок шлюз выполняет необходимое преобразование сигналов и перенаправляет запрос сигнальной системе получателя. Кроме того, шлюз должен также преобразовывать ответные сигналы и перенаправлять ответ инициатору звонка. Наконец, после обмена сигналами и установки канала связи между абонентами шлюз должен выполнять преобразование передаваемых голосовых данных из кодировки, используемой на одном конце соединения в кодировку, используемую на другом конце соединения.

Стандарты для IP-телефонии были предложены двумя группами. Международный телекоммуникационный союз (ITU), определил семейство протоколов, названное H.323, а инженерная группа IETF предложила протокол передачи сигналов, именуемый *протокол инициирования сеанса связи* (*Session Initiation Protocol*, или SIP). В следующих разделах эти два подхода будут рассмотрены в общих чертах.

### 29.10.1. Стандарты семейства H.323

Изначально ITU создавал стандарты семейства H.323 для передачи голосовых данных с помощью технологий локальных сетей. Затем этот набор стандартов был расширен, чтобы сделать возможной передачу голосовых данных по объединенной IP-сети. В настоящее время ожидается, что он будет поддержан всеми телефонными компаниями. H.323 не является отдельным протоколом, в нем объединено много протоколов с целью создания функциональной системы IP-телефонии. Например, кроме шлюзов в стандарте H.323 определяются специальные устройства, называемые *контроллерами зоны* (*gatekeeper*). Каждый контроллер зоны предоставляет точку подключения телефонов по протоколу IP. Чтобы получить разрешение на исходящий звонок и дать возможность телефонной системе правильно маршрутизировать входящие звонки, каждый IP-телефон должен быть зарегистрирован контроллером зоны. Необходимые для этого протоколы включены в стандарт H.323.

Кроме спецификации протоколов для передачи голосовых и видеоданных в реальном масштабе времени, в стандарте H.323 абонентам разрешается обмениваться обычными данными. В результате пары пользователей, участвующих в аудио- и видеоконференции, может совместно использовать электронную доску, а также посыпать изображения или обмениваться копиями документов.

Стандарт H.323 состоит из четырех основных протоколов, перечисленных в табл. 29.2.

**Таблица 29.2. Протоколы семейства H.323, используемые в IP-телефонии**

Протокол	Назначение
H.225.0	Передача сигналов, используемая для установки телефонного звонка
H.245	Передача сигналов управления и обратной связи, поступающих во время звонка
RTP	Передача данных в реальном масштабе времени (данные и синхронизация)
T.120	Обмен данными, связанными со звонком

Этот набор протоколов охватывает все аспекты создания системы IP-телефонии, включая регистрацию телефона, систему сигнализации, кодировку и передачу данных в реальном масштабе времени (как голосовых, так и видео), и функции управления.

На рис. 29.3 показаны взаимосвязи между протоколами семейства H.323. Как видно из рисунка, в семействе используются базовые протоколы UDP и TCP, запущенные поверх протокола IP.



*Рис. 29.3. Взаимосвязи между протоколами, которые включены в стандарт ITU H.323 для IP-телефонии*

### 29.10.2. Протокол инициирования сеанса связи (SIP)

Группа IETF предложила альтернативу стандарту H.323 — *протокол инициирования сеанса связи* (*Session Initiation Protocol*, или *SIP*). Протокол SIP описывает только систему сигнализации. В нем нет никаких рекомендаций по поводу использования специальных кодеков или протокола RTP для передачи данных в реальном масштабе времени. Таким образом, протокол SIP не является полноценной заменой стандарта H.323, поскольку в нем не предусмотрены все функциональные возможности последнего.

В протоколе SIP используется метод взаимодействия типа клиент/сервер, причем серверы могут быть двух типов. *Сервер пользовательского посредника* (*user agent server*) запускается в SIP-телефоне. Ему присваивается идентификатор (например, *user@site*), и он может принимать входящие звонки. Второй тип сервера — *промежуточный* (т.е. находящийся между двумя SIP-телефонами). Он служит для выполнения ряда задач, типа установки коммутируемого канала связи и переадресации звонка. По сути промежуточный сервер выполняет роль *роху-сервера*, который может переадресовывать входящий запрос на соединение следующему роху-серверу, расположенному по пути к вызываемому абоненту, либо роль *сервера переадресации*, который сообщает вызывающему абоненту, как достичь нужного адресата.

Для представления информации о звонке в протоколе SIP используется сопутствующий ему протокол, который называется *протоколом описания сеанса связи* (*Session Description Protocol*, или *SDP*). Протокол SDP особенно важен при организации конференц-связи, поскольку в этом случае участники присоединяются к конференции и отключаются от нее динамически. С помощью протокола SDP определяются такие детали, как тип кодировки потока, номеров портов протоколов и адреса группового вещания.

## 29.11. Резервирование ресурсов и качество обслуживания

Под термином *качество обслуживания* (*Quality Of Service*, или *QoS*) мы будем понимать набор статистических характеристик, которые гарантированно может обеспечить сетевая система, — величину потерь, задержку, пропускную

способность и дрожание. Считается, что изохронная сеть, спроектированная для достижения требуемой пропускной способности, обеспечивает гарантированное качество обслуживания, в то время как сеть с коммутацией пакетов, не гарантирующая доставку пакетов, не обеспечивает гарантированного качества обслуживания. Необходимо ли обеспечивать гарантированное качество обслуживания при передаче голосовых и видеоданных в реальном масштабе времени по протоколу IP? Если да, то как это должно быть реализовано? Вокруг этих двух вопросов и ведется основная дискуссия. С одной стороны, проектировщики, разрабатывающие телефонную систему, настаивают на том, что для качественного воспроизведения голоса требуется, чтобы для каждого телефонного звонка базовая сетевая система обеспечивала гарантированное качество обслуживания в отношении величины задержки и количества потерь пакетов. С другой стороны, проектировщики, разрабатывающие сети на основе протокола IP, уверяют, что глобальная сеть Internet работает достаточно хорошо и без обеспечения гарантированного качества обслуживания и что недопустимо для каждого потока обеспечивать гарантированное качество обслуживания, поскольку из-за наличия маршрутизаторов система становится слишком дорогой и медленной.

Результатом разногласий относительно качества обслуживания стали многочисленные предложения, реализации и эксперименты. Хотя в глобальной сети Internet требуемое качество обслуживания и не обеспечивается, тем не менее она уже давно используется для пересылки аудиопотоков. Чтобы обеспечить необходимое качество обслуживания для каждого отдельного соединения, следует применять сетевые технологии, которые используются при разработке телефонной системы, и производные от них. Кроме того, в главе 7, “Протокол IP: доставка дейтаграмм без установки соединения”, было сказано, что инженерная группа IETF выбрала консервативный подход в виде схемы *дифференцированного обслуживания*, которая позволяет разделить трафик на отдельные классы в соответствии с требуемым качеством обслуживания. В схеме дифференцированного обслуживания, благодаря отказу от управления системой на низком уровне, процесс пересылки пакетов данных значительно упрощен.

## **29.12. Качество обслуживания, загрузка и максимальная пропускная способность сети**

Споры вокруг качества обслуживания напоминают дебаты, проводившиеся когда-то относительно распределения ресурсов операционной системы. Речь тогда шла о стратегиях операционной системы в отношении распределения памяти и времени центрального процессора. Основная проблема заключается в степени загрузки системы, поскольку если сеть располагает достаточными ресурсами для передачи всего трафика, то не нужно вводить ограничений, касающихся качества обслуживания. Если же реальный трафик превышает пропускную способность сети, то удовлетворить все потребности пользователей не сможет никакая система качества обслуживания. Другими словами, если величина загрузки сети составляет всего 1% от ее пропускной способности, то система качества обслуживания попросту не нужна. Если же загрузка сети составляет 101%, то при любой системе качества обслуживания такая сеть будет давать сбои. Сторонники, приводящие доводы в пользу тех или иных механизмов качества обслуживания, утверждают, что они нужны для достижения двух целей. Во-первых, разделяя существующие ресурсы среди большого количества пользователей, они делают систему более “честной”. Во-вторых, благодаря ограничению трафика со стороны каждого пользователя они позволяют увеличить процент загрузки сети без опасности возникновения в ней затора.

Один из главных аргументов против сложных механизмов качества обслуживания основан на том, что современные сетевые технологии в состоянии обеспечить достаточно высокую скорость передачи данных. В самом деле, пропускная способность современных сетей передачи данных существенно увеличилась. До тех пор пока продолжается быстрый рост пропускной способности сети, поддержка сложных механизмов качества обслуживания приводит к увеличению накладных расходов. Однако если величина трафика будет расти быстрее, чем величина пропускной способности сети, то поддержка механизмов качества обслуживания может стать чисто экономической проблемой. Например, устанавливая более высокие цены за поддержку более высокого качества обслуживания, провайдеры Internet могут ограничивать величину трафика пользователя за счет стоимости своих услуг.

## 29.13. Протокол RSVP

Если все же необходимо обеспечить заданный уровень качества обслуживания, то как это можно сделать в рамках сети на основе протокола IP? Перед тем как анонсировать схему дифференцированного обслуживания инженерная группа IETF работала над проблемой обеспечения заданного качества обслуживания в рамках IP-сети. В ходе этой работы было создано два протокола: *протокол резервирования ресурсов (Resource Reservation Protocol, или RSVP)* и *протокол общей открытой службы правил (Common Open Policy Services, или COPS)*.

Поддержка качества обслуживания в IP-сетях не может быть просто добавлена на уровне приложений. Для этого нужно изменить основную инфраструктуру базовой сети — маршрутизаторы должны зарезервировать определенные ресурсы (такие как полоса пропускания канала связи) для передачи каждого потока данных между парой конечных точек. При этом существуют две особенности. Во-первых, перед отправкой данных конечные точки должны запросить необходимые ресурсы. При этом все маршрутизаторы, расположенные по пути следования пакетов между этими точками, должны предоставить запрошенные ресурсы. Эту процедуру можно рассматривать как одну из форм сигнализации. Во-вторых, по мере прохождения дейтаграмм в потоке данных, маршрутизаторы должны ограничивать величину трафика. Такое *ограничение трафика (traffic policing)*, необходимо для того, чтобы величина пересылаемого трафика не выходила за указанные границы. Управление организацией очередей и пересылкой данных необходимо по двум причинам. Маршрутизатор должен придерживаться определенных правил организации очереди, благодаря чему исключаются непредусмотренные отклонения величины задержки. Кроме того, маршрутизатор также должен сглаживать выбросы, которые иногда происходят при отправке большого количества пакетов в единицу времени. Последнюю операцию иногда называют *выравниванием трафика (traffic shaping)*. Она позволяет сгладить *пульсации* трафика в сети. Например, при средней скорости потока данных 1 Мбит/с, может возникнуть ситуация, когда на протяжении одной миллисекунды величина трафика будет составлять 2 Мбит/с, а на протяжении другой миллисекунды — 0 Мбит/с. Маршрутизатор может сгладить этот выброс за счет организации временной очереди для приходящих дейтаграмм, из которой они будут отсылаться с равномерной скоростью 1 Мбит/с.

Протокол RSVP предназначен для обработки запросов на резервирование ресурсов и рассылки ответов на них. Он не является протоколом маршрутизации и устанавливает правила только для организации потока данных между двумя конечными точками. Протокол RSVP используется перед отправкой данных. Чтобы инициировать сквозной поток передачи данных, конечные пункты сначала посыпают RSVP-сообщение, в котором определяется путь к получателю. При

этом в дейтаграмме, содержащей это сообщение, устанавливается специальный параметр, запрашивающий внимание маршрутизатора (router alert). Он гарантирует, что маршрутизаторы “обратят внимание” на это сообщение и проанализируют его содержимое. Получив RSVP-ответ с определением пути, конечный пункт посыпает запрос на резервирование ресурсов для данного потока. В запросе указывается желаемый диапазон качества обслуживания. При этом каждый маршрутизатор, пересылающий запрос получателю, должен обеспечить у себя резервирование ресурсов, указанных в запросе. Если какой-либо маршрутизатор, расположенный по пути следования запроса, не может этого сделать, он отклоняет запрос и с помощью протокола RSVP посыпает отрицательный ответ отправителю. Если все маршрутизаторы, расположенные по пути следования запроса, могут его удовлетворить, то отправителю возвращается положительный RSVP-ответ.

Каждый поток RSVP является *симплексным* (т.е. односторонним). Если какое-либо из приложений требует гарантированного качества обслуживания в двух направлениях, то запрос на резервирование ресурсов по протоколу RSVP должны послать оба конечных пункта. Поскольку маршрутизация RSVP-пакетов выполняется так же, как и любых других дейтаграмм (т.е. через существующую систему маршрутизации), нет никакой гарантии, что оба потока пройдут через одни и те же маршрутизаторы. Поэтому если запрос на требуемое качество обслуживания был одобрен в одном направлении, нельзя гарантировать, что он также будет одобрен и в другом направлении. Все сказанное выше можно подытожить так.

*Конечный пункт посылает RSVP-запрос по объединенной IP-сети на резервирование ресурсов для симплексного потока, в котором указывает желаемый диапазон качества обслуживания. Если все маршрутизаторы, расположенные по пути следования запроса, могут его удовлетворить, то отправителю возвращается положительный RSVP-ответ. В противном случае присыпается отрицательный ответ. Если какое-либо из приложений требует гарантированного качества обслуживания в двух направлениях, то запрос на резервирование ресурсов по протоколу RSVP должны послать оба конечных пункта.*

## 29.14. Протокол COPS

Получив RSVP-запрос, маршрутизатор должен оценить его с точки зрения выполнимости (т.е. имеются ли у маршрутизатора ресурсы, необходимые для удовлетворения запроса) и непротиворечивости (т.е. не противоречит ли запрос принятым правилам и ограничениям). Оценку выполнимости маршрутизатор может сделать самостоятельно, поскольку он сам может решить, как распределить ширину полосы канала связи, память и ресурсы центрального процессора, доступные локально. Однако оценка непротиворечивости требует глобального подхода, поскольку одни и те же правила должны выполнять все маршрутизаторы.

Для реализации концепции глобальных правил в предложенной IETF структуре используется двухуровневая модель, причем взаимодействие между уровнями выполняется по принципу клиент/сервер. Когда маршрутизатор получает RSVP-запрос, он становится клиентом и посыпает запрос серверу, который называется *точкой принятия решений* (*Policy Decision Point*, или *PDP*), чтобы определить, соответствует ли запрос принятым правилам и ограничениям. PDP не обрабатывает трафик, а просто проверяет, удовлетворяют ли запросы глобальным правилам. Если PDP санкционирует данный запрос, то маршрутизатор

должен его выполнить. В этом случае он функционирует как *точка выполнения решений* (*Policy Enforcement Point*, или *PEP*) и гарантирует, что трафик не будет выходить за рамки принятых правил.

Протокол COPS определяет механизм взаимодействия типа клиент/сервер между маршрутизатором и PDP (или между маршрутизатором и локальным PDP, если используется несколько уровней серверов принятия решений). Хотя в протоколе COPS определен уникальный формат заголовка сообщения, его основной формат имеет много общего с пакетом протокола RSVP. В частности, в протоколе COPS используется тот же формат, что и в RSVP для индивидуальных элементов в запросе. Таким образом, когда маршрутизатор получает RSVP-запрос, он может извлечь элементы правил, поместить их в сообщение COPS и послать результат серверу принятия решения.

## 29.15. Резюме

Аналоговые данные (например, звуковой сигнал) можно закодировать в цифровой форме. Для этого применяется специальное оборудование, называемое кодеком. В телефонии для представления цифровых голосовых данных используется специальный стандарт импульсно-кодовой модуляции. При этом цифровые данные генерируются со скоростью 64 Кбит/с.

Для передачи данных по IP-сети в реальном масштабе времени используется протокол RTP. В каждом RTP-сообщении содержится два важных элемента: порядковый номер, используемый получателем для упорядочивания сообщений и обнаружения потерянных дейтаграмм, и временная метка, используемая получателем для определения момента, когда надо воспроизвести закодированные данные. Для предоставления информации об источниках и обеспечения возможности объединения нескольких потоков в мишене используется смежный протокол управления, RTCP.

В настоящий момент продолжается спор вокруг того, нужно ли для передачи данных в реальном масштабе времени обеспечивать гарантированное качество обслуживания. Перед тем как анонсировать схему дифференцированного обслуживания, инженерная группа IETF создала два протокола, которые могут использоваться для обеспечения заданного уровня качества обслуживания для каждого потока данных. Для запроса заданного уровня качества обслуживания для конкретного потока конечные пункты используют протокол RSVP. Промежуточные маршрутизаторы принимают или отклоняют этот запрос. После получения RSVP-запроса маршрутизатор подключается к точке принятия решений с помощью протокола COPS и проверяет, соответствует ли запрос принятым правилам и ограничениям.

## Материал для дальнейшего изучения

Стандарт для протоколов RTP и RTCP описан Шульцринном (Schulzrinne) и др. в [RFC 1889]. Передача избыточных аудиоданных по протоколу RTP описана Перкинсом (Perkins) и др. в [RFC 2198]. Использование протокола RTP для проведения аудио- и видеоконференций описывается Шульцринном в [RFC 1890]. Связанный протокол RTSP, используемый для потоковой передачи данных в реальном масштабе времени, описывается Шульцринном, Rao (Rao) и Ланфиером (Lanphier) в [RFC 2326].

Спецификация протокола RSVP приводится Жангом (Zhang) и др. в [RFC 2205]. Протокол COPS описывается Бойлем (Boyle) и др. в [draft-rap-cops-06.txt].

## Упражнения

- 29.1. Ознакомьтесь со спецификацией протокола потоковой передачи данных в реальном масштабе времени (RTSP). Каковы главные различия между протоколами RTSP и RTP?
- 29.2. Известно, что ширина полосы пропускания часто приводится как пример того, что может гарантированно обеспечить механизм качества обслуживания. Докажите, что величина задержки является более важным ресурсом. (*Подсказка.* Какое ограничение может быть снято при наличии достаточного количества денег?)
- 29.3. Что будет делать программа поддержки протокола, если она получит RTP-сообщение с гораздо большим порядковым номером, чем ожидается? Почему?
- 29.4. Действительно ли в протоколе RTP необходимы порядковые номера, или вместо них можно использовать временную метку? Поясните свой ответ.
- 29.5. Отдали бы вы предпочтение объединенной сети, в которой для всего трафика требовалось бы обеспечить гарантированное качество обслуживания? Почему?
- 29.6. Измерьте степень использования канала связи, через который ваш компьютер подключен к глобальной сети Internet. Если бы для всего трафика требовалось обеспечить гарантированное качество обслуживания, то улучшилось или ухудшилось бы качество его работы? Поясните свой ответ.

# 30

## Приложения: управление объединенной сетью (SNMP)

### 30.1. Введение

Помимо протоколов, которые обеспечивают работу служб сетевого уровня, и использующих эти службы прикладных программ, в объединенной сети необходимо программное обеспечение, позволяющее администраторам устранять возникшие проблемы, управлять маршрутизацией и обнаруживать компьютеры, нарушающие стандарты протоколов. Такие действия обобщенно называют *управлением объединенной сетью*. В этой главе рассмотрены принципы, лежащие в основе работы программного обеспечения для управления объединенной сетью TCP/IP, а также описан протокол, используемый в этих целях.

### 30.2. Уровень протоколов управления

Изначально протоколы управления входили в состав протоколов канального уровня многих глобальных сетей. Если работоспособность одного из коммутаторов пакетов нарушалась, сетевой администратор мог изменить режим его работы, послав коммутатору специальный *управляющий пакет* (*control packet*). После получения этого пакета коммутатор приостанавливал нормальную работу и переходил в режим ответа на поступающие от администратора команды. В результате администратор мог проанализировать состояние коммутатора и обнаружить проблему, просмотреть или изменить таблицу маршрутизации, проверить состояние одного из коммуникационных интерфейсов или перезагрузить коммутатор. Устранив неполадку администратор переключал коммутатор в режим нормальной работы. Поскольку средства управления входили в состав протокола низшего уровня, администраторы зачастую могли управлять коммутаторами, даже если происходили сбои в протоколах высшего уровня.

В отличие от однородной глобальной сети, в объединенной сети TCP/IP нет единого протокола канального уровня. Объединенная сеть состоит из нескольких физических сетей, соединенных между собой IP-маршрутизаторами. Поэтому управление такой объединенной сетью отличается от управления обычной сетью. Во-первых, один администратор чаще всего должен управлять устройствами разных типов, включая IP-маршрутизаторы, мосты, модемы, рабочие станции и принтеры. Во-вторых, в управляемых объектах может не использоваться общий протокол канального уровня. В-третьих, совокупность машин, управляемых администратором, может размещаться в любых точках объединенной сети. В частности, администратору может понадобиться управлять одной или несколькими машинами, которые не подключены к той же физической сети, что и компьютер

самого администратора. Следовательно, может случиться так, что администратор не сможет взаимодействовать по сети с управляемыми им машинами, если в программе управления не будут использоваться протоколы, позволяющие установить сквозное соединение между двумя машинами через объединенную сеть. Вследствие этого протокол управления объединенной сетью, используемый совместно с семейством протоколов TCP/IP, должен функционировать на уровне, расположенному выше транспортного уровня. Вывод таков.

*В объединенной сети TCP/IP администратору необходимо следить за работой и управлять маршрутизаторами, а также другими сетевыми устройствами. Поскольку такие устройства могут быть подключены к любым сетям, используемые для управления объединенной сетью протоколы должны функционировать на уровне приложений и взаимодействовать через один из протоколов транспортного уровня семейства TCP/IP.*

Разработка программного обеспечения для управления объединенной сетью, которое функционирует на уровне приложений, имеет несколько преимуществ. Поскольку при этом можно создавать программы поддержки протоколов, которые не зависят от сетевого аппаратного обеспечения, один набор протоколов можно использовать для всех типов сетей. Поскольку эти программы не зависят также от типа аппаратного обеспечения управляемой машины, одни и те же протоколы можно использовать для всех управляемых устройств. Для администратора возможность использования одного набора протоколов управления означает унификацию устройств в сети — для всех маршрутизаторов применяется одинаковый набор команд. Более того, поскольку для взаимодействия управленческое программное обеспечение использует протокол IP, администратор может управлять маршрутизаторами, разбросанными по всей объединенной сети TCP/IP, не подключаясь напрямую к каждой физической сети или маршрутизатору.

Конечно, создание управленческого программного обеспечения, функционирующего на уровне приложений, не лишено недостатков. Администратору скорее всего не удастся связаться с маршрутизатором, которым необходимо управлять, если его операционная система, программное обеспечение протокола IP или протокола транспортного уровня будут работать некорректно. Например, если таблица маршрутизации произвольного маршрутизатора повреждена, исправить в ней ошибки или перезагрузить машину с удаленного компьютера будет невозможно. Если операционная система маршрутизатора перестает нормально функционировать, получить доступ к прикладной программе, реализующей протоколы управления объединенной сетью, будет также невозможно, даже если маршрутизатор все еще может реагировать на аппаратные прерывания и маршрутизировать пакеты.

### 30.3. Структурная модель

Несмотря на потенциальные недостатки, на практике идея обеспечения работы управленческого программного обеспечения семейства протоколов TCP/IP на уровне приложений оказалась довольно эффективной. Самое главное преимущество идеи — поместить протоколы сетевого управления на высший уровень становится очевидным при рассмотрении большой объединенной сети, в которой компьютеру администратора не нужно напрямую подключаться ко всем физическим сетям, содержащим управляемые объекты. На рис. 30.1 показана структура такой объединенной сети. Администратор запускает клиентскую программу управления (management client, или MC), которая связывается с программами — агентами управления (management agent, или MA), запущенными на устройствах всей объединенной сети

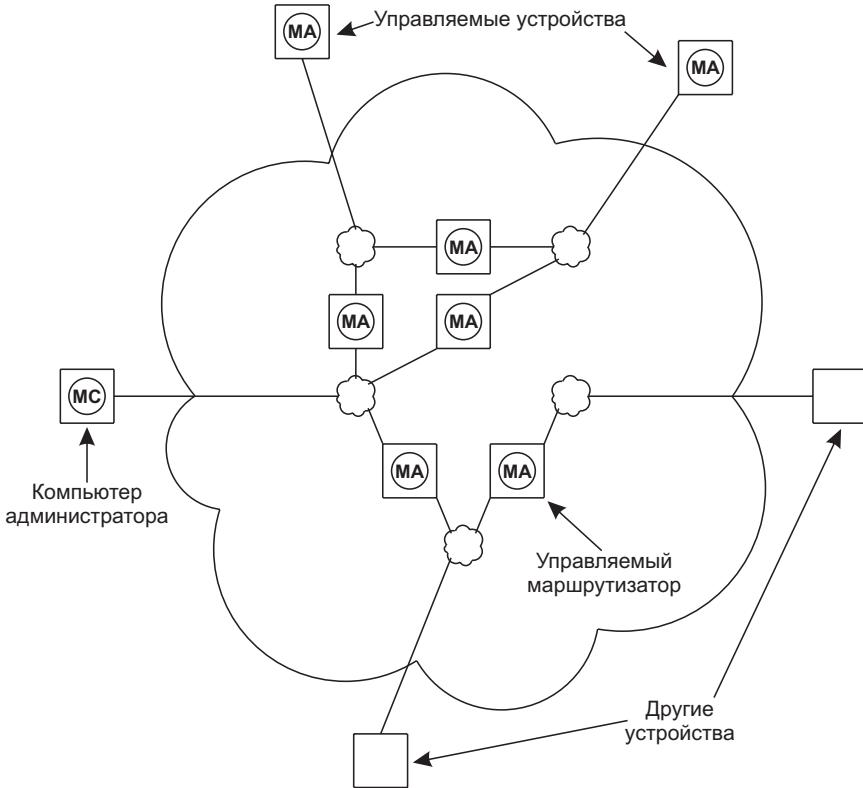


Рис. 30.1. Пример модели сетевого управления

Как показано на рис. 30.1, клиентские программы обычно запускаются на рабочей станции администратора. Программа-сервер запускается на каждом маршрутизаторе или узле сети<sup>1</sup>, которые вовлечены в процесс управления. Формально программное обеспечение сервера называется *агентом управления* (*management agent*) или просто *агентом*. Администратор запускает клиентскую программу на локальном компьютере и указывает, с каким из агентов она должна вступить во взаимодействие. Связавшись с агентом, клиентская программа посыпает либо запросы на необходимую информацию, либо команды на изменение режима работы маршрутизатора. Конечно, не все устройства в большой объединенной сети находятся под управлением одного администратора. Обычно администраторы управляют только устройствами, находящимися в их локальных сетевых центрах. В большом сетевом центре управление может осуществляться одновременно несколькими администраторами.

В программном обеспечении для управления объединенной сетью используется механизм аутентификации, благодаря чему только уполномоченные администраторы могут получить доступ к определенному устройству и управлять им. В некоторых протоколах управления поддерживается несколько уровней авторизации, что позволяет администратору получить заданные права доступа к каждому устройству. Например, маршрутизатор можно сконфигурировать так, чтобы не-

<sup>1</sup> Вспомним, что под термином *узел сети*, употребляемым в стандарте семейства протоколов TCP/IP, может пониматься как устройство (например, принтер), так и обычный компьютер.

сколько администраторов могли получать от него информацию, но при этом только один из них мог изменять ее или управлять маршрутизатором.

## 30.4. Структура протоколов

В протоколах сетевого управления<sup>2</sup>, входящих в семейство протоколов TCP/IP, проблема, связанная с процессом управления, разделена на две части, для каждой из которых приняты свои стандарты. Первая часть связана с обменом информацией. В протоколе определено, каким образом клиентские программы, запущенные на компьютере администратора, взаимодействуют с агентом. В протоколе определен также формат и назначение сообщений, которыми обмениваются клиенты и серверы, а также форма имен и адресов. Вторая часть связана с данными, которые обрабатывает клиентская программа, запущенная на компьютере администратора. В протоколе определены элементы данных, которые должны храниться на управляемом устройстве, а также имя каждого элемента данных и используемый для представления имени синтаксис.

### 30.4.1. Стандартный протокол сетевого управления

В семействе протоколов TCP/IP в качестве стандарта для сетевого управления используется *простой протокол сетевого управления* (*Simple Network Management Protocol*, или *SNMP*). В процессе разработки было создано три версии этого протокола. Поэтому текущая версия обозначается *SNMPv3*, а предшествующие ей версии — *SNMPv1* и *SNMPv2*. Отличия между ними незначительны: все три версии имеют одинаковую общую структуру, и большинство их возможностей совместимо сверху вниз.

Помимо определения деталей, таких как формат сообщений и используемые транспортные протоколы, в стандарте протокола SNMP определен набор операций и описана каждая из них. Далее мы убедимся, что используемый подход основан на принципе минимализма, т.е. все функциональные возможности протокола обеспечивается с помощью нескольких операций.

### 30.4.2. Стандарт для управляющей информации

В управляемом устройстве должна храниться соответствующая управляющая информация, а также информация о состоянии, доступ к которой может получить администратор. Например, на маршрутизаторе хранятся статистические данные, содержащие информацию о состоянии его сетевых интерфейсов, количество входящих и исходящих пакетов, числе потерянных дейтаграмм, а также количество генерированных сообщений об ошибках. В модеме хранятся статистические данные о количестве отправленных и полученных символов, средней скорости передачи данных в бит/с и числе поступивших звонков. Хотя протокол SNMP позволяет администратору получить доступ к этим статистическим данным, в нем явно не определено, к каким данным и на каких устройствах можно получить доступ. Все подробности для каждого типа устройства определены в отдельном стандарте. В стандарте, который называется *база управляющей информации* (*Management Information Base*, или *MIB*), определены элементы данных, которые должны храниться в управляемом устройстве, операции, которые можно выполнять на каждом устройстве, а также их назначение. Например, в базе MIB для

<sup>2</sup> Формально существует отличие между протоколами управления объединенной сетью (*internet management protocols*) и протоколами сетевого управления (*network management protocols*). Однако с самого начала протоколы управления объединенной сетью TCP/IP называли протоколами *сетевого управления*, поэтому мы будем придерживаться принятой терминологии.

протокола IP определено, что программное обеспечение должно вести подсчет всех октетов, поступающих через каждый сетевой интерфейс, а также то, что программы сетевого управления могут только считывать данные счетчиков.

В базе MIB для семейства протоколов TCP/IP управляющая информация разделена на большое количество категорий. Выбор соответствующей категории очень важен, поскольку используемые для определения элементов данных идентификаторы включают в себя код определенной категории. В табл. 30.1 приведены некоторые из этих категорий. Каждая категория кодируется в виде идентификатора, который используется для определения объекта данных.

**Таблица 30.1. Категории базы MIB**

<i>Категория</i>	<i>Содержит информацию об ...</i>
system	операционной системе узла сети или маршрутизатора
interfaces	отдельных сетевых интерфейсах
at	преобразовании адресов (например, выполняемые протоколом ARP)
ip	программном обеспечении протокола IP
icmpr	программном обеспечении протокола межсетевых управляющих сообщений (ICMP)
tcp	программном обеспечении протокола управления передачей (TCP)
udp	программном обеспечении протокола передачи пользовательских дейтаграмм (UDP)
ospf	программном обеспечении открытого протокола поиска кратчайших маршрутов (OSPF)
bgr	программном обеспечении протокола граничного шлюза (BGP)
rmon	удаленном сетевом мониторинге
rip-2	программном обеспечении протокола маршрутной информации (RIP2)
dns	программном обеспечении системы доменных имен (DNS)

То, что определение элементов данных базы MIB не зависит от используемого протокола сетевого управления, выгодно как производителям сетевого оборудования, так и его потребителям. Производитель может включить в свое изделие, например в маршрутизатор, агентские программы протокола SNMP и гарантировать, что эти программы, даже после принятия новых определений элементов базы MIB, будут соответствовать стандарту. Потребитель может воспользоваться одной клиентской программой для управления несколькими устройствами, в которых версии базы MIB незначительно отличаются. Понятно, что устройство, в котором отсутствуют новые элементы данных MIB, не может предоставить содержащуюся в этих элементах информацию. Однако, поскольку все управляемые устройства используют один и тот же язык для взаимодействия, все они могут выполнить синтаксический анализ запроса и либо предоставить затребованную информацию, либо отослать сообщение об ошибке, указав на отсутствие у них такого элемента.

### **30.5. Примеры переменных базы MIB**

В версиях 1 и 2 протокола SNMP переменные были собраны в одну большую базу MIB. При этом описание полного набора значений переменных находилось в одном большом документе RFC. После публикации стандарта второго поколения MIB-II, разработчики IETF пошли другим путем. Они разбили этот документ на

большое количество отдельных документов MIB, в каждом из которых определялись переменные для определенного типа устройства, и разрешили публиковать новые документы MIB. Таким образом, в процессе стандартизации было создано более 100 отдельных документов MIB, в которых определены более 10000 отдельных переменных. Например, в настоящее время существуют отдельные документы RFC, в которых определены переменные базы MIB, связанные с такими устройствами, как аппаратный мост, источник бесперебойного питания, коммутатор ATM и модем коммутируемой линии передачи. Более того, многие производители определили переменные базы MIB для выпускаемых ими специфических аппаратных или программных продуктах.

Подробное рассмотрение нескольких элементов данных базы MIB, связанных с протоколами семейства TCP/IP, поможет нам прояснить их содержимое. В табл. 30.2 перечислены возможные переменные базы MIB и их категории.

**Таблица 30.2. Переменные базы MIB и их категории**

Переменная базы MIB	Категория	Назначение
sysUpTime	system	Время, прошедшее с момента последней перезагрузки
ifNumber	interfaces	Количество сетевых интерфейсов
ifMtu	interfaces	Значение MTU для определенного интерфейса
ipDefaultTTL	ip	Значение, используемое в поле времени жизни дейтаграмм протокола IP
ipInReceives	ip	Количество полученных дейтаграмм
ipForwDatagrams	ip	Количество отправленных дейтаграмм
ipOutNoRoutes	ip	Количество ошибок при выполнении маршрутизации
ipReasmOKs	ip	Количество собранных дейтаграмм
ipFragOKs	ip	Количество фрагментированных дейтаграмм
ipRoutingTable	ip	Таблица IP-маршрутизации
icmpInEchos	icmp	Количество полученных эхо-запросов протокола ICMP
tcpRtoMin	tcp	Минимальное время повторной передачи, установленное в протоколе TCP
tcpMaxConn	tcp	Максимально разрешенное количество TCP-соединений
tcpInSegs	tcp	Количество полученных сегментов протокола TCP
udpInDatagrams	udp	Количество полученных дейтаграмм протокола UDP

Большинство перечисленных в табл. 30.2 элементов представляет собой числовые значения, т.е. каждое значение может храниться в одной целочисленной переменной. Однако в базе MIB определены и более сложные структуры. Например, переменная базы MIB ipRoutingTable относится ко всей таблице маршрутизации. Для определения содержимого элемента таблицы маршрутизации используются дополнительные переменные базы MIB. Это позволяет программам поддержки протоколов сетевого управления извлекать информацию, содержащуюся в отдельном элементе таблицы, включая префикс сети, маску адреса и поля, содержащие адреса ближайших точек перехода. Разумеется, переменные базы MIB представляют собой только логические определения каждого элемента данных, поскольку внутреннее представление данных в маршрутизаторе может

отличаться от определения базы МIB. При поступлении запроса агентские программы, работающие на маршрутизаторе, преобразуют форматы, принятые для переменных базы МIB и представлением данных, используемым конкретным устройством для хранения информации.

### 30.6. Структура управляющей информации

Помимо стандартов, определяющих переменные базы МIB и их значения, существует отдельный стандарт, в котором определен набор правил, используемых для определения и идентификации переменных базы МIB. Эти правила входят в спецификацию *структуре управляющей информации* (*Structure of Management Information*, или *SMI*). Чтобы не усложнять протоколы сетевого управления, в спецификации SMI налагаются ограничения на типы переменных, которые могут находиться в базе МIB, а также приводятся правила именования переменных и определения их типов. Например, в стандарт SMI входят определения таких термов, как *IpAddress* (определен в виде строки, состоящей из 4 октетов) и *Counter* (определен как целое число, значение которого может находиться в интервале от 0 до  $2^{32}-1$ ). Также в стандарте сказано, что они являются термами, используемыми для определения переменных базы МIB. Более важно то, что определенные в SMI правила описывают, как в базе МIB определяются таблицы значений (например, таблица IP-маршрутизации).

### 30.7. Формальные определения с использованием ASN.1

В стандарте SMI указано, что определения и ссылки на все переменные базы МIB должны быть выполнены с использованием *абстрактной синтаксической записи версии 1* (*Abstract Syntax Notation 1*, или *ASN.1*)<sup>3</sup>, утвержденной организацией ISO. ASN.1 — это формальный язык, который имеет два основных отличия. Во-первых, это форма записи информации, которая используется в документах, предназначенных для прочтения человеком. Во-вторых, это — компактное кодированное представление той же информации в протоколах связи. В обоих случаях точная формальная запись устраняет любую двусмысленность как в плане представления данных, так и их значения. Например, вместо указания того, что переменная содержит целое число, разработчик протоколов, использующий запись ASN.1, должен определить ее точную форму и диапазон допустимых числовых значений. Такая точность особенно важна в том случае, когда реализации протоколов должны работать на разнотипных компьютерах, в которых используются различные представления элементов данных.

Помимо обеспечения однозначной интерпретации документов стандартов, запись ASN.1 также позволяет упростить реализацию протоколов сетевого управления и гарантирует возможность их взаимодействия. В ней точно определено, каким образом следует закодировать имена и элементы данных в сообщении. Следовательно, при представлении документации базы МIB в формате ASN.1, текст, читаемый человеком, можно автоматически преобразовать в закодированный вид, используемый в сообщениях. Обобщенно можно сказать так.

*В протоколах сетевого управления семейства TCP/IP для определения имен и типов переменных, содержащихся в базе управляющей информации (MIB), используется формальная запись под названием ASN.1. Точный синтаксис устраниет неоднозначность при интерпретации названия и содержимого переменных.*

<sup>3</sup> Аббревиатура ASN.1 произносится так: “A-S-N точка 1”.

## 30.8. Структура и представление имен объектов базы MIB

Уже было сказано, что с помощью ASN.1 определяются как сами элементы данных, так и их имена. Однако для того, чтобы разобраться в формате обозначения имен переменных базы MIB, необходимо сначала получить представление о базовом пространстве имен. Имена, используемые для обозначения переменных базы MIB, выбираются из пространства имен *идентификаторов объектов*, поддерживаемого организациями ISO и ITU. В этом пространстве имен можно обозначить все возможные объекты. Пространство имен не ограничивается только переменными, используемыми в процессе сетевого управления. В него также входят имена для произвольных объектов (например, имя имеет каждый документ международного стандарта протоколов).

Пространство имен идентификаторов объекта является *абсолютным (глобальным)*. Это означает, что благодаря своей структуре имена являются уникальными в глобальном масштабе. Подобно большинству крупных пространств имен, по своей сути являющихся абсолютными, пространство имен идентификаторов объектов имеет иерархическую структуру. Ответственность за управление частью пространства имен разделяется на каждом уровне. Это позволяет отдельным группам получать полномочия на присвоение некоторых имен, не согласовывая при этом каждое присвоение с центральным органом управления<sup>4</sup>.

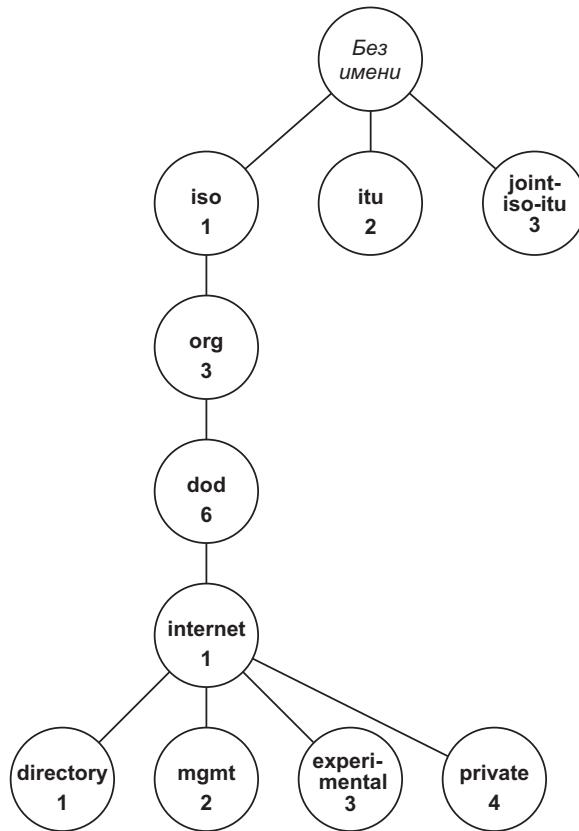
Корень иерархической структуры идентификаторов объектов не имеет имени, но имеет три прямых потомка, управляемых ISO, ITU или совместно ISO и ITU. Потомкам присваиваются как короткие текстовые строки, так и идентифицирующие их целые числа (текстовые строки используются людьми для интерпретации имен объектов, а целые числа используются программным обеспечением компьютера для создания сжатых закодированных представлений имен). Организация ISO выделила одно поддерево для использования другими организациями по национальным или международным стандартам (включая организации по стандартам США), а Национальный институт стандартов и технологий (U.S. National Institute for Standards and Technology<sup>5</sup>, или NIST) выделил отдельное поддерево для Министерства обороны США. И, наконец, совет IAB подал прошение в Министерство обороны на выделение ему поддерева в пространстве имен. На рис. 30.2 изображены составляющие иерархии идентификаторов объектов и показано расположение узла, используемого протоколами сетевого управления семейства TCP/IP.

Имя объекта в иерархии представляет собой последовательность числовых меток в узлах, расположенных от корня по пути следования к объекту. Отдельные компоненты последовательности отделяются друг от друга точками. Например, имя, представленное в виде последовательности чисел 1.3.6.1.2, соответствует узлу, обозначенному на рис. 30.2, как mgmt, и относится к поддереву *управления объединенной сетью* (*Internet management subtree*). Базе MIB назначен узел, относящийся к поддереву mgmt, с меткой mib и числовым значением 1. Поскольку все переменные базы MIB относятся к этому узлу, их имена начинаются с префикса 1.3.6.1.2.1.

Ранее было сказано, что в базе MIB переменные сгруппированы по двум категориям. Точное назначение категорий теперь можно легко объяснить: они представляют собой поддеревья узла mib пространства имен идентификаторов объектов. На рис. 30.3 изображена часть поддерева имен, относящаяся к узлу mib.

<sup>4</sup> Здесь читатели должны вспомнить принцип распределения полномочий на управление иерархическим пространством имен, упомянутый при рассмотрении системы доменных имен в главе 24.

<sup>5</sup> Организация NIST раньше называлась Национальным бюро стандартов (National Bureau of Standards).



*Рис. 30.2. Часть построенного по иерархическому принципу пространства имен идентификаторов объектов, используемая для именования переменных базы MIB. Имя объекта состоит из числовых меток, расположенных от корня по пути следования к объекту*

Синтаксис имен можно объяснить с помощью двух примеров. На рис. 30.3 показано, что категории, обозначенной как ip, присвоено числовое значение 4. Следовательно, имена всех переменных базы MIB, соответствующие протоколу IP, имеют идентификатор, начинающийся с префикса 1.3.6.1.2.1.4. Если бы возникла необходимость написать текстовые метки вместо числовых представлений, имя имело бы следующий вид:

iso.org.dod.internet.mgmt.mib.ip

Переменной базы MIB под названием ipInReceives присвоен числовой идентификатор 3, относящийся в пространстве имен к узлу ip. Следовательно, его имя имеет такой вид:

iso.org.dod.internet.mgmt.mib.ip.ipInReceives

Соответствующее ему числовое представление выглядит так:  
1.3.6.1.2.1.4.3

Когда имена переменных базы MIB используются в сообщениях протоколов сетевого управления, к каждому имени добавляется суффикс. В случае простых переменных суффикс 0 означает, что — это экземпляр переменной с указанным

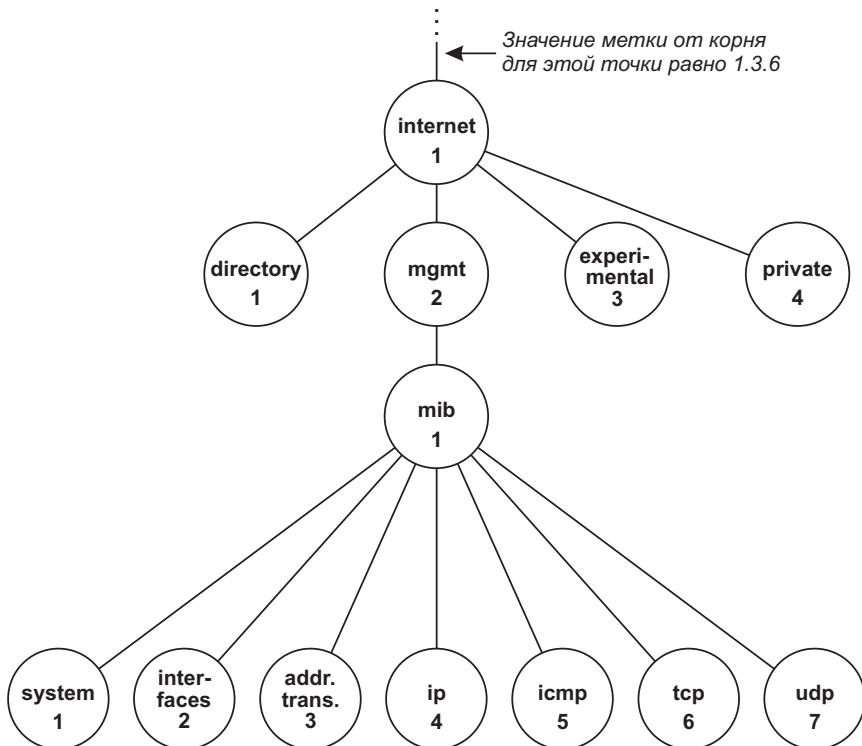


Рис. 30.3. Часть пространства имен идентификаторов объектов, принадлежащая узлу *mib* IAB. Каждое поддерево соответствует одной из категорий переменных базы MIB

именем. Поэтому, если указанная выше переменная будет помещена в сообщение, отосланное некоторому маршрутизатору, ее числовое представление будет иметь следующий вид:

1.3.6.1.2.1.4.3.0

Это означает, что речь идет об экземпляре переменной *ipInReceives*, находящейся на указанном маршрутизаторе. Обратите внимание, что не существует каких-то признаков, по которым можно определить, какое числовое значение или суффикс назначены конкретной переменной. Для того чтобы узнать, какие числовые значения присвоены каждому типу объекта, необходимо обратиться к официальным стандартам. Таким образом, программы, которые обеспечивают преобразование между текстовой формой и базовыми числовыми значениями, должны получать эту информацию в полном объеме, обращаясь к таблицам соответствия. Не существует явно заданного алгоритма вычисления, с помощью которого можно осуществить такое преобразование.

В качестве второго, более сложного примера рассмотрим переменную базы MIB *ipAddrTable*, содержащую список IP-адресов для каждого сетевого интерфейса. Эта переменная существует в пространстве имен в виде поддерева, начинающегося с узла *ip*; ей присвоено числовое значение 20. Следовательно, для обращения к этой переменной используется префикс

`iso.org.dod.internet.mgmt.mib.ip.ipAddrTable`

## Или числовой эквивалент

1.3.6.1.2.1.4.3.20

В терминах языка программирования таблица IP-адресов представляет собой одномерный массив, каждый элемент которого является структурой (записью), содержащей пять элементов: IP-адрес, целочисленный индекс интерфейса, соответствующего данному элементу, маска подсети, широковещательный IP-адрес и целое число, определяющее максимальный размер дейтаграммы, которую может собрать (восстановить) маршрутизатор. Разумеется, маловероятно, чтобы маршрутизатор хранил в своей памяти данную информацию именно в виде описанного выше массива. Маршрутизатор может хранить эту информацию в большом количестве переменных, или ему потребуется извлечь ее из нескольких структур с помощью указателей. Однако в базе MIB предусмотрено имя для такого массива, как будто он существует. Поэтому преобразование информации из внутреннего представления в массив выполняют агенты управления, запущенные на отдельных маршрутизаторах. Этот принцип можно сформулировать так.

*Хотя стандарты базы MIB определяют детали построения структур данных, в них не оговаривается конкретная реализация. Напротив, определения базы MIB по сути являются универсальным виртуальным интерфейсом, с помощью которого администратор может получить доступ к данным. Преобразование информации из внутреннего представления в формат элемента базы MIB выполняется агентами управления.*

Используя форму записи ASN.1, переменную ipAddrTable можно определить следующим образом:

```
ipAddrTable ::= SEQUENCE OF IpAddrEntry
```

В таком представлении строки SEQUENCE и OF являются ключевыми словами, которые определяют, что переменная ipAddrTable — это одномерный массив, состоящий из элементов IpAddrEntry. Согласно определению, каждый элемент в массиве состоит из пяти полей (здесь предполагается, что элемент IpAddress уже определен), как показано ниже.

```
IpAddrEntry ::= SEQUENCE {
    ipAdEntAddr
       IpAddress,
    ipAdEntIfIndex
        INTEGER,
    ipAdEntNetMask
       IpAddress,
    ipAdEntBcastAddr
       IpAddress,
    ipAdEntReasmMaxSize
        INTEGER (0..65535)
}
```

Чтобы присвоить числовые значения элементу IpAddrEntry и каждому элементу последовательности, состоящей из IpAddrEntry, нужно дать еще несколько определений. Например, запись

```
ipAddrEntry { ipAddrTable 1 }
```

определяет, что элемент IpAddrEntry относится к категории IpAddrTable и имеет числовое значение 1. Аналогично, запись

```
ipAdEntNetMask { ipAddrEntry 3 }
```

присваивает элементу ipAdEntNetMask числовое значение 3 (поскольку он третий по счету в определении структуры ipAddrEntry), который принадлежит к категории ipAddrEntry.

Уже было сказано, что структура переменной ipAddrTable напоминает одномерный массив. Однако существует огромная разница между тем, каким образом программисты обращаются к элементам массива в своих программах, и тем, как программы поддержки сетевого управления используют таблицы в базе MIB. С точки зрения программистов массив представляет собой набор элементов, которые имеют индекс, используемый для выбора определенного элемента. Например, для выбора третьего элемента из массива `xuz` программист может написать `xuz[3]`. В синтаксисе ASN.1 для доступа к элементам таблицы не используются целочисленные индексы. Вместо них для выбора элемента в таблице базы MIB к его имени добавляется суффикс. В случае рассмотрения примера таблицы IP-адресов, в стандарте определено, что используемый для выбора элемента этой таблицы суффикс должен состоять из IP-адреса. С точки зрения синтаксиса IP-адрес (в десятичном представлении с разделительными точками) добавляется к концу имени объекта и образует ссылку. Таким образом, чтобы обратиться к полю сетевой маски ipAdEntNetMask в элементе таблицы IP-адресов, соответствующем адресу 128.10.2.3, используется имя

```
iso.org.dod.internet.mgmt.mib.ip.ipAddrTable.ipAddrEntry.ipAdEntNetMask.  
128.10.2.3
```

В числовом представлении ссылка имеет следующий вид:

```
1.3.6.1.2.1.4.20.1.3.128.10.2.3
```

Хотя добавление индекса к концу имени элемента может показаться неудобным, это является мощным средством, позволяющим клиентам выполнять поиск элементов в таблицах, не располагая при этом информацией о количестве элементов в таблице или типе данных, используемых в качестве индекса. В следующем разделе будет показано, как протоколы сетевого управления используют эту возможность для поэлементного анализа таблицы.

## 30.9. Простой протокол сетевого управления (SNMP)

В протоколах сетевого управления определен процесс взаимодействия между клиентской программой сетевого управления, вызываемой администратором, и серверной программой сетевого управления, выполняющейся на узле сети или маршрутизаторе. Помимо определения формы и назначения информационных сообщений, а также представления имен и значений, используемых в этих сообщениях, в протоколах сетевого управления также определены административные взаимосвязи между управляемыми маршрутизаторами. Это означает, что в этих протоколах предусмотрена аутентификация администраторов.

Вы можете предположить, что протоколы сетевого управления должны содержать большое количество команд. Например, в некоторых ранних протоколах поддерживались команды, позволяющие администратору выполнять следующие действия: *перезагружать* систему, *добавлять* или *удалять* маршруты, *отключать* или *подключать* определенный сетевой интерфейс, *удалять* находящиеся в кэше *адресные привязки*. Главный недостаток построения протоколов управления на основе команд заключается в возникающей в результате этого сложности их структуры. Действительно, для каждой операции с элементом данных в протоколе должна быть предусмотрена отдельная команда. Например, команда для удаления элемента из таблицы маршрутизации, отличается от ко-

манды, используемой для отключения интерфейса. Поэтому при введении новых элементов данных в протокол необходимо внести соответствующие изменения.

В протоколе SNMP используется альтернативный подход к проблеме сетевого управления. Вместо определения большого набора команд, в протоколе SNMP все операции построены по *принципу выборки-хранения* (*fetch-store*)<sup>6</sup>. Теоретически в протокол SNMP входят только две команды, позволяющие администратору выполнить выборку значения из элемента данных или сохранить значение в элементе данных. Все другие операции определены как производные этих двух операций. Например, хотя в протоколе SNMP нет явно заданной операции *перезагрузки* операционной системы, эквивалентную ей операцию можно определить путем объявления элемента данных, который задает время до следующей перезагрузки, и позволив администратору присвоить этому элементу значение (в том числе равное нулю).

Главными преимуществами использования принципа выборки-хранения являются устойчивость, простота и гибкость. Устойчивость протокола SNMP очень высока, поскольку его определение остается неизменным даже при добавлении новых элементов в базу MIB, а новые операции определены в нем как следствие сохранения данных в этих элементах. Программу поддержки протокола SNMP просто реализовать, проанализировать и отладить, поскольку в этом протоколе не предусмотрены частные случаи для каждой команды. И наконец, протокол SNMP является чрезвычайно гибким, поскольку в нем можно записать любую команду в компактной форме.

Разумеется, с точки зрения администратора протокол SNMP остается чрезвычайно закрытым. Эта проблема решается за счет создания пользовательского интерфейса к программам поддержки сетевого управления. С его помощью легко представить операции в форме обычных команд, привычных человеку (например, команда *перезагрузить*). Таким образом удается свести практически на нет разницу между использованием администратором протокола SNMP и других протоколов сетевого управления. Производители продают программное обеспечение для сетевого управления, в котором предусмотрен графический пользовательский интерфейс. В таком программном обеспечении отображаются схемы сетевых соединений и используется принцип “навести и щелкнуть” для взаимодействия с пользователем.

Как показано в табл. 30.3, кроме рассмотренных нами двух команд, в протоколе SNMP предусмотрены и другие операции. Например, команда *get-next-request* позволяет выполнять поэлементный перебор таблицы элементов.

**Таблица 30.3. Набор возможных команд протокола SNMP**

Команда	Описание
get-request	Извлечь значение из заданной переменной
get-next-request	Извлечь значение, не зная точного имени его переменной
get-bulk-request	Извлечь большой объем данных (например, целую таблицу)
response	Ответ на любой из перечисленных выше запросов
set-request	Сохранить значение в заданной переменной
inform-request	Ссылка на дополнительные данные (например, для proxy)
snmpv2-trap	Ответ, вызванный событием
report	В настоящее время не определен

<sup>6</sup> Принцип выборки-хранения взят из системы протоколов управления HEMS. Более детально этот принцип описан Парtridgeм (Partridge) и Тревиттом (Trevitt) в [RFC 1021, 1022, 1023 и RFC 1024].

С помощью команд `get-request` и `set-request` обеспечиваются базовые операции выборки и хранения, а с помощью команды `response` — получение ответа. В протоколе SNMP определено, что операции должны быть *неделимыми* (*atomic*). Это означает, что если в SNMP-сообщении указаны операции с несколькими переменными, сервер либо выполняет все операции, либо не выполняет ни одну из них. В частности, операции присваивания не выполняются, если в процессе выполнения одной из них происходит ошибка. С помощью команды `trap` администраторы могут запрограммировать серверы так, чтобы они отсылали информацию, когда происходит какое-либо событие. Например, SNMP-сервер можно запрограммировать так, чтобы в том случае, когда одна из подключенных сетей перестает использоваться (например, в результате отказа одного из интерфейсов), администратору посыпалось соответствующее сообщение.

### 30.9.1. Выполнение поиска по таблицам при использовании имен

Уже было сказано, что в стандарте ASN.1 не предусмотрены механизмы для объявления массивов и их индексирования, как это принято в языках программирования. Однако к отдельным элементам таблицы можно обратиться, добавив суффикс к идентификатору объекта, принадлежащего таблице. Иногда клиентской программе необходимо проанализировать значения элементов таблицы, для которых ей не известны корректные суффиксы. Команда `get-next-request` позволяет клиенту выполнять поэлементный перебор таблицы, не располагая при этом информацией о количестве находящихся в ней элементов. Правила довольно просты. Посылая команду `get-next-request`, клиент указывает префикс действительного идентификатора объекта,  $P$ . Агент анализирует набор идентификаторов объекта для всех контролируемых им переменных и посыпает ответное сообщение для той переменной, которая является следующей в лексикографическом порядке. Это означает, что агенту должны быть известны имена всех переменных, выраженные в форме ASN.1; кроме того, он должен иметь возможность выбрать первую переменную с идентификатором объекта, большим  $P$ . Поскольку в базе MIB для индексации таблицы используются суффиксы, клиент может отослать префикс соответствующего таблице идентификатора объекта и получить первый элемент в этой таблице. Клиент может также посыпать имя первого элемента в таблице и получить второй, и т.д.

Рассмотрим один из примеров такого поиска. Вспомним, что в переменной `ipAddrTable` для идентификации элементов таблицы используются IP-адреса. Клиент, которому не известно, какие именно IP-адреса находятся в таблице маршрутизации данного устройства, не может сформировать полный идентификатор объекта. Однако, для выполнения поиска в таблице клиент может использовать команду `get-next-request`, послав агенту следующий префикс:

```
iso.org.dod.internet.mgmt.mib.ip.ipAddrTable.ipAddrEntry.ipAdEntNetMask
```

В числовом представлении он имеет следующий вид:

```
1.3.6.1.2.1.4.20.1.3
```

При этом сервер возвращает значение поля сетевой маски для первого элемента таблицы `ipAddrTable`. Чтобы запросить значение следующего элемента таблицы, клиент должен использовать полученный от сервера полный идентификатор объекта.

## 30.10. Формат SNMP-сообщений

В отличие от протоколов TCP/IP протокол SNMP не предусматривает в сообщениях полей фиксированного формата — в них используется стандартное кодирование ASN.1. Поэтому такие сообщения человеку трудно декодировать и

анализировать. После рассмотрения определения SNMP-сообщения, представленного в формате ASN.1, мы ознакомимся в общих чертах с методом кодирования ASN.1, а также рассмотрим пример закодированного SNMP-сообщения.

В листинге 30.1 показано, как описать сообщение протокола SNMP с помощью грамматики, соответствующей стилю ASN.1. В общем можно сказать, что каждый элемент грамматики состоит из описательного имени, после которого следует объявление типа элемента. Текст, следующий за двумя тире, является комментарием. Например, такой элемент как

```
msgVersion INTEGER (0..2147483647)
```

объявляет, что имя msgVersion является неотрицательным целым числом, меньшим или равным 2147483647.

### Листинг 30.1. Формат SNMP-сообщения, представленный в форме записи ASN.1

```
SNMPv3Message ::=  
SEQUENCE {  
    msgVersion INTEGER (0..2147483647),  
    -- для SNMPv3 здесь указывается цифра 3  
    msgGlobalData HeaderData,  
    msgSecurityParameters OCTET STRING,  
    msgData ScopedPduData  
}
```

Как показано в листинге 30.1, каждое SNMP-сообщение состоит из четырех основных частей: целого числа, идентифицирующего *версию* протокола, дополнительных данных заголовка, набора параметров защиты и области данных, несущих полезную нагрузку. Для каждого из использованных термов необходимо дать точное определение. Например, в листинге 30.2 показано, как определяется содержимое раздела HeaderData.

### Листинг 30.2. Определение раздела HeaderData SNMP-сообщения

```
HeaderData ::= SEQUENCE {  
    msgID INTEGER (0..2147483647),  
    -- используется для сопоставления запросов и ответов  
    msgMaxSize INTEGER (484..2147483647),  
    -- максимальный размер ответного сообщения, который может принять от-  
    правитель  
    msgFlags OCTET STRING (SIZE(1)),  
    -- отдельные биты флагков определяют свойства сообщения:  
    -- бит 7 используется авторизация  
    -- бит 6 используется режим конфиденциальности  
    -- бит 5 запрашивает ответ  
    msgSecurityModel INTEGER (1..2147483647)  
    -- определяет точный формат следующих далее параметров защиты }
```

Область данных SNMP-сообщения разделена на *модули данных протокола* (*Protocol Data Unit*, или *PDU*). Каждый элемент PDU содержит запрос (посылаемый клиентом) или ответ (посылаемый агентом). В версии протокола SNMPv3 каждый элемент PDU можно послать как в виде обычного текста, так и в зашифрованной форме для конфиденциальности. Соответственно в грамматике определено понятие выбора *CHOICE*. В терминологии языков программирования это понятие называется *размеченным обединением* (*discriminated union*).

```

ScopedPduData ::= CHOICE {
    plaintext ScopedPDU,
    encryptedPDU OCTET STRING
        -- зашифрованное значение ScopedPDU
}

```

Определение зашифрованного элемента PDU начинается с идентификатора машины<sup>7</sup>, с помощью которой он был создан. После идентификатора машины указывается имя контекста и октеты зашифрованного сообщения.

```

ScopedPDU ::= SEQUENCE {
    contextEngineID OCTET STRING,
    contextName OCTET STRING,
    data ANY
        -- т.е. PDU, как будет определено ниже
}

```

Элемент `data` в определении структуры `ScopedPDU` имеет тип `ANY`, поскольку в поле `contextName` определяются его явные детали. В модели обработки сообщений протокола SNMPv3 (SNMPv3 Message Processing Model, или v3MP), определено, что в элементе `data` должен находиться один из элементов PDU протокола SNMP, как показано в листинге 30.3. Синтаксис для каждого типа запроса должен быть определен дополнительно.

### Листинг 30.3. Определение элемента PDU протокола SNMP, представленное в форме ASN.1

```

PDU ::==
CHOICE {
    get-request
        GetRequest-PDU,
    get-next-request
        GetNextRequest-PDU,
    get-bulk-request
        GetBulkRequest-PDU,
    response
        Response-PDU,
    set-request
        SetRequest-PDU,
    inform-request
        InformRequest-PDU,
    snmpV2-trap
        SNMPv2-Trap-PDU,
    report
        Report-PDU,
}

```

В этом определении указано, что каждый элемент SNMP-данных состоит из одного из восьми типов. Чтобы закончить определение SNMP-сообщения, мы должны дополнитель но определить синтаксис восьми отдельных типов. Например, в листинге 30.4 показано определение сообщения `get-request`. Формально оно определяется через элемент `GetRequest-PDU`.

---

<sup>7</sup> В протоколе SNMPv3 существуют отличия между *приложением* (*application*), которое использует службу протокола SNMP, и *машиной* (*engine*), которая представляет собой низкоуровневое программное обеспечение, посылающее запросы и принимающее ответы.

#### Листинг 30.4. Определение сообщения `get-request`, представленное в форме ASN.1

```
GetRequest-PDU ::= [0]
  IMPLICIT SEQUENCE {
    request-id
      Integer32,
    error-status
      INTEGER (0..18),
    error-index
      INTEGER (0..max-bindings),
    variable-bindings
      VarBindList
  }
```

Чтобы ознакомиться с определением остальных термов, обратитесь к спецификации стандарта. Каждый из элементов `error-status` и `error-index` представляет собой состоящее из одного октета целое число, значение которого в запросе равно нулю. При возникновении ошибки в ответное сообщение вместо нуля помещается соответствующий код, идентифицирующий ее причину. И наконец, элемент `VarBindList` содержит список идентификаторов объектов, для которых клиент ищет значения. В определении элемента `VarBindList`, сделанного в термах ASN.1, указано, что он представляет собой последовательность пар, состоящих из имени объекта и значения. В ASN.1 эти пары представлены как последовательность двух элементов. Таким образом, в самом простейшем запросе элемент `VarBindList` представляет собой последовательность двух элементов: имени и нуля.

### 30.11. Пример закодированного SNMP-сообщения

Для представления закодированных элементов в ASN.1 используются поля переменной длины. В общем случае каждое поле начинается с заголовка, который определяет тип объекта и его длину в байтах. Например, каждая последовательность типа `SEQUENCE` начинается с октета, содержащего шестнадцатеричное число 30. Следующий за ним октет определяет количество последующих октетов данной последовательности.

В листинге 30.5 представлен пример SNMP-сообщения, который показывает, как значения кодируются в виде октетов. Представленное в листинге сообщение является командой `get-request`, в которой определен элемент данных `sysDescr` (числовой идентификатор объекта 1.3.6.1.2.1.1.1.0). Поскольку в листинге показан пример реального сообщения, в нем содержится большое количество дополнительной информации. В частности, сообщение содержит раздел `msgSecurityParameters`, который нами еще не рассматривался. В этом выбранном в качестве примера сообщении для защиты параметров используется вариант `UsmSecurityParameters`. Однако сделанные выше определения помогут вам разобраться со структурой остальных частей сообщения. В листинге 30.5 показан пример закодированного сообщения `get-request` для элемента данных `sysDescr`, а также комментарии, поясняющие числовые значения октетов, выраженных в шестнадцатеричном представлении. Связанные между собой октеты сгруппированы в строки и размещаются в сообщении непрерывно.

**Листинг 30.5. Пример закодированного сообщения `get-request` для элемента данных `sysDescr`**

```
30      67      02      01      03
SEQUENCE len=103 INTEGER len=1 vers=3
 30      0D      02      01      2A
SEQUENCE len=13  INTEGER len=1 msgID=42
 02      02      08      00
INTEGER  len=2 maxmsgsize=2048
 04      01      04
string   len=1 msgFlags=0x04 (значение битов: noAuth, noPriv, reportable)
 02      01      03
INTEGER  len=1 used-based security
 04      25      30      23
string   len=37 SEQUENCE len=35 UsmSecurityParameters
 04      0C      00      00      00      63      00      00      00
string   len=12 msgAuthoritativeEngineID ...
  A1      C0      93      8E      23
engine   is at IP address 192.147.142.35, port 161
 02      01      00
INTEGER  len=1 msgAuthoritativeEngineBoots=0
 02      01      00
INTEGER  len=1 msgAuthoritativeEngineTime=0
 04      09      43      6F      6D      65      72      42      6F      6F      6B
string   len=9 -----msgUserName value is "ComerBook"-----
 04      00
string   len=0 msgAuthenticationParameters (none)
 04      00
string   len=0 msgPrivacyParameters (none)
 30      2C
SEQUENCE len=44 ScopedPDU
 04      0C      00      00      00      63      00      00
string   len=12 -----contextEngineID-----
 00      A1      C0      93      8E      23
-----
 04      00
string   len=0 contextName = "" (default)
CONTEXT [0] IMPLICIT SEQUENCE
  A0      1A
getreq. len=26
 02      02      4D      C6
INTEGER  len=2 request-id = 19910
 02      01      00
INTEGER  len=1 error-status = noError(0)
 02      01      00
INTEGER  len=1 error-index=0
 30      0E
SEQUENCE len=14 VarBindList
 30      0C
SEQUENCE len=12 VarBind
 06      08
OBJECT  IDENTIFIER name
 2B      06      01      02      01      01      01      00
 1.3    . 6     . 1     . 2     . 1     . 1     . 1     .
 05      00
null    len=0 (no value specified)
```

Как показано в листинге 30.5, сообщение начинается с кода 30, соответствующего последовательности SEQUENCE, длина которой 103 октета<sup>8</sup>. Первый элемент в последовательности представляет собой целое число размером 1 октет, которое определяет *версию* протокола. Так, значение 3 указывает, что это сообщение протокола SNMPv3. Последующие поля определяют идентификатор сообщения и максимальный размер ответного сообщения, которое может принять отправитель. Закрытая информация, в том числе имя пользователя (ComerBook), следует за заголовком сообщения.

Элемент GetRequest-PDU находится в конце сообщения. Последовательность, обозначенная как ScopedPDU, определяет контекст, в котором следует интерпретировать остальную часть сообщения. Октет A0 определяет код операции get-Request. Поскольку старший бит установлен, интерпретация этого октета *зависит от контекста*. Это означает, что шестнадцатеричное значение A0 определяет элемент GetRequest-PDU только в контексте и не является универсально зарезервированным значением. Вслед за октетом запроса указан октет, определяющий длину запроса — 26 октетов. Идентификатор запроса занимает два октета, но размер каждого из элементов — error-status и error-index — один октет. И наконец, последовательность пар содержит одну привязку — один идентификатор объекта и *нулевое* значение. Как и ожидалось, идентификатор закодирован обычным образом, за исключением того, что две первые числовые метки объединены в один октет.

## 30.12. Новые возможности протокола SNMP версии 3

Уже было сказано, что версия 3 протокола SNMP построена на основе его предыдущих версий. При этом она существенно расширяет и дополняет их структуру и возможности. Основные изменения коснулись областей защиты и управления. Создание третьей версии протокола преследовало две цели. Во-первых, протокол SNMPv3 разработан таким образом, чтобы в нем поддерживались обобщенные и гибкие правила безопасности. Благодаря этому взаимодействие между администратором и управляемыми устройствами можно привести в соответствие с принятыми в организации принципами защиты. Во-вторых, в новой версии упрощена система управления безопасностью.

Для универсальности и гибкости в протокол SNMPv3 включены средства, обеспечивающие несколько аспектов защиты, причем каждое средство можно сконфигурировать независимо от остальных. Например, в версии 3 поддерживаются: *автентификация сообщений* которая гарантирует, что те или иные команды исходят от уполномоченного администратора; *конфиденциальность*, которая гарантирует, что никто не может прочитать сообщения во время их следования по маршруту между компьютером администратора и управляемым устройством; *автентификация и удаленная конфигурация*, которые означают, что уполномоченный администратор может внести изменения в конфигурацию перечисленных выше элементов защиты, физически не находясь рядом с устройством.

## 30.13. Резюме

Протоколы сетевого управления позволяют администраторам осуществлять текущий контроль над маршрутизаторами и узлами сети, а также управлять ими. Клиентская программа сетевого управления, выполняющаяся на рабочей

<sup>8</sup> В SNMP-сообщении элементы последовательности используются очень часто, поскольку в стандарте протокола SNMP вместо обычных конструкций, принятых в языках программирования, таких как array или struct, используется последовательность SEQUENCE.

станции администратора, связывается с одним или несколькими серверами, называемыми агентами, которые запущены на управляемых устройствах. Поскольку объединенная сеть состоит из неоднородных машин и сетей, программы поддержки сетевого управления в семействе протоколов TCP/IP выполняются на уровне приложений и используют транспортные протоколы объединенной сети (например, протокол UDP) для осуществления взаимодействия между клиентами и серверами.

Стандартным протоколом сетевого управления в семействе TCP/IP является протокол SNMP, т.е. простой протокол сетевого управления. Спецификация протокола SNMP определяет протокол управления низшего уровня, в котором предусмотрены две основных операции: извлечение значения переменной и сохранение значения в переменной. В протоколе SNMP другие операции являются следствием изменения значений специальных переменных. В протоколе SNMP определен формат сообщений, которыми обмениваются компьютер администратора и управляемый объект.

Совокупность переменных, которые поддерживаются управляемым объектом, определяется в наборе дополнительных стандартов, прилагаемых к протоколу SNMP. Из набора переменных сформирована база управляющей информации (*MIB*). Переменные базы MIB описываются с помощью стандартного синтаксиса ASN.1. ASN.1 — это формальный язык, предназначенный для точного описания синтаксиса объектов. Он поддерживает как развернутую форму имен и значений объектов, предназначенную для прочтения человеком, так и сжатую закодированную форму, предназначенную для машинной обработки. В ASN.1 используется иерархическое пространство имен, с помощью которого обеспечивается глобальная уникальность всех имен базы MIB. Однако распределение части пространства имен выполняется отдельными административными группами.

## Материал для дальнейшего изучения

Кейс (Case) и др. в [RFC 2570] рассматривают в общих чертах протокол SNMPv3, приводят предпосылки и причины его разработки и обсуждают отличия между различными версиями. В этом документе вы найдете список всех документов RFC, относящихся к протоколу SNMPv3, а также комментарии по поводу того, какие из стандартов версии v2 еще применяются. Отдельные аспекты протокола рассматриваются во многих других документах RFC. Например, Вийнен (Wijnen) и др. в [RFC 2575] представляют модель управления доступом, основанную на наблюдении, а Кейс и др. в [RFC 2572] обсуждают процесс управления сообщениями.

Спецификация для ASN.1 и принципы кодирования описаны в стандартах ISO [60] и [61]. Определение языка, который используется для описания модулей MIB, а также описание типов данных приводится Мак-Клори (McCloghrie) и др. в [RFC 2578, 2579, 2580]. Кейс и др. в [RFC 1907] определяют стандарт базы MIB версии 2.

Ранее предложенный вариант протокола сетевого управления под названием HEMS описан Тревиттом (Trewitt) и Партиджем (Partridge) в [RFC 1021, 1022, 1023 и 1024]. Предшественник протокола SNMP, который называется простым протоколом мониторинга шлюза (*Simple Gateway Monitoring Protocol* или *SGMP*) рассмотрен Дэйвином (Davin), Кейсом, Федором (Fedor) и Шофтталем (Schoeftstall) в [RFC 1028].

## Упражнения

- 30.1. Перехватите пакет протокола SNMP с помощью сетевого анализатора и декодируйте его поля.
- 30.2. Прочтайте стандарт и выясните, как с помощью формы записи ASN.1 кодируются первые два числовых значения из поля идентификатора объекта в один октет. Почему так происходит?
- 30.3. Прочтайте два стандарта и сравните версии протоколов SNMPv2 и SNMPv3. При каких условиях можно воспользоваться возможностями системы безопасности версии 2, а при каких — нет?
- 30.4. Предположим, разработчикам базы MIB необходимо определить переменную, которая соответствует двумерному массиву. Как с помощью записи ASN.1 можно обратиться к такой переменной?
- 30.5. В чем преимущества и недостатки определения глобально уникальных имен, представленных в формате ASN.1, для переменных базы MIB?
- 30.6. Обратитесь к стандартам и найдите для каждого элемента, показанного в листинге 30.5, соответствующее ему определение.
- 30.7. Если у вас есть клиентская программа протокола SNMP, попробуйте использовать ее для чтения значений переменных базы MIB своего локального маршрутизатора. В чем преимущество того, что любой администратор может прочитать переменные на всех маршрутизаторах? В чем недостаток?
- 30.8. Прочтайте спецификацию базы MIB и найдите определение переменной `ipRoutingTable`, соответствующей таблице IP-маршрутизации. Создайте программу, которая будет использовать протокол SNMP для установки связи с несколькими маршрутизаторами, и выясните, вызывают ли какие-либо элементы в таблицах маршрутизации возникновение маршрутных петель. Укажите, какие именно имена в виде ASN.1 должна генерировать такая программа.
- 30.9. Рассмотрите реализацию агента протокола SNMP. Имеет ли смысл размещать переменные базы MIB в памяти маршрутизатора в такой же форме, как они описаны в протоколе SNMP? Обоснуйте свой ответ.
- 30.10. Докажите, что аббревиатура SNMP (простой протокол сетевого управления) является следствием неправильного употребления терминологии, поскольку протокол SNMP не является “простым”.
- 30.11. Прочтайте о защищенном протоколе IPsec, описанном в главе 32, “Безопасность в объединенной сети и брандмауэры (IPsec)”. Если в организации используется протокол IPsec, нужно ли также пользоваться средствами защиты протокола SNMPv3? Обоснуйте свой ответ.
- 30.12. Имеет ли смысл использовать протокол SNMP для управления всеми устройствами сети. Поясните свой ответ. (*Подсказка.* Рассмотрите простое аппаратное устройство, наподобие модема для коммутируемой линии передачи).



# 31

## Обзор структуры протоколов

### 31.1. Введение

На основе протокола TCP/IP было создано большое количество приложений, которые нельзя рассмотреть в одной книге. Вообще говоря, любой программист может создать собственный протокол уровня приложений и использовать для сквозной передачи данных между источником и получателем протокол TCP или UDP. Фактически, при написании распределенных приложений на основе протокола TCP/IP программисты должны определить протокол уровня приложений.

Естественно, не обязательно вникать в детали всех протоколов, важно знать, какие протоколы существуют и как их можно использовать. В этой главе приведен краткий обзор структуры основных протоколов и показано, какие из них могут использоваться в приложениях.

### 31.2. Взаимосвязи между протоколами

На рис. 31.1 показаны взаимосвязи между основными протоколами, которые были рассмотрены в предыдущих главах. Каждый прямоугольник соответствует одному протоколу и расположен над протоколами, которые он использует. Например, почтовый протокол SMTP использует протокол TCP, который, в свою очередь, использует протокол IP. Прикладные программы могут использовать все протоколы, лежащие выше уровня протокола IP.

Кое-что в рис. 31.1 требует пояснения. Нижний уровень представляет протоколы, которые зависят от используемых аппаратных средств. В этот уровень включены все протоколы управления аппаратными средствами (например, доступа к среде передачи данных и логического управления связью). Как уже было сказано, мы предполагаем, что к этому уровню можно отнести любую систему передачи пакетов, при условии что она может использоваться для передачи дейтаграмм по протоколу IP. Таким образом, если система сконфигурирована на пересылку дейтаграмм через туннель, то вход в туннель может рассматриваться как интерфейс аппаратных средств, несмотря на его программную реализацию.

Ко второму уровню снизу отнесены протоколы канального уровня и протоколы преобразования адресов типа SLIP, PPP, ARP и ATMARP. Конечно, такие протоколы требуются для организации далеко не всех сетевых технологий. Например, протокол ARP используется в сетях, поддерживающих на аппаратном уровне режим широковещания и не требующих установки соединения с получателем, как сеть Ethernet. Протокол ATMARP используется в сетях с множественным доступом, не поддерживающих режим широковещания, типа ATM. Протокол RARP используется достаточно редко и только в бездисковых рабочих станциях. Другие протоколы канального уровня или протоколы привязки адре-

сов, могут находиться на этом же уровне в рис. 31.1, но в настоящее время они не так широко распространены.

На третьем уровне снизу находится протокол IP. В него включен также протокол обработки ошибок и протокол управляющих сообщений ICMP, а также дополнительный протокол управления многоадресатными группами IGMP. Обратите внимание, что IP — единственный протокол, который занимает все пространство уровня по ширине (см. рис 31.1). Это говорит о том, что все протоколы низшего уровня доставляют поступающую информацию протоколу IP, а все протоколы высшего уровня должны использовать протокол IP для отправки исходящих дейтаграмм. Уровень протокола IP показан в прямой зависимости от уровня аппаратных средств, поскольку для передачи дейтаграмм по протоколу IP используются драйверы сетевого оборудования или протоколы доступа к передающей среде. (Напомним, что для определения физического адреса устройства по его IP-адресу используется протокол ARP.)

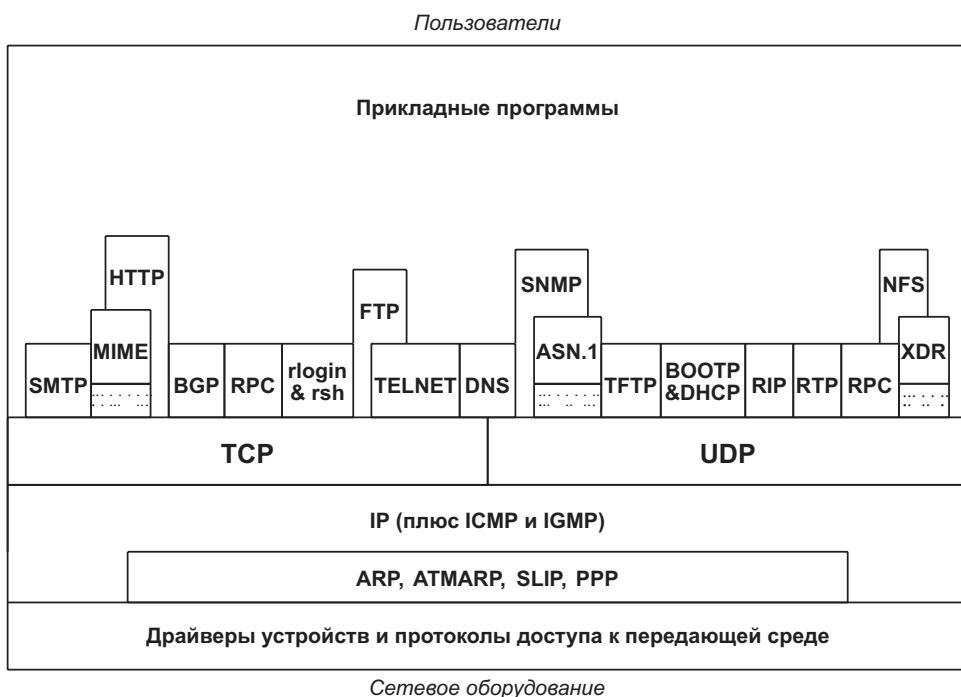


Рис. 31.1. Взаимосвязи между основными высоковневыми протоколами семейства TCP/IP

Протоколы TCP и UDP относятся к следующему (транспортному) уровню. Следует отметить, что, кроме упомянутых здесь двух протоколов, были предложены и другие транспортные протоколы, однако пока что они не нашли широкого применения.

Наиболее сложные взаимосвязи между различными протоколами можно наблюдать на уровне приложений. Напомним, что в протоколе FTP используются команды виртуального терминала, которые определены в протоколе TELNET, для передачи информации по управляющему соединению. В тоже время, для передачи данных по специальному соединению используется протокол TCP. Также напомним, что в протоколе HTTP используется синтаксис заголовков и типы содержимого стандарта MIME. Таким образом, из рис. 31.1 видно, что протокол

FTP зависит от TELNET и от TCP, а HTTP — от MIME и от TCP. В системе доменных имен (DNS) используются для взаимодействия протоколы UDP и TCP, что и показано на рисунке. В системе NFS компании Sun используются стандарты представления внешних данных (XDR) и дистанционного вызова процедур (RPC). Протокол RPC появляется на рис. 31.1 дважды, потому что подобно системе доменных имен в нем может использоваться как протокол UDP, так и TCP.

В протоколе SNMP используется *абстрактная синтаксическая запись версии 1* (*Abstract Syntax Notation*, или ASN.1). И хотя в протоколе SNMP может использоваться как протокол UDP, так и TCP, на рисунке показана только зависимость от протокола UDP, поскольку протокол TCP используется лишь в немногих реализациях протокола SNMP. Поскольку в стандартах XDR, ASN.1 и MIME просто описываются синтаксические соглашения и способы представления данных, в них не используется ни протокол TCP, ни UDP. Таким образом, хотя на рис. 31.1 показано, что SNMP и NFS зависят от UDP, обратите внимание на наличие пунктирной области ниже ASN.1 и XDR, которая означает, что эти стандарты не зависят от протокола UDP. На рассматриваемом нами рисунке было опущено несколько деталей. Например, можно доказать, что протокол IP зависит от протоколов BOOTP/DHCP, или что множество протоколов зависит от DNS, поскольку в программном обеспечении, реализующем такие протоколы, требуется выполнять привязку имен.

### 31.3. Модель песочных часов

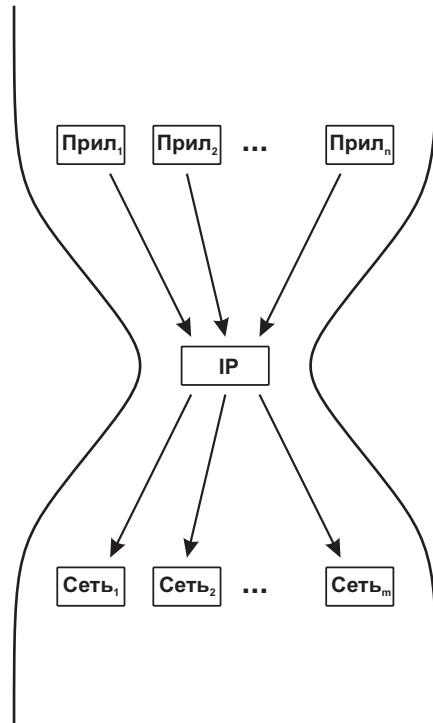
Разработчики представили протоколы глобальной сети Internet в виде *модели песочных часов*. Поскольку протокол IP находится в центре всего процесса взаимодействия, он формирует центр песочных часов. Из всех рассмотренных протоколов, IP — единственный протокол, общий для всех приложений. В конечном счете IP-дейтаграммы передаются по всем соединениям в объединенной сети. Таким образом, способность к универсальному взаимодействию достигается за счет поддержки протокола IP во всех возможных сетевых технологиях. Эта концепция проиллюстрирована на рис. 31.2, где показаны зависимости между протоколом IP, приложениями и базовыми сетевыми технологиями.

### 31.4. Доступ со стороны прикладных программ

В большинстве систем ограничен доступ со стороны прикладных программы к протоколам низкого уровня. Другими словами, в большинстве систем разрешен доступ со стороны прикладных программам только к протоколам TCP или UDP, а также к любым другим протоколам высокого уровня, реализованным на их основе (например, FTP). На самом деле операционная система может ограничить доступ к транспортным протоколам, разрешив открывать порты протокола TCP или UDP с малыми номерами только для привилегированных программ.

Хотя прямой доступ к протоколу IP со стороны приложений — явление достаточно редкое, в некоторых системах для обеспечения этой возможности предусмотрены специальные механизмы. Например, механизм, называемый *пакетным фильтром*, позволяет привилегированным программам управлять процессом демультиплексирования фреймов. С помощью специальных примитивов пакетного фильтра прикладная программа может установить критерии для перехвата пакетов. Например, программа может указать, что следует перехватывать все пакеты, содержащие заданное значение в поле *типа* фрейма. Получив команду на фильтрацию, операционная система будет помещать все пакеты, соответствующие указанному типу, в очередь. Прикладная программа использует механизм фильтрации пакетов для извлечения пакета из очереди. Для та-

ких систем схема, показанная на рис. 31.1, должна быть несколько расширена: на ней нужно показать возможность доступа приложения к нижним уровням протоколов.



*Рис. 31.2. Модель песочных часов. Протокол IP находится в центре, потому что все приложения зависят от него, а сам протокол IP поддерживается во всех сетевых технологиях*

### 31.5. Резюме

Большинство развитых функциональных возможностей семейства протоколов TCP/IP стали возможны благодаря наличию высоковысоконвейных служб и использующих их прикладных программ. Протоколы высокого уровня, которые используются в этих программах, построены на основе двух транспортных служб: ненадежной службы доставки дейтаграмм (UDP) и надежной потоковой транспортной службе (TCP). Обычно в основу работы приложений положена модель взаимодействия типа клиент/сервер. Для взаимодействия с клиентом серверы используют стандартные номера портов протокола, поэтому клиенты всегда “знают”, как соединиться с ними.

С помощью протоколов высокого уровня реализованы пользовательские службы типа World Wide Web, удаленного входа в систему, передачи почты и файлов. Преимущества создания таких служб на основе объединенной сети состоят в том, что появляется универсальная возможность взаимодействия и упрощаются протоколы уровня приложений. В частности, когда такие службы используются двумя машинами, подключенными к объединенной сети, применение транспортных протоколов, устанавливающих непосредственные соединения между получателями, позволяет гарантировать, что программа клиента, запущенная на одной машине, может непосредственно связаться с сервером, запущенным на другой машине. Поскольку при реализации прикладных служб типа

электронной почты используются транспортные протоколы, устанавливающие непосредственное соединение между получателями, работа этих служб не зависит от функционирования промежуточных систем, выполняющих обработку данных (например, транспортировку целого сообщения).

В этой главе было показано, что существует большое разнообразие протоколов уровня приложений и что между ними существуют весьма сложные взаимосвязи. Хотя в объединенной сети может быть запущено большое количество приложений, поддерживающих разные протоколы уровня приложений, львиная доля пакетов, проходящих по сети, относится к одной основной службе — World Wide Web.

## Материал для дальнейшего изучения

Одна из проблем, возникающих при разбиении семейства протоколов на уровни, — найти оптимальное сочетание функциональных возможностей протокола на каждом из уровней. В работе Эджа (Edge) [46] сравниваются протоколы сквозной передачи (end-to-end) данных с методом последовательной передачи от узла к узлу. В статье [115] Зальцер (Saltzer), Рид (Reed) и Кларк (Clark) доказали важность сквозного контроля при передаче данных. В серии документов [RFC 956, 957 и 958] Миллс (Mills) описывает протоколы уровня приложений для синхронизации часов и приводит отчеты о проведенных экспериментах.

## Упражнения

- 31.1. Программы поддержки некоторых протоколов уровня приложений можно написать так, чтобы они использовали другие протоколы уровня приложений, путем простой конвертации запросов. Например, можно создать программу, которая бы принимала FTP-запрос, конвертировала его в TFTP-запрос, отправляла результат серверу TFTP, получала от него файл и передавала бы его по протоколу FTP первоначальному отправителю. Каковы преимущества и недостатки такого подхода к проектированию приложений?
- 31.2. Рассмотрите процесс конвертирования запросов, описанный в предыдущем упражнении. Для каких пар протоколов, изображенных на рис. 31.1, можно выполнить такую конвертацию?
- 31.3. Для некоторых прикладных программ, вызываемых пользователями, может понадобиться прямой доступ к протоколу IP без использования протоколов TCP или UDP. Приведите примеры таких программ. (*Подсказка.* Рассмотрите протокол ICMP.)
- 31.4. Где, по вашему мнению, нужно разместить на рис. 31.1 протоколы поддержки многоадресатной передачи?
- 31.5. В протоколе DNS могут использоваться оба протокола: и TCP, и UDP. Выясните, позволяет ли операционная система вашего компьютера одному процессу принимать DNS-запросы, поступающие и через TCP-соединение, и через протокол UDP.
- 31.6. Рассмотрите сложное приложение, например систему *X window*, и выясните, какие протоколы в нем используются.
- 31.7. Где на рис. 31.1 должен размещаться протокол OSPF?

- 31.8.** Из рис. 31.1 можно сделать вывод, что протокол FTP зависит от протокола TELNET. Вызывает ли ваш локальный клиент FTP программу TELNET или клиент FTP содержит встроенную реализацию протокола TELNET?
- 31.9.** Создайте схему протоколов, аналогичную представленной на рис. 31.1, для Web-браузера. Какие протоколы в нем используются?

# 32

## Безопасность в объединенной сети и брандмауэры (IPsec)

### 32.1. Введение

Как известно для надежного хранения материального имущества нужны крепкие двери и сложные замки. Точно так же, чтобы предотвратить нежелательную утечку информации, необходимо предпринять ряд мер в работе компьютеров и сетей передачи данных. Проблема безопасности информации в объединенной сети является важной и в тоже время трудно разрешимой. Важность проблемы заключается в том, что информация на сегодняшний день имеет существенную ценность — она может быть куплена или продана как непосредственно, так и косвенно (т.е. использоваться для создания новых продуктов и услуг, приносящих высокие прибыли). Обеспечить безопасность в объединенной очень трудно, поскольку для этого требуется не только понимание технических тонкостей работы сетевых аппаратных средств и протоколов, но и знание того, как и когда участвующие в процессе сетевого обмена пользователи, компьютеры, службы и сети могут доверять друг другу. Средства безопасности должны соблюдаться на каждом компьютере и в каждом протоколе. Одно слабое звено может поставить под угрозу безопасность всей сети. И, что более важно, поскольку семейство протоколов TCP/IP может использоваться широким кругом разных пользователей, а также при создании разнообразных служб и сетей, и, поскольку объединенная сеть может пересекать много как политических, так и организационных границ, то задействованные индивидуумы и организации могут не договориться между собой об уровне доверия или правилах, касающихся обработки данных.

В этой главе рассматриваются два главных принципа, положенных в основу безопасности объединенной сети: внешняя безопасность и шифрование. Правила внешней безопасности позволяют организации определить службы и сети, которые могут быть доступны извне, а также допустимые пределы, в которых посторонние лица могут пользоваться ее ресурсами. Шифрование покрывает большинство других аспектов безопасности. Главу мы начнем с рассмотрения основных концепций и терминологии.

### 32.2. Безопасность ресурсов

Термины *безопасность сети* (*network security*) и *информационная безопасность* (*information security*) в широком смысле относятся к секретности, т.е. гарантии того, что информация и службы, имеющиеся в сети, не будут доступны для несанкционированного использования. Безопасность подразумевает механизм защиты,

гарантирующий целостность данных, невозможность несанкционированного доступа к вычислительным ресурсам, шпионажа или перехвата сообщений, а также вторжения в работу служб. Конечно, как нельзя гарантировать абсолютную защищенность материальных ценностей от посягательств, так нельзя гарантировать и полную безопасность сети. Руководство организации должно уделять внимание безопасности своих сетей, как это делается в отношении зданий и офисов: даже несложные меры безопасности могут воспрепятствовать преступлению, значительно усложнив его совершение.

Обеспечение информационной безопасности требует охраны как физических, так и виртуальных ресурсов. К физическим ресурсам можно отнести как пассивные устройства для хранения информации, такие как жесткие диски и компакт-диски, так и активные устройства, такие как компьютеры пользователей. В сетевом окружении понятие физической безопасности относится к кабелям, мостам и маршрутизаторам, а также ко всем устройствам, составляющим инфраструктуру сети. Действительно, хотя о физической безопасности редко упоминается, она часто играет важную роль при планировании полной безопасности. Очевидно, что меры физической безопасности могут предотвратить перехват сообщений путем несанкционированного подключения к сети. Эффективные меры физической безопасности способствуют также предотвращению диверсии (например, умышленному выводу из строя маршрутизатора с целью изменения маршрута следования пакетов и их передачи по альтернативному, незащищенному пути).

Обеспечить безопасность виртуального ресурса, такого как информация, обычно более трудно, чем обеспечить физическую безопасность, поскольку информация не имеет физических границ. Информационная безопасность охватывает много аспектов защиты, основные из которых перечислены ниже.

- *Целостность данных.* Безопасная система должна защищать информацию от несанкционированного изменения или повреждения.
- *Доступность данных.* Система должна гарантировать, что несанкционированный пользователь не сможет помешать законному доступу к данным (например, никто из посторонних не должен иметь возможность блокировать клиентов фирмы от доступа к ее Web-узлу).
- *Секретность, или конфиденциальность.* Система не должна позволять несанкционированным пользователям создавать копии данных во время их передачи по сети, а также анализировать их содержимое в том случае, если копии все же сделаны.
- *Авторизация.* Хотя при осуществлении мер физической безопасности часто выполняют классификацию людей и ресурсов по широким категориям (например, лицам не являющимся служащими компании, запрещается доступ в некоторые помещения), меры информационной безопасности должны быть более избирательны (например, часть полей записи базы данных служащих должна быть доступна только для сотрудников отдела кадров, другая — только для начальника, а третья — для бухгалтерии).
- *Аутентификация.* Система должна позволять двум взаимодействующим между собой объектам проверять подлинность друг друга.
- *Запрещение повторного использования.* Чтобы посторонние не могли перехватывать копии пакетов с целью их дальнейшего использования, система не должна обрабатывать копии повторно переданных пакетов.

### **32.3. Информационная стратегия**

Прежде чем приступить к реализации системы безопасности сети, следует оценить риски и разработать четкую стратегию доступа к информации и ее защиты. В стратегии определяется круг лиц и уровень их доступа к той или иной информации, правила, которых эти лица должны придерживаться в целях неразглашения информации, а также ответственность за нарушение этих правил.

Разработка информационной стратегии во многом зависит от человеческого фактора, потому что

*люди представляют собой наиболее уязвимое звено в любой системе безопасности. Служащий компании либо по злому умыслу, либо по неосторожности, либо не зная принятой в компании стратегии, может поставить под угрозу самую совершенную систему безопасности.*

### **32.4. Безопасность глобальной сети Internet**

Обеспечить безопасность в глобальной сети Internet особенно трудно, поскольку дейтаграммы, передающиеся от отправителя до конечного получателя, проходят через несколько промежуточных сетей и маршрутизаторов, которые не принадлежат ни отправителю, ни конечному получателю и не контролируются ими. Таким образом, поскольку дейтаграммы могут быть перехвачены и изменены без ведома отправителя, их содержимому нельзя доверять. В качестве примера рассмотрим сервер, который использует процедуру *аутентификации источника* (*source authentication*) для проверки того, что запросы поступают от авторизованных клиентов. Процедура аутентификации источника требует, чтобы сервер при получении каждой дейтаграммы проверял IP-адрес отправителя, и принимал запросы только от компьютеров, адреса которых перечислены в специальном списке. Данный вид аутентификации обеспечивает *слабую защиту*, поскольку ее можно легко обойти. В частности, один из промежуточных маршрутизаторов может контролировать трафик, идущий к нашему серверу и от него, и фиксировать IP-адреса авторизованных клиентов. После этого промежуточный маршрутизатор может воссоздать запрос и указать в нем один из зафиксированных адресов отправителя и перехватить ответ сервера. Суть сказанного можно подытожить так.

*Механизм авторизации, в котором для подтверждения подлинности отправителя используется его IP-адрес, не удовлетворяет требованиям безопасности объединенной сети. Любой самозванец, получивший контроль над промежуточным маршрутизатором, может легко обойти систему защиты и выступить в роли авторизованного клиента.*

Для более строгой аутентификации требуется механизм *шифрования*. Отправитель шифрует сообщение с помощью математической функции, которая изменяет значение битов согласно *ключу*, известному только отправителю. Для расшифровки сообщения получатель должен использовать другую математическую функцию и ключ. Тщательный выбор алгоритма шифрования и ключа могут сделать расшифровку сообщения и его подделку на промежуточных машинах практически невозможными.

### **32.5. Безопасность протокола IP (IPsec)**

Группа IETF разработала набор протоколов, которые обеспечивают безопасную связь в глобальной сети Internet. Все вместе они называются семейством протоколов *IPsec* (сокращение от *IP security*, или *защищенный протокол IP*).

В этих протоколах аутентификация и шифрование данных выполняются на уровне протокола IP. Причем они могут использоваться и с IPv4, и с IPv6<sup>1</sup>. Важно то, что вместо полного определения функциональных возможностей или алгоритма шифрования, который нужно использовать, группа IETF сделала систему одновременно и гибкой, и расширяемой. Например, при создании приложений, использующих протоколы IPsec, у программиста есть возможность выбора: применить средство аутентификации, проверяющее достоверность отправителя, или же средство шифрования, которое помимо всего прочего гарантирует конфиденциальность передаваемой информации. Причем возможность выбора является “асимметричной” (например, аутентификация может выполняться в одном направлении и не выполняться в другом). Кроме того, в стандарте протоколов IPsec явно не оговорен тип шифрования или алгоритмом аутентификации. Вместо этого, в стандарте определена общая структура системы, которая позволяет каждой паре взаимодействующих конечных пунктов выбирать алгоритмы и параметры шифрования (например, размер ключа). Чтобы гарантировать способность к взаимодействию, в семейство протоколов IPsec все-таки включен набор алгоритмов шифрования, которые должны поддерживаться во всех реализациях этих протоколов. Суть сказанного выше можно подытожить так.

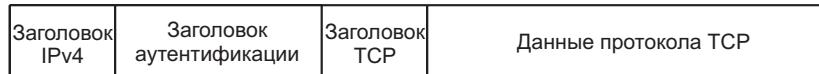
*IPsec — это не один протокол, а набор алгоритмов, обеспечивающих безопасную передачу данных и общую структуру, которая позволяет паре взаимодействующих объектов использовать любые алгоритмы шифрования, гарантирующие соответствующую безопасность соединения.*

## 32.6. Заголовок аутентификации семейства протоколов IPsec

Разработчики семейства протоколов IPsec не стали вносить изменения в формат основного заголовка IP-дейтаграммы или создавать дополнительное поле ее параметров. В протоколах IPsec используется отдельный заголовок *аутентификации* (*Authentication Header*, или *AH*), несущий информацию об аутентификации. На рис. 32.1 показан пример наиболее простого использования заголовка аутентификации с протоколом IPv4. Новый заголовок помещен непосредственно после IP-заголовка.



(а)



(б)

*Рис. 32.1. Структура обычной дейтаграммы IPv4 (а); структура той же дейтаграммы после добавления заголовка аутентификации IPsec (б)*

<sup>1</sup> В этой главе приводятся примеры только для протокола IPv4. Подробное описание протокола IPv6 и примеры заголовков IPsec в дейтаграммах IPv6, приведены в главе 33, “Будущее протокола TCP/IP (IPv6)”.

Как показано на рис. 32.1, заголовок аутентификации протокола IPsec помещен непосредственно после заголовка IP-дейтаграммы, но перед заголовком протокола транспортного уровня. Кроме того, значение поля *типа протокола* в заголовке IP-дейтаграммы изменено на величину 51, что указывает на присутствие заголовка аутентификации.

Возникает вопрос: как при наличии заголовка аутентификации протокола IPsec получатель может определить тип данных, передаваемых в дейтаграмме? Ведь значение поля типа протокола IP-дейтаграммы изменено и всегда равно одному и тому же значению — 51. Для решения этой проблемы в заголовке аутентификации предусмотрено поле *типа следующего заголовка*, в котором и указывается тип передаваемых в дейтаграмме данных. Программы протокола IPsec переписывают оригинальное значение поля типа протокола IP-дейтаграммы в поле типа следующего заголовка. При получении дейтаграммы анализируется информация системы безопасности из заголовка аутентификации для проверки отправителя, а затем используется значение поля *типа следующего заголовка* для дальнейшего демультиплексирования дейтаграммы. Формат заголовка аутентификации показан на рис. 32.2.

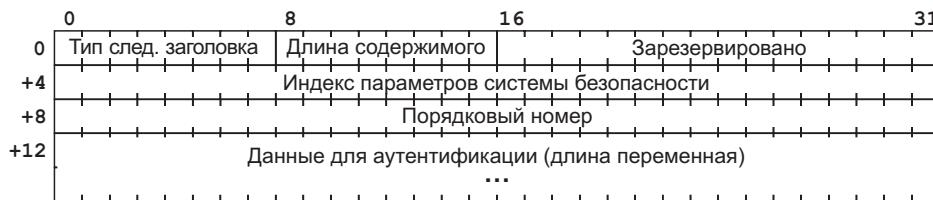


Рис. 32.2. Формат заголовка аутентификации протокола IPsec. В поле *типа следующего заголовка* хранится оригинальное значение поля *типа протокола IP-дейтаграммы*

Интересно, что в поле *длины содержимого* не указывается размер области данных дейтаграммы. Вместо этого в него помещается длина заголовка аутентификации. В остальных полях находится информация, использующаяся для гарантированной безопасной передачи данных. В поле *порядкового номера* содержится уникальный порядковый номер каждого посланного пакета. Этот номер начинается с нуля при выборе определенного алгоритма безопасности и последовательно увеличивается. Значение поля *индекса параметров системы безопасности* определяет используемую схему безопасности, а поле *данные для аутентификации* содержит данные для выбранной схемы безопасности.

### 32.7. Ассоциация обеспечения безопасности

Чтобы понять причину использования индекса параметров системы безопасности, надо отметить, что в принятой схеме безопасности может быть задействована масса деталей, благодаря которым обеспечивается большое разнообразие параметров. Например, принятая схема безопасности включает алгоритм аутентификации, ключ (или ключи), используемые в алгоритме, время жизни, в течение которого ключ остается действительным, время жизни, в течение которого получатель может использовать данный алгоритм, список адресов отправителей, которые уполномочены использовать данную схему. Естественно, вся эта информация не может поместиться в заголовке аутентификации.

Чтобы сэкономить место в заголовке, в протоколе IPsec было введено абстрактное понятие, называемое *ассоциацией обеспечения безопасности* (*Security Association*)

*Association*, или *SA*), и оговорено, что каждый получатель может получить все сведения относительно используемой схемы безопасности из этой ассоциации.

Каждой ассоциации присваивается номер, называемый *индексом параметров системы безопасности*, по которому она идентифицируется. Прежде чем отправитель сможет использовать протокол IPsec для связи с получателем, он должен узнать значение индекса для требуемой ассоциации. Затем отправитель помещает это значение в поле индекса параметров системы безопасности каждой исходящей дейтаграммы.

Индексы не являются глобальными. Получатель создает необходимое количество ассоциаций системы безопасности и каждой из них присваивает значение индекса. Получатель может определить время жизни для каждой ассоциации и по его истечении использовать значение индекса для другой ассоциации. Следовательно, индекс системы безопасности нельзя корректно интерпретировать без запроса к получателю (например, индекс 1 может иметь совершенно различные значения для двух получателей). Сказанное выше можно подытожить так.

*Получатель использует индекс параметров системы безопасности, чтобы идентифицировать ассоциацию безопасности для пакета. Индексы не являются глобальными. Для идентификации ассоциации необходима комбинация адреса получателя и индекса параметров системы безопасности.*

### 32.8. Инкапсуляция зашифрованных данных в протоколе IPsec

Чтобы можно было обрабатывать зашифрованные данные по аналогии с данными для аутентификации, в протоколе IPsec используется механизм *инкапсуляции зашифрованных данных* (*Encapsulating Security Payload*, или *ESP*), который намного сложнее механизма аутентификации. Если значение поля типа протокола IP-дейтаграммы равно 50, это означает, что в дейтаграмме находятся зашифрованные данные. На рис. 32.3 схематично изображена такая дейтаграмма.

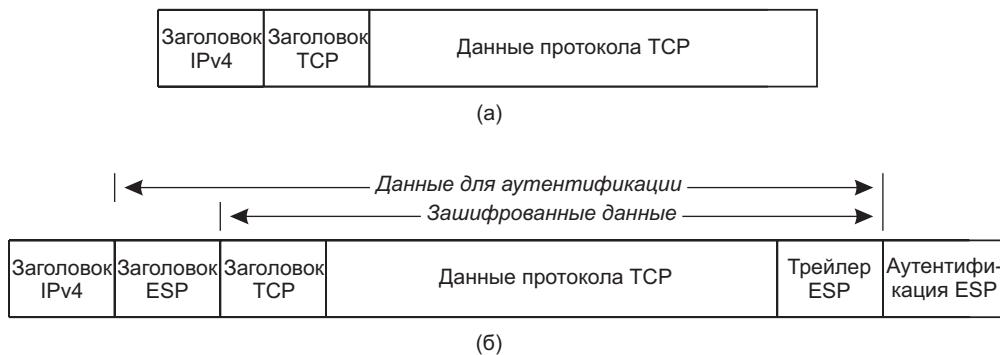


Рис. 32.3. Схематичное изображение дейтаграммы (а), и та же дейтаграмма, в которую инкапсулированы зашифрованные данные протокола IPsec. Использование шифрования означает, что идентифицировать значения полей достаточно сложно (б).

Как показано на рис. 32.3, при использовании зашифрованных данных к дейтаграмме добавляются три дополнительные области. Заголовок ESP следует сразу за заголовком IP-дейтаграммы и предшествует зашифрованной области данных. Трейлер ESP зашифрован вместе с передаваемыми данными. Поле аутентификации ESP имеет переменную длину и следует сразу за зашифрованными данными.

При инкапсуляции зашифрованных данных используются практически те же элементы, которые мы рассматривали в заголовке аутентификации, только порядок их следования другой. Например, заголовок ESP состоит из 8 октетов. В нем указывается индекс параметров системы безопасности и порядковый номер пакета, как показано на рис. 32.4.

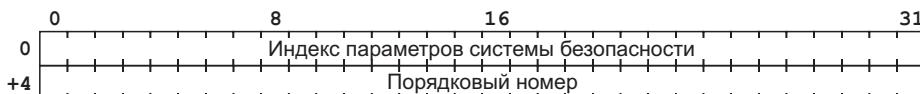


Рис. 32.4. Формат заголовка ESP

Трейлер ESP состоит из необязательного поля переменной длины, использующегося для выравнивания общей длины пакета, поля, в котором указана длина поля выравнивания, и поля *типа следующего заголовка*, за которым следует поле переменной длины, содержащее данные для аутентификации ESP (рис. 32.5).

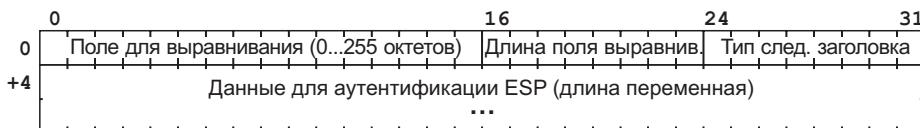


Рис. 32.5. Формат трейлера ESP

Поле для выравнивания не является обязательным. Тем не менее оно может принудительно вводиться в пакет. Для этого имеются три причины. Во-первых, для работы некоторых алгоритмов дешифрования требуется, чтобы после зашифрованного сообщения следовал ряд нулевых октетов. Во-вторых, обратите внимание, что поле типа следующего заголовка выровнено по правой границе 4-октетного поля. Это выравнивание очень важно, поскольку в спецификации IPsec требуется, чтобы данные для аутентификации, которые следуют за трейлером, были выровнены на границу 4-октетного поля. Таким образом, поле для выравнивания может использоваться по своему прямому назначению. В-третьих, в некоторых системах к каждой дейтаграмме может добавляться случайное количество октетов выравнивания, чтобы в случае ее перехвата на одной из промежуточных станций злоумышленники не смогли бы по размеру дейтаграммы догадаться о ее назначении.

## 32.9. Аутентификация и изменяемые поля заголовка

По идеи механизм аутентификации протокола IPsec создавался для того, чтобы можно было гарантировать, что прибывающая дейтаграмма полностью идентична дейтаграмме, посланной отправителем. Однако на практике обеспечить это невозможно. Чтобы понять, почему, напомним, что взаимодействие по протоколу IP выполняется по принципу “от машины к машине”. Это означает, что при передаче дейтаграммы от одного маршрутизатора к другому она будет проходить по иерархии протоколов каждого устройства. В частности, каждый промежуточный маршрутизатор уменьшает на единицу значение поля времени жизни IP-дейтаграммы и пересчитывает контрольную сумму ее заголовка.

В спецификации IPsec поля заголовка IP-дейтаграммы, которые должны изменяться при ее передаче, называются *изменяемыми*. Чтобы такие поля не влияли на процесс аутентификации и не приводили к ошибкам вычислений, в спецификации IPsec говорится, что они должны быть исключены из аутентификации. Таким образом, получив дейтаграмму, программы поддержки протокола IPsec аутентифицирует только неизменяемые поля (например, адрес отправителя и тип протокола).

### 32.10. Туннелирование по закрытому каналу на основе протоколов IPsec

В главе 20, “Взаимодействие частных сетей (NAT, VPN)”, уже говорилось о том, что для обеспечения секретности при передаче данных между двумя сетевыми центрами применяется технология VPN, в которой наряду с туннелированием IP-в-IP используется шифрование. Стандарты IPsec специально разрабатывались для создания зашифрованного туннеля. В частности, в них определены специальные “туннельные” версии и для заголовка аутентификации, и для заголовка инкапсуляции зашифрованных данных (ESP). Структура дейтаграмм в режиме туннелирования показана на рис. 32.6.

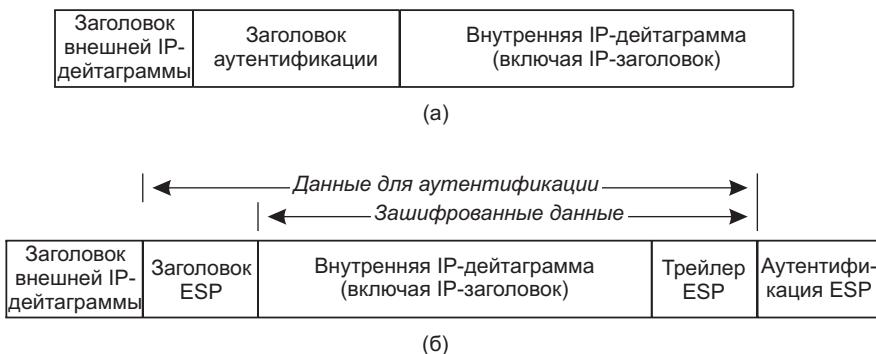


Рис. 32.6. Структура дейтаграмм, использующихся в режиме туннелирования по закрытому каналу на основе протокола IPsec, содержащих данные для аутентификации (а) и зашифрованные данные (б). Содержимое внутренней дейтаграммы полностью защищено

### 32.11. Обязательные алгоритмы шифрования

В спецификации IPsec определен минимальный набор алгоритмов, поддержка которых обязательна (т.е. ими должны быть снабжены все программные реализации). Особенности реализации определены в стандарте для каждого конкретного случая. Список обязательных алгоритмов приведен в табл. 32.1.

Таблица 32.1. Алгоритмы шифрования, которые являются обязательными для реализации в IPsec

Аутентификация	
HMAC с MD5	RFC 2403
HMAC с SHA-1	RFC 2404
Инкапсуляция зашифрованных данных	
DES в режиме цепочки связанных блоков (CBC)	RFC 2405
HMAC с MD5	RFC 2403
HMAC с SHA-1	RFC 2404
Отсутствие аутентификации	
Отсутствие шифрования	

## 32.12. Механизм защищенных сокетов

К середине 1990-ых, когда стало очевидно, что для проведения торговых операций в глобальной сети Internet нужно обеспечить безопасность передаваемых данных, несколькими группами были предложены механизмы безопасности для использования в Web. Одно из предложений стало стандартом де-факто, хотя оно и не было формально принято группой IETF.

Технология под названием *протокол защищенных сокетов* (*Secure Sockets Layer*, или *SSL*) была первоначально создана компанией Netscape. Как следует из названия протокола SSL, он находится на том же уровне, что и API сокетов. Когда клиент использует SSL для связи с сервером, средства протокола SSL позволяют сторонам аутентифицировать друг друга. После этого обе стороны согласовывают алгоритм шифрования, который они будут использовать. И наконец, протокол SSL позволяет двум сторонам установить зашифрованный канал связи (т.е. канал, при передаче данных по которому они шифруются с использованием выбранного алгоритма при заданном уровне секретности).

## 32.13. Брандмауэры и доступ к сети Internet

Механизмы управления доступом к объединенной сети предназначены для защиты определенной сети или всех сетей организации от нежелательного подключения несанкционированных пользователей. Эти механизмы не позволяют посторонним получать или изменять внутренние данные организации, а также каким-либо образом препятствовать взаимодействию пользователей по корпоративной локальной сети. Для создания успешного механизма управления доступом нужно тщательно продумать комплекс мер: ограничения на топологию сети, промежуточное хранение информации и фильтры пакетов.

Основным компонентом для управления доступом к объединенной сети является специализированное устройство, называемое *брандмауэром* (*firewall*)<sup>2</sup>. Обычно брандмауэр устанавливается между внутренней сетью организации и каналом, ведущим к внешним сетям (например, к глобальной сети Internet). Брандмауэр разделяет объединенную сеть на две области, которые неофициально называются *внутренней* и *внешней*.

## 32.14. Множественные соединения и самые слабые связи

Хотя сама идея брандмауэра проста, ее реализация усложняется множеством факторов. Один из них — внутренняя сеть организации может иметь несколько внешних соединений. При этом руководство организации должно сформировать *периметр безопасности* (*security perimeter*), установив брандмауэр на каждое внешнее соединение. Чтобы гарантировать эффективность периметра безопасности, во всех брандмауэрах должны использоваться одинаковые ограничения доступа. В противном случае злоумышленники могут обойти ограничения, наложенные одним брандмауэром, и зайти в объединенную сеть организации через другой брандмауэр<sup>3</sup>. Все сказанное выше можно подытожить так.

---

<sup>2</sup> Термин *брандмауэр* позаимствован из строительства, где он обозначает толстую несгораемую стену, благодаря которой секция строения становится непроницаемой для огня.

<sup>3</sup> Хорошо известно, что уровень безопасности системы определяется ее самым слабым звеном. Этот принцип был назван *аксиомой самой слабой связи* по аналогии с поговоркой, что крепость цепи определяется ее самым слабым звеном.

*Если объединенная сеть организации имеет несколько внешних соединений, то на каждое из них необходимо установить брандмауэр и координировать их работу. Если на всех брандмаузерах будут использоваться разные ограничения доступа, то безопасность сети организации может оказаться под угрозой.*

### 32.15. Реализация брандмауэра

В каком же виде должен быть реализован брандмауэр? Теоретически брандмауэр должен попросту блокировать все несанкционированные соединения, осуществляемые между компьютерами организации и компьютерами, находящимися за ее пределами. На практике же, детали реализации зависят от применяемых сетевых технологий, пропускной способности внешнего канала связи, степени его загрузки и принятых в организации правил использования сети. Таким образом, нет единого решения, приемлемого для всех организаций. Поэтому создание эффективного брандмауэра в соответствии с техническими условиями заказчика может оказаться трудной задачей.

Чтобы брандмауэр не замедлял работу сети, его аппаратное и программное обеспечение должно быть оптимизировано на решение конкретной задачи. К счастью, в большинство коммерческих маршрутизаторов включен быстродействующий механизм фильтрации, который выполняет большую часть работы. Администратор может сконфигурировать фильтр в маршрутизаторе так, чтобы маршрутизатор блокировал прохождение определенных дейтаграмм. После того как будет подробно описана работа фильтра, мы покажем, как на его основе создать брандмауэр. В конце главы будет показано, как можно использовать фильтр в сочетании с другим механизмом для обеспечения безопасного и в то же время гибкого процесса взаимодействия.

### 32.16. Фильтры пакетов

Во многих коммерческих маршрутизаторах реализован механизм, усовершенствующий обычную маршрутизацию и позволяющий администратору управлять обработкой пакетов. Этот механизм неофициально называется *фильтром пакетов*. Для его работы требуется, чтобы администратор определил, как маршрутизатор должен обрабатывать каждую дейтаграмму. Например, администратор может настроить маршрутизатор так, чтобы он *отфильтровывал* (т.е. блокировал) все дейтаграммы, полученные от определенного отправителя или предназначенные для использования в определенном приложении. При этом все другие дейтаграммы маршрутизатор будет перенаправлять к получателям как обычно.

Происхождение термина *фильтр пакетов* связано с тем, что механизм фильтрования не оставляет никаких следов о прохождении нежелательной дейтаграммы (например, записи в системном журнале или в списке полученных дейтаграмм). Фильтр обрабатывает каждую дейтаграмму отдельно. Получив дейтаграмму, маршрутизатор сначала пропускает ее через свой фильтр пакетов, а затем выполняет необходимую обработку этой дейтаграммы. Если дейтаграмма не удовлетворяет условиям фильтрации, маршрутизатор тут же ее аннулирует.

Поскольку в семействе протоколов TCP/IP ничего не сказано по поводу стандарта для фильтров пакетов, при реализации фильтра каждый разработчик маршрутизатора может создать фильтр, поддерживающий нестандартные возможности и выбрать для его конфигурации собственный интерфейс. В некоторых маршрутизаторах администратор может сконфигурировать работу фильтра для каждого интерфейса, в то время как в других одна конфигурация парамет-



Рис. 32.7. Маршрутизатор с двумя интерфейсами

ров фильтра применяется сразу для всех интерфейсов. Обычно при определении дейтаграмм, которые фильтр должен блокировать, администратор может использовать любую комбинацию IP-адресов отправителя и получателя, типа протокола, номера портов протокола отправителя и получателя. Пример параметров настройки фильтра для маршрутизатора с двумя интерфейсами (рис. 32.7) показан в табл. 32.2. Маршрутизатор, в котором поддерживается фильтр пакетов, составляет основной компонент брандмауэра.

Таблица 32.2. Пример параметров настройки фильтра

Номер интерфейса	IP-адрес отправителя	IP-адрес получателя	Протокол	Порт отправителя	Порт получателя
2	*	*	TCP	*	21
2	*	*	TCP	*	23
1	128.5.0.0/16	*	TCP	*	25
2	*	*	UDP	*	43
2	*	*	UDP	*	69
2	*	*	TCP	*	79

В этом примере администратор заблокировал все входящие дейтаграммы, предназначенные для популярных сетевых служб, а также один из видов отправляемых дейтаграмм. Фильтр блокирует все исходящие дейтаграммы, полученные от узла, IP-адрес сети которого соответствует 16-битовому префиксу 128.5.0.0, и предназначенные для удаленного сервера обработки электронной почты (порт 25 протокола TCP, соответствующий протоколу SMTP). Фильтр также блокирует входящие дейтаграммы, предназначенные для служб FTP (порт 21 протокола TCP), TELNET (порт 23 протокола TCP), WHOIS (порт 43 протокола UDP), TFTP (порт 69 протокола UDP) и FINGER (порт 79 протокола TCP).

### 32.17. Безопасность и параметры фильтра пакетов

Хотя в списке параметров конфигурации фильтра, приведенного в табл. 32.2, перечислено небольшое количество стандартных служб, которые должны блокироваться, такой подход не годится для создания эффективного брандмауэра по трем причинам. Во-первых, количество портов стандартных служб очень велико, и оно постоянно растет. Таким образом, если вносить в список каждую новую службу, то администратор должен постоянно модифицировать этот список. Если он что-либо забудет, то брандмауэр может стать уязвимым. Во-вторых, большая часть трафика объединенной сети не содержит пакеты, в которых указаны номера портов стандартных служб. Причина в том, что обычно номера портов службам присваиваются динамически, как, например, в службе *дистанционного вызова процедур* (*Remote Procedure Call*, или *RPC*). Кроме того, программисты могут выбирать нестандартные номера портов для собственных приложений типа клиент/сервер. В-третьих, даже если внести в список номера портов всех популярных служб, брандмауэр все равно останется уязвимым для *туннелирования*. При туннелировании можно обойти систему безопасности, если узел или маршрутизатор, относящиеся к внутренней части сети, будет принимать инкапсулированные дейтаграм-

мы из внешней части сети, удалять один уровень инкапсуляции и пересылать дейтаграммы службе, которая обычно блокирована брандмауэром.

Как эффективно использовать фильтр пакетов при создании брандмауэра? Для ответа на этот вопрос необходимо рассмотреть процесс фильтрации с другой стороны. Речь о том, что вместо определения параметров дейтаграммы, которая должна быть отфильтрована, брандмауэр должен по умолчанию блокировать все дейтаграммы, кроме тех, которые предназначены для указанных сетей, узлов и портов протоколов. Для них соединение с внешним миром будет разрешено. Таким образом, при настройке брандмауэра администратор руководствуется принципом: все что явно не разрешено — запрещено. Затем он должен тщательно рассмотреть стратегию информационной безопасности организации и в соответствии с ней открыть нужные порты. Во многих реализациях фильтров пакетов администратор должен определить набор дейтаграмм, которые надо пропускать, вместо набора дейтаграмм, которые надо блокировать. Сказанное выше можно подытожить так.

*Для того чтобы брандмауэр, использующий фильтрование дейтаграмм, был эффективным, он должен ограничивать прохождение дейтаграмм для всех IP-адресов отправителя и получателя, протоколов и номеров портов, кроме тех, которые указаны администратором явно. Для облегчения настройки брандмауэра можно использовать такие реализации фильтра пакетов, в которых администратор должен определить набор пропускаемых дейтаграмм вместо набора блокируемых дейтаграмм.*

### 32.18. Влияние ограничения доступа на клиентов

Казалось бы, многие потенциальные проблемы безопасности можно решить, запретив посторонним машинам доступ к серверу организации путем фильтрации дейтаграмм, прибывающих по неизвестному порту протокола. Однако применение такого брандмауэра имеет одно последствие: компьютер, находящийся во внутренней сети, не может стать клиентом и получить доступ к службе, запущенной на сервере, находящемся во внешней сети. Чтобы понять почему это так, напомним, что клиент, в отличие от сервера, не использует стандартный номер порта. После запуска программа клиента обращается к операционной системе за свободным номером порта протокола (т.е. таким номером, который не относится ни к стандартным портам, ни к портам, используемым на компьютере клиента в настоящее время). Пытаясь связаться с сервером, расположенным во внешней сети, клиент скорее всего создаст одну или несколько дейтаграмм и отправит их серверу. У каждой исходящей дейтаграммы в качестве номера порта получателя указывается один из стандартных номеров портов протокола, а в качестве порта отправителя — номер порта, полученный программой клиента у операционной системы. Когда такие дейтаграммы проходят через брандмауэр во внешнюю сеть, он не блокирует их. При формировании ответа сервер поменяет номера портов местами в заголовке дейтаграммы. При этом порт клиента становится портом получателя, а порт сервера — портом отправителя. Однако когда такая дейтаграмма достигнет брандмауэра, она будет блокирована, поскольку она адресована по неизвестному порту получателя. Эту важную идею можно сформулировать так.

*Если брандмауэр организации будет отфильтровывать все входящие дейтаграммы, посланные по неизвестному порту протокола, за исключением портов, соответствующих службам, которые организация сделала доступными извне, приложение, запущенное во внутренней сети, не может стать клиентом сервера, работающего во внешней сети.*

### 32.19. Доступ к Internet в обход брандмауэра через proxy-серверы

Очевидно, что не все организации могут настроить свои брандмауэры так, чтобы те блокировали дейтаграммы, поступающие по неизвестному номеру порта протокола. Бывают случаи, когда необходимо создать безопасный брандмауэр, который предотвратит нежелательный доступ из вне, и в то же время позволит пользователям внутренней сети получить доступ ко внешним службам. При этом необходимо выработать специальный механизм безопасности, который и является второй основной частью структуры брандмауэра.

В общем случае организация может обеспечить безопасный доступ ко внешним службам только через защищенный компьютер. Понятно, что сделать все компьютерные системы в организации защищенными задача не из легких. Поэтому обычно с каждым брандмауэром связывают один защищенный компьютер и устанавливают на этом компьютере набор шлюзов уровня приложения. Для того чтобы такой компьютер мог служить в качестве безопасного канала связи, его степень защиты должна быть чрезвычайно высока. Поэтому такой компьютер часто называют *бастионным узлом*. Место бастионного узла в общей системе безопасности организации показано на рис. 32.8. Бастионный узел обеспечивает безопасный доступ ко внешним службам. При этом от брандмауэра не требуется, чтобы он пропускал дейтаграммы, адресованные произвольным получателям во внутренней сети.

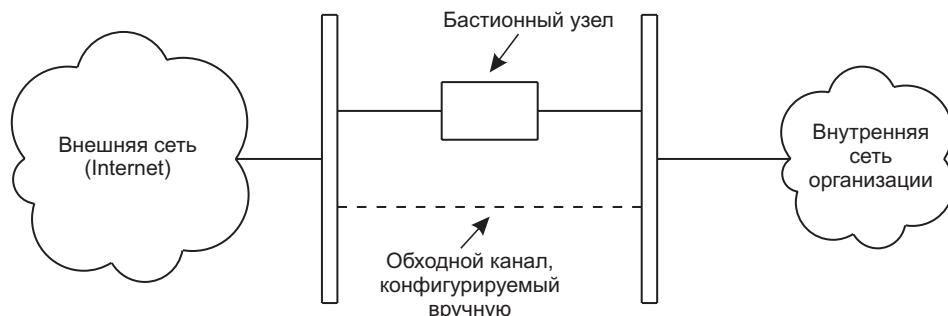


Рис. 32.8. Структурная схема брандмауэра, содержащего бастионный узел

Как показано на рис. 32.8, брандмауэр имеет два логических барьера. Внешний барьер блокирует весь входящий трафик, кроме (1) дейтаграмм, предназначенных для служб, запущенных на бастионном узле, которые организация хочет сделать доступными извне, и (2) дейтаграмм, предназначенных для клиентов, работающих на бастионном узле. Внутренний барьер также блокирует весь поступающий трафик, кроме дейтаграмм, отправленных бастионным узлом. В большинстве брандмауэрдов предусмотрен также обходной канал, конфигурируемый вручную. Он позволяет администраторам временно передавать весь трафик или некоторую его часть между внутренним и внешним узлом сети (например, при тестировании или поиске неисправности в сети).

Чтобы понять, как работает бастионный узел, рассмотрим доступ к службе Web. Поскольку брандмауэр не допускает, чтобы компьютер, подключенный ко внутренней сети, получал входящие дейтаграммы, пользователь не может использовать броузер для прямого доступа к службе Web, находящейся во внешней сети. Поэтому на бастионном узле запускается proxy-сервер службы Web. Каждый броузер внутри организации конфигурируется так, чтобы все запросы

посылались proxy-серверу. Всякий раз, когда пользователь выбирает ссылку или вводит URL, его браузер связывается с proxy-сервером. Proxy-сервер связывается с Web-сервером, получает указанную страницу, а затем доставляет ее пользователю.

### 32.20. Подробнее о структуре брандмауэра

Теперь, когда понятна основная идея построения брандмауэра, ее реализация не должна казаться сложной. Для реализации каждого из барьеров, показанных на рис. 32.8, потребуется маршрутизатор, поддерживающий фильтрацию пакетов<sup>4</sup>. Маршрутизаторы взаимодействуют с бастионным узлом с помощью отдельных сетей. Например, организация, сеть которой подключена к глобальной сети Internet, могла бы реализовать брандмауэр так, как показано на рис. 32.9.

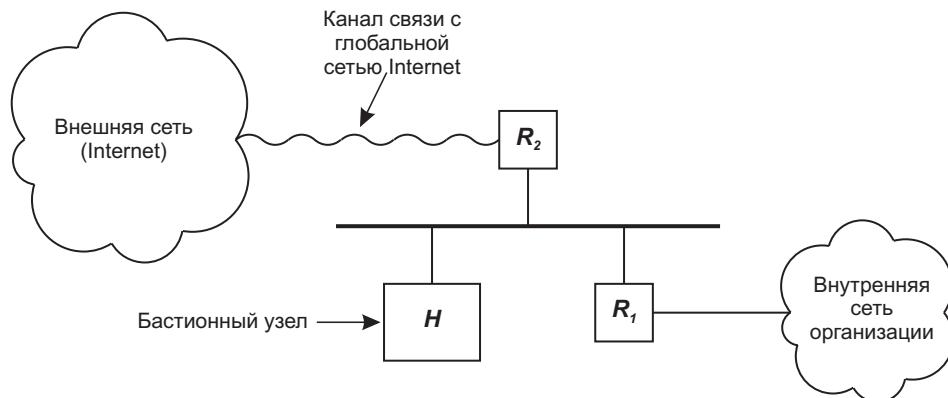


Рис. 32.9. Брандмауэр, реализованный на основе двух маршрутизаторов и бастионного узла. Один из маршрутизаторов связан с глобальной сетью Internet

Как показано на рис. 32.9, внешний барьер реализован на маршрутизаторе  $R_2$ . Он фильтрует весь входящий трафик, кроме дейтаграмм, предназначенных для бастионного узла,  $H$ . Внутренний барьер реализован на маршрутизаторе  $R_1$ , который изолирует остальную часть внутренней сети организации от посторонних. Он блокирует все входящие дейтаграммы, кроме тех, которые приходят от бастионного узла.

Конечно, надежность всего брандмауэра зависит от степени безопасности бастионного узла. Если злоумышленник сможет получить доступ к компьютеру бастионного узла, то по сути он получит полный доступ ко внутренней сети. Кроме того, злоумышленник может воспользоваться изъянами в системе безопасности операционной системы бастионного узла или одного из сетевых приложений, которые на нем выполняются. Таким образом, администраторы должны быть особенно внимательны при выборе и конфигурировании программного обеспечения для бастионного узла. Вывод такой.

*Хотя бастионный узел необходим для осуществления связи с внешней сетью в обход брандмауэра, надежность последнего зависит от степени безопасности бастионного узла. Злоумышленник, который знает изъяны в системе безопасности операционной системы бастионного узла, может получить полный доступ ко внутренней сети организации.*

<sup>4</sup> В некоторых организациях используется конфигурация с одиночным брандмауэром, в которой все функциональные возможности реализованы на одном физическом маршрутизаторе.

## 32.21. Тупиковая сеть

На первый взгляд может показаться, что на рис. 32.9 изображена одна лишняя сеть, которая соединяет два маршрутизатора и бастионный узел. Такую небольшую сеть часто называют *тупиковой* (*stub network*). Возникает вопрос: необходима ли тупиковая сеть, или можно подключить бастионный узел непосредственно к одной из внутренних сетей организации? Ответ зависит от величины трафика, ожидаемого извне. Тупиковая сеть изолирует внутреннюю сеть организации от трафика, создаваемого входящими дейтаграммами. В частности, поскольку маршрутизатор  $R_2$  пропускает все дейтаграммы, предназначенные для бастионного узла, внешние узлы могут посыпать произвольное количество таких дейтаграмм по тупиковой сети. Если пропускная способность внешнего канала, связывающего организацию с Internet, меньше пропускной способности тупиковой сети, то отдельная физическая сеть для тупиковой сети может быть просто не нужна. Однако, хотя применение отдельной физической сети несколько увеличивает стоимость брандмауэра, зато позволяет полностью обезопасить инфраструктуру внутренней сети от внешних вторжений.

## 32.22. Альтернативный вариант реализации брандмауэра

Брандмауэр, структура которого показана на рис. 32.9, хорошо работает, если организация имеет один выделенный последовательный канал, соединяющий ее с остальной частью глобальной сети Internet. Однако в некоторых случаях топология сетевого центра может быть несколько иной. Предположим, у компании есть три или четыре крупных клиента, каждому из которых надо заносить в базу данных или извлекать из нее большие объемы информации. При этом администрация компании хочет иметь только один брандмауэр и разрешить клиентам связываться с внешним миром<sup>5</sup>. Одна из возможных структур брандмауэра, которая обеспечивает несколько внешних соединений показана на рис. 32.10.

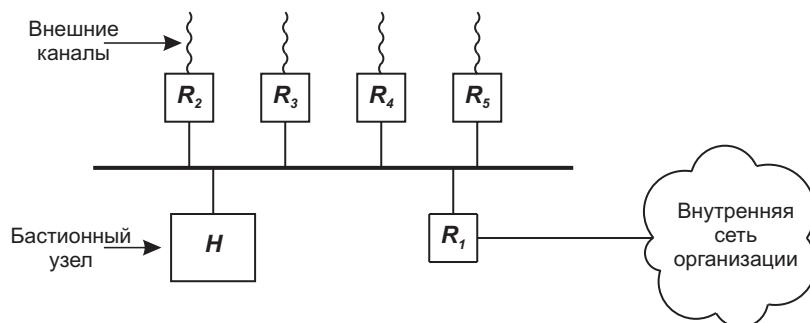


Рис. 32.10. Альтернативная структура брандмауэра, которая обеспечивает поддержку нескольких внешних соединений через один брандмауэр. Использование одного брандмауэра для нескольких соединений уменьшает стоимость системы

Как показано на рис. 32.10, в альтернативной структуре функции брандмауэра несколько расширены за счет поддержки нескольких внешних сетей, в которые ведут внешние каналы связи. Чтобы защитить внутреннюю корпоративную

<sup>5</sup> Использование одного брандмауэра для всех подключений является более дешевым и более легким для управления, чем использование отдельного брандмауэра для каждого соединения.

сеть, применяется маршрутизатор  $R_1$ . Он выполняет те же функции, что и прежде (см. рис. 32.9), т.е. блокирует все входящие дейтаграммы за исключением тех, которые отправлены бастионным узлом. Внешние сети подключаются к брандмауэру через маршрутизаторы  $R_2, \dots, R_5$ .

Чтобы понять, зачем в брандмауэре, поддерживающем несколько внешних соединений, обычно устанавливают один маршрутизатор для каждого соединения, напомним, что по определению все внешние сетевые центры не доверяют друг другу. Другими словами организация, управляющая брандмауэром, не доверяет полностью никакой из внешних организаций, и никакие внешние организации не доверяют друг другу полностью. Фильтр пакетов, поддерживаемый каждым из маршрутизаторов, к которому подключен внешний канал, должен быть сконфигурирован так, чтобы он ограничивал трафик для этого соединения. В результате владелец брандмауэра может гарантировать, что, хотя все внешние каналы подключены к одной общей сети, никакая дейтаграмма не попадет из одного внешнего канала в другой. Таким образом, организация, управляющая брандмауэром, может гарантировать своим клиентам, что их соединение безопасно. Все сказанное выше можно подытожить так.

*Подключение нескольких внешних сетевых центров к одному брандмауэру обычно выполняется через выделенные маршрутизаторы (т.е. каждый внешний канал подключается кциальному маршрутизатору). При этом можно предотвратить нежелательные потоки данных от одного внешнего сетевого центра к другому.*

### 32.23. Текущий контроль и регистрация

Текущий контроль — один из наиболее важных аспектов проектирования брандмауэра. Сетевой администратор, ответственный за брандмауэр, должен быть осведомлен о всех попытках злоумышленников взломать его систему безопасности. Если брандмауэр ничего не сообщает о произошедших инцидентах, то администратор никогда не узнает о проблемах.

Текущий контроль может быть *активным* или *пассивным*. При активном контроле брандмауэр уведомляет администратора как только происходит инцидент. Главным преимуществом активного контроля является скорость — администратор немедленно узнает о потенциальной проблеме. Главное неудобство заключается в том, что активные системы контроля зачастую выдают так много информации, что администратор не может целиком охватить ее или вовремя выявить проблему. Поэтому большинство администраторов предпочитает использовать пассивный контроль или комбинацию пассивного контроля с сообщениями системы активного контроля о некоторых особых опасных инцидентах.

При пассивном контроле брандмауэр регистрирует каждый инцидент в файле журнала на диске. Обычно система пассивного контроля регистрирует информацию о нормальном трафике, например статистику прохождения и количество отфильтрованных дейтаграмм. При этом администратор может получить доступ к журналу в любое время. Большинство администраторов использует для этого специальную компьютерную программу. Главное преимущество системы пассивного контроля заключается в том, что она регистрирует происходящие события. Проанализировав журнал сообщений, администратор может выявить узкие места в системе безопасности, а если что-то все-таки случится, он сможет восстановить хронологию предшествующих событий. Важно то, что администратор может периодически (например, анализировать журнал раз в день), чтобы определить, увеличивается или уменьшается со временем количество попыток несанкционированного доступа к сети организации.

## 32.24. Резюме

Проблемы безопасности в объединенной сети возникают из-за того, что она связывает между собой различные организации, которые не могут полностью доверять друг другу. Разработано несколько технологий, которые обеспечивают защищенность информации при передаче через объединенную сеть. Стандарт IPsec позволяет программистам выбрать одну из двух основных схем защиты. Первая обеспечивает аутентификацию дейтаграмм, а вторая — аутентификацию дейтаграмм и защиту их содержимого. При использовании протокола IPsec структура дейтаграммы изменяется. В нее добавляется заголовок аутентификации или инкапсулируются зашифрованные данные. В последнем случае к дейтаграмме добавляется заголовок и трейлер, а ее содержимое зашифровывается. В стандарте IPsec предусмотрен общий механизм, который позволяет каждой паре взаимодействующих объектов выбирать алгоритм шифрования. Поскольку шифрование часто используется при создании туннелей (например, в VPN), в стандарте IPsec определяется специальный режим зашифрованного туннеля.

Для управления доступом к объединенной сети используются брандмауэры. Каждый внешний канал организации подключается через брандмауэр для того, чтобы обезопасить внутреннюю сеть от несанкционированного доступа извне. Брандмауэр состоит из двух логических барьеров (внешнего и внутреннего) и защищенного компьютера, называемого бастионным узлом. В каждом барьере используется фильтр пакетов для ограничения трафика дейтаграмм. На бастионном узле запускаются службы, которые организация хочет сделать видимыми извне, а также proxy-серверы, которые позволяют пользователям внутренней сети получить доступ к внешним серверам. Параметры фильтрации конфигурируются в соответствии с принятыми в организации правилами информационной безопасности. Обычно брандмауэр блокирует все поступающие извне дейтаграммы, кроме тех, которые предназначены для бастионного узла.

Существует несколько способов реализации брандмаузеров. Выбор способа зависит от того, какое количество внешних каналов существует в организации. В большинстве случаев каждый барьер в брандмауэре реализуется на основе маршрутизатора, содержащего фильтр пакетов. В брандмауэре может также использоваться тупиковая сеть, которая позволяет изолировать внешний трафик от внутреннего.

## Материал для дальнейшего изучения

В середине 1990-ых годов, IETF объявила курс на повышение безопасности систем, работающих в Internet и потребовала, чтобы каждая рабочая группа рассмотрела возможность включения механизмов обеспечения безопасности в свои разработки. Поэтому было создано много документов RFC, посвященных проблемам безопасности в объединенной сети, в которых предложены стратегия, процедуры и механизмы обеспечения безопасности. Структура протоколов IPsec описана Кентом (Kent) и Эткинсоном (Atkinson) в [RFC 2401]. Заголовок аутентификации IPsec рассмотрен Кентом и Эткинсоном в [RFC 2402]. Безопасность передаваемой информации определена теми же авторами в [RFC 2406].

Система безопасности для конкретных протоколов уровня приложений описана в отдельных RFC. Например, система безопасности, основанная на представлениях (view-based), описана Вайненом (Wijnen) и др. в [RFC 2575]. А модель обеспечения безопасности, использующая идентификацию пользователя как субъекта доступа, представлена Блюменталем (Blumenthal) и Вайненом (Wijnen) в [RFC 2574]. Обе модели безопасности предназначены для использования в SNMPv3.

Брандмауэры и другие вопросы, связанные с безопасной работой объединенной сети TCP/IP рассматриваются Чесвиком (Cheswick) и Белловином (Bellovin)

в книге [22]. Служба аутентификации *kerberos* описана Колом (Kohl) и Нейманом (Neuman) в [RFC 1510]. Использование службы *kerberos* для аутентификации сеансов TELNET рассматривается Борманом (Borman) в [RFC 1411].

## Упражнения

- 32.1. Во многих сетевых центрах на бастионном узле запускается специальное программное обеспечение, просматривающее все входящие файлы перед тем, как допустить их использование внутри организации. С какой целью это делается?
- 32.2. Прочитайте описание фильтра пакетов, реализованного в одном из коммерческих маршрутизаторов, к которому вы можете получить доступ. Опишите возможности фильтра.
- 32.3. Зарегистрируйте в системном журнале весь трафик, поступающий на ваш сетевой центр. Проанализируйте его и определите процент дейтаграмм, в которых используются стандартные номера портов протокола (как отправителя, так и получателя). Удивляет ли вас полученный результат?
- 32.4. Если на вашем компьютере установлено криптографическое программное обеспечение, измерьте время, необходимое для того, чтобы зашифровать файл размером 10 Мбайт, передать его на другой компьютер и расшифровать его. Сравните результат со временем, необходимым для передачи незашифрованного файла.
- 32.5. Опросите пользователей своего сетевого центра, посылают ли они конфиденциальную информацию по электронной почте? Знают ли пользователи, что в протоколе SMTP сообщения посылаются в текстовом (ASCII) виде, и что любой, кто следит за сетевым трафиком, может прочитать содержимое их сообщения?
- 32.6. Опросите работников своего сетевого центра и выясните у них, какое количество персональных компьютеров и модемов они используют для пересылки входящей и исходящей информации. Узнайте у них, правильно ли они понимают стратегию информационной безопасности организации.
- 32.7. Может ли брандмауэр использоваться с другим набором протоколов, например AppleTalk или NetWare? Почему?
- 32.8. Можно ли использовать NAT в брандмауэре? Каковы могут быть последствия такого использования?
- 32.9. Военные распространяют закрытую информацию по принципу минимальной необходимости. Будет ли подобная стратегия информационной безопасности работать в вашей организации? Почему?
- 32.10. Назовите две причины, по которым люди, занимающиеся разработкой стратегии информационной безопасности организации, должны быть отделены от тех, кто управляет компьютерами организации и сетевыми системами.
- 32.11. В некоторых организациях брандмауэры используются для изоляции групп пользователей внутри организации. Приведите примеры, когда внутренние брандмауэры могут улучшить работу сети и примеры, когда внутренние брандмауэры могут ее ухудшить.
- 32.12. Если в вашей организации задействован протокол IPsec, выясните, какие в нем используются алгоритмы. Каков размер ключа?

# 33

## *Будущее протокола TCP/IP (IPv6)*

### **33.1. Введение**

Тесная связь развития технологии TCP/IP и глобальной сети Internet обусловлена несколькими причинами. Во-первых, Internet — наибольшая из созданных глобальных сетей, в которой используется семейство протоколов TCP/IP. Поэтому многие проблемы, связанные с размером сети, в Internet возникают до того, как они появляются в других сетях, использующих протоколы семейства TCP/IP. Во-вторых, исследования и разработка новых протоколов семейства TCP/IP финансируются частными и государственными компаниями, деятельность которых зависит от стабильной работы Internet. Поэтому они склонны финансировать связанные с этим проекты. В-третьих, поскольку большинство исследователей ежедневно пользуются Internet, они непосредственно заинтересованы в решении проблем. И эта заинтересованность приводит к улучшению качества предоставляемых услуг и расширению функциональных возможностей.

Учитывая то, что работа миллионов пользователей и десятков тысяч сетевых центров, разбросанных по всему миру, зависит от ежедневного использования глобальной сети Internet, может показаться, что Internet является полностью стабильным средством производства. Уже пройдена ранняя стадия развития, когда каждый пользователь был также экспертом. На современном этапе мало кто из пользователей до конца понимает технологию Internet. Но несмотря на видимость стабильности ни Internet, ни набор протоколов TCP/IP не являются неизменными. Организации находят новые применения технологии, исследователи решают возникающие проблемы сети, инженеры совершенствуют используемое оборудование. Короче говоря, технология продолжает развиваться.

Цель этой главы — обсудить происходящий эволюционный процесс, а также рассмотреть одно из наиболее значительных инженерных достижений: проект изменения протокола IP. Если проект будет внедрен поставщиками Internet-услуг, то это окажет значительное влияние на протоколы семейства TCP/IP и глобальную сеть Internet.

### **33.2. Зачем что-то менять?**

Базовая технология TCP/IP хорошо работала более двух десятилетий. Зачем же ее менять? Необходимость пересмотра стандартов протоколов связана с изменением технологий, лежащих в их основе, и распространением этих протоколов.

- *Новые компьютерные и коммуникационные технологии.* Компьютерное и сетевое оборудование продолжает развиваться. И как только новые технологии появляются, они сразу же внедряются в Internet.

- *Новые приложения.* Так как программисты постоянно придумывают новые способы использования протоколов TCP/IP, возникает необходимость в поддержке дополнительных протоколов. Например, привлечение внимания к IP-телефонии за последние годы дало толчок к созданию протоколов передачи данных в реальном масштабе времени.
- *Увеличение размеров и нагрузки.* Уже в течение многих лет происходит непрерывный экспоненциальный рост Internet. Сейчас размер объединенной сети удваивается каждые девять месяцев или даже быстрее. В 1999 году новый узел в Internet появлялся примерно каждые две секунды. Так же быстро вырос трафик, в связи с широким распространением анимированных изображений, видео- и аудиоклипов.

### 33.3. Новые правила

Распространяясь в новые страны, Internet существенно меняется: право на управление частью объединенной сети получают новые административные органы. Смена управления приводит к изменениям в административных правилах и делегирует механизмы внедрения этих правил. Как мы выяснили, и структура объединенной сети Internet, и используемые в ней протоколы, развиваются, все более удаляясь от централизованной модели. Развитие продолжается за счет того, что к Internet подключается большее количество национальных магистральных сетей. Это приводит к созданию все более сложных правил регулирования взаимодействием. С подобными проблемами сталкиваются многочисленные корпорации при объединении внутренних локальных сетей на основе протокола TCP/IP, поскольку они пытаются сначала определить правила взаимодействия, а затем развить механизмы внедрения этих правил. Таким образом, многие усилия исследователей и инженеров, связанные с протоколами семейства TCP/IP продолжают сосредоточиваться на поиске способов поддержки работы новых административных групп.

### 33.4. Причины пересмотра стандарта IPv4

В настоящее время в семействе протоколов TCP/IP и в глобальной сети Internet основной механизм передачи информации обеспечивается за счет протокола IP версии 4 (*IPv4*), который почти не изменился с момента его создания в конце 1970-х годов<sup>1</sup>. Успешное использование этой версии на протяжении многих лет свидетельствует о ее надежности и гибкости ее структуры. Со временем разработки стандарта IPv4 производительность процессоров увеличилась более чем на два порядка, стандартные объемы памяти возросли более чем в 100 раз, пропускная способность магистральной сети Internet увеличилась более чем в 7000 раз, появились новые технологии построения локальных сетей, количество узлов в объединенной сети Internet возросло от ничтожно малого до более чем 56 миллионов. Поскольку изменения произошли не в один момент, адаптация к ним была непрерывным процессом.

Несмотря на логичную структуру стандарта IPv4 в ближайшее время он должен быть заменен. В главе 10, “Бесклассовая адресация и подсети (CIDR)”, описана основная причина необходимости обновления протокола IP. Речь идет о том, что в ближайшем будущем возникнет проблема нехватки адресного пространства. Во времена разработки протокола IP 32-битового адресного пространства

---

<sup>1</sup> Версии с 1 по 3 никогда не были формально определены, а версия номер 5 была присвоена протоколу ST.

было вполне достаточно. Локальные сети использовались только в небольшом количестве организаций; еще меньше организаций было подключено к глобальным сетям. Сейчас же в большинстве корпораций средних размеров используются многочисленные локальные сети, а в наиболее крупных из них имеются корпоративные распределенные сети. Следовательно, даже при аккуратном распределении и использовании технологии NAT в ближайшее 32-битовое адресное пространство не сможет обеспечить потребности всемирной сети Internet с учетом ее ожидаемого роста после 2020 года.

Необходимость увеличения адресного пространства — не единственная (хотя и главная) причина пересмотра структуры семейства протоколов TCP/IP. В частности, чтобы протокол IP можно было использовать в приложениях, работающих в реальном масштабе времени, были предложены системы, в которых потоковая передача дейтаграмм осуществляется после предварительного резервирования ресурсов. С целью повышения надежности приложений для электронной коммерции, следующая версия протокола IP разрабатывается с уже встроенной поддержкой средств безопасности, таких как аутентификация.

### 33.5. Путь к новой версии протокола IP

Для того чтобы сформулировать новую версию стандарта протокола IP, проблемной группе IETF потребовалось много лет. Так как IETF создает *открытые* стандарты, она предложила поучаствовать в этом процессе всему сообществу Internet. Производители компьютеров, комплектующих и программного обеспечения, пользователи, менеджеры, программисты, работники телефонных компаний и индустрии кабельного телевидения — все излагали свои требования к следующей версии протокола IP и обсуждали различные предложения.

Многие из предложенных решений удовлетворяли отдельным целям или устраивали определенную часть сообщества. В одном из основных предложений предполагалось сделать протокол IP более гибким за счет увеличения его сложности и увеличения накладных расходов на обработку дейтаграмм. В другом решении предлагалось использовать модификацию протокола OSI CLNS. В третьем (кстати, одном из самых интересных проектов) предполагалось сохранение большинства принципов старого протокола IP и создание простых расширений для поддержки *большего* адресного пространства. Это проект называли *SIP*<sup>2</sup> (*Simple IP*). Он стал основой для расширенного варианта стандарта, в который включили идеи из других предложений. Расширенная версия была названа *Simple IP Plus (SIPP)* и, в конечном счете, легла в основу проекта следующей версии протокола IP.

Выбрать новую версию протокола IP было непросто. В результате огромной популярности глобальной сети Internet во всем мире возрос спрос на программные продукты, в которых используется протокол IP. Поэтому многие деловые круги видят открывающиеся экономические возможности и надеются, что новая версия протокола IP даст им преимущества в конкурентной борьбе. Кроме того, в этот процесс были вовлечены конкретные люди: некоторые в силу своего авторитета в технических вопросах, другие же воспринимали активное участие в нем, как веху в своей карьере. Поэтому дискуссии были жаркими.

---

<sup>2</sup> Аббревиатура *SIP* сейчас относится к протоколу инициализации сеанса связи (*Session Initiation Protocol*), который используется для установки связи (например, в IP-телефонии).

### 33.6. Название следующей версии протокола IP

Проблемная группа IETF решила присвоить пересмотренной версии протокола IP номер 6 и назвать его *IPv6*<sup>3</sup>, чтобы отличать ее от современной версии *IPv4*. Решение пропустить версию номер 5 было принято из-за ряда возникших ошибок и недоразумений. Например, IAB опубликовала документ, в котором ошибочно называла следующую версию протокола *IP версии 7*, что многих ввело в заблуждение. Экспериментальному протоколу, называемому *Stream Protocol (ST, или потоковый протокол)*, присвоили номер версии 5, что также привело к недоразумению. Некоторые подумали, что протокол ST был выбран в качестве замены протокола IP. В конечном счете, чтобы устранить путаницу, IETF выбрала номер версии 6.

### 33.7. Свойства протокола IPv6

Предложенный протокол IPv6 сохранил многие особенности, которые способствовали успеху версии IPv4. Фактически, разработчики считают протокол IPv6 модифицированной версией протокола IPv4. Например, в протоколе IPv6 по прежнему используется режим доставки дейтаграмм, не требующий установки соединения с получателем (т.е. каждая дейтаграмма маршрутизируется независимо); передающая сторона может выбирать размер дейтаграммы, а также должна установить максимальное количество переходов, которое может совершить дейтаграмма до того, как она будет аннулирована одним из промежуточных маршрутизаторов. Как будет показано ниже, протокол IPv6 также сохранил большинство возможностей, обеспечиваемых параметрами протокола IPv4, включая возможности фрагментации и маршрутизации от источника.

Несмотря на многие принципиальные сходства в IPv6 изменена большая часть внутренней структуры протокола. Например, в нем используются адреса с увеличенной разрядностью и добавлено несколько новых свойств. Более важным является то, что в протоколе IPv6 полностью пересмотрен формат дейтаграммы. Вместо полей параметров переменной длины в нем используется набор заголовков фиксированного формата. Подробности будут изучены нами после рассмотрения структуры основных изменений и причин их принятия.

Изменения, внесенные в протокол IPv6, можно разделить на семь категорий.

- *Увеличение адресного пространства.* Пожалуй, самое существенное изменение коснулось увеличения количества битов, занимаемых адресом. В протоколе IPv6 размер адреса увеличен в четыре раза, с 32 до 128 бит. Адресное пространство этого протокола настолько велико, что в обозримом будущем оно не может исчерпаться.
- *Расширенная иерархия адресов.* В IPv6 используется большее адресное пространство для создания дополнительных уровней иерархии адресов. В частности, в IPv6 можно определить иерархию провайдеров Internet, а также иерархическую структуру в пределах данного сетевого центра.
- *Гибкий формат заголовка.* В протоколе IPv6 используется совершенно новый и несовместимый со старым формат дейтаграммы. В отличие от заголовка фиксированного формата протокола IPv4, в протоколе IPv6 определен ряд дополнительных заголовков.
- *Улучшенный механизм параметров.* Как и в протоколе IPv4, в IPv6 в дейтаграмму может включаться дополнительная управляющая информация.

<sup>3</sup> В некоторых документах этот проект называют IPng, или IP — The Next Generation (протокол IP следующего поколения).

В IPv6 добавлен ряд новых параметров, которые обеспечивают дополнительные возможности, недоступные в IPv4.

- *Поддержка возможности расширения протокола.* Возможно, наиболее существенным изменением является то, что, в отличие от протокола IPv4, в котором жестко установлена внутренняя структура протокола, в IPv6 заложена возможность расширения. Благодаря этому проблемная группа IETF может адаптировать этот протокол к любому сетевому оборудованию или к новым приложениям.
- *Поддержка автоконфигурации и перенумерации.* В протоколе IPv6 предусмотрены средства, позволяющие автоматически назначать адреса компьютерам в изолированной сети. Это позволяет начать процесс сетевого взаимодействия без обращения к серверу за параметрами конфигурации или предварительной ручной настройки. В протоколе IPv6 также предусмотрены средства, позволяющие администратору динамически перенумеровывать сети.
- *Поддержка распределения ресурсов.* В протоколе IPv6 предусмотрено два средства для предварительного распределения сетевых ресурсов: концепция потока и спецификация дифференцированных служб. В последнем случае используется подход, аналогичный тому, что был в стандарте IPv4.

### 33.8. Общая структура дейтаграммы протокола IPv6

В протоколе IPv6 полностью изменился формат дейтаграммы. Дейтаграмма протокола IPv6 состоит из основного заголовка фиксированного размера, за которым может следовать несколько дополнительных заголовков, а после них — данные (рис. 33.1).



### 33.9. Формат основного заголовка дейтаграммы IPv6

Несмотря на поддержку большего адресного пространства, в основном заголовке протокола IPv6 содержится меньше информации, чем в заголовке дейтаграммы IPv4. Параметры и некоторые фиксированные поля, находящиеся в заголовке дейтаграммы IPv4, в IPv6 перенесены в дополнительные заголовки. В общем можно сказать, что изменения в формате заголовка дейтаграммы отражают изменения, сделанные в стандарте протокола.

- Изменилась граница выравнивания полей дейтаграммы (с 32 до 64 бит).
- Удалено поле длины заголовка, а вместо поля длины дейтаграммы используется поле длины передаваемых данных.
- Размер полей, содержащих адреса отправителя и получателя, увеличен до 16 октетов каждый.

- Информация о фрагментации перенесена из фиксированных полей основного заголовка в дополнительный заголовок.
  - Поле, в котором хранилось *время жизнидейтаграммы*, заменено полем, содержащим *максимальное количество переходов*.
  - Поле *типа обслуживания* заменено полем *класса трафика* и расширено полем, содержащим *метку потока*.
  - Поле *типа протокола* заменено полем, определяющим тип следующего заголовка.

Формат основного заголовка протокола IPv6 показан на рис. 33.2. Некоторые поля основного заголовка IPv6 непосредственно соответствуют полям заголовка IPv4. Так в первом 4-битовом поле указывается версия протокола. В дейтаграмме протокола IPv6 в этом поле всегда содержится значение 6. Как и в протоколе IPv4, адреса отправителя и получателя указываются в специальных полях. Однако в протоколе IPv6 под каждый адрес выделено не 4, а 16 октетов. Поле *максимального количества переходов* соответствует полю времени жизни дейтаграммы протокола IPv4. В отличие от IPv4, где значение времени жизни дейтаграммы интерпретируется как комбинация количества переходов и максимального времени передачи, в протоколе IPv6 соответствующее значение строго интерпретируется как максимальное число переходов, которое дейтаграмма может совершить, до того как она будет аннулирована.



*Рис. 33.2. Формат 40-октетного основного заголовка, расположенного в начале каждой дейтаграммы протокола IPv6*

Длина дейтаграммы в протоколе IPv6 указывается по-другому. Во-первых, так как размер основного заголовка фиксирован и равен 40 октетам, то в поле, содержащем длину самого заголовка, нет необходимости. Во-вторых, в протоколе IPv6 вместо поля, содержащего длину дейтаграммы (как в протоколе IPv4), используется 16-битовое поле, в котором указывается количество октетов, содержащихся в дейтаграмме, за исключением заголовка. Таким образом, в дейтаграмме протокола IPv6 может передаваться до 64К октетов данных.

Два поля в основном заголовке предназначены для принятия решений по пересылке дейтаграммы. Поле типа обслуживания, которое было в заголовке дейтаграммы протокола IPv4 заменено на поле класса трафика. Кроме того, в IPv6 появились новые механизмы, поддерживающие резервирование ресурсов и позволяющие маршрутизатору связывать каждую дейтаграмму со сделанным распределением ресурсов. В основе этих механизмов лежит абстрактное понятие *потока* (*flow*). Поток представляет собой маршрут, проложенный в объединенной сети, вдоль которого промежуточными маршрутизаторами гарантируется

связь определенного качества. В поле метки потока основного заголовка содержится информация, используемая маршрутизаторами для сопоставления дейтаграммы с определенным потоком, а также величина приоритета. Например, если двум приложениям необходимо передать видеоданные, то они могут создать поток, в пределах которого будет обеспечиваться соответствующие задержка и пропускная способность. С другой стороны, провайдер может запросить у абонентов спецификацию необходимого им качества связи, а затем с помощью потока ограничивать трафик, исходящий от определенного компьютера или приложения. Заметим, что потоки можно также использовать в пределах организации для управления сетевыми ресурсами и контроля над их равномерным распределением между приложениями. При сопоставлении дейтаграммы с определенным потоком маршрутизатором используется и адрес отправителя дейтаграммы, и идентификатор потока. Подведем итог.

*Каждая дейтаграмма протокола IPv6 начинается с 40-октетного основного заголовка, в котором содержатся поля адресов отправителя и получателя, поле максимального количества переходов, поле класса трафика, поле метки потока и поле типа следующего заголовка. Таким образом, в дейтаграмме IPv6, кроме данных, должно содержаться по крайней мере еще 40 октетов информации.*

### **33.10. Дополнительные заголовки дейтаграммы протокола IPv6**

В качестве компромисса между универсальностью и эффективностью разработчиками была выбрана следующая структура заголовка: за основным заголовком фиксированного формата следует набор необязательных дополнительных заголовков. Действительно, чтобы протокол IPv6 был универсальным, в него должны быть включены механизмы, поддерживающие такие функции, как фрагментация, маршрутизация от источника и аутентификация. С другой стороны, расположить поля фиксированного формата, поддерживающие все механизмы, в заголовке дейтаграммы было бы неэффективно, поскольку в большинстве случаев они не будут использоваться. А использование в IPv6 адресов большего размера делало бы это еще более неэффективным. Например, при передаче дейтаграмм в пределах одной локальной сети, заголовок, содержащий пустые поля адресов, может занимать значительную часть каждого фрейма. Разработчики понимали, что невозможно заранее предсказать, какие средства могут понадобиться в будущем.

Принцип дополнительных заголовков протокола IPv6 аналогичен полям параметров дейтаграммы IPv4. При этом отправитель может выбирать, какие дополнительные заголовки следует включить в дейтаграмму, а какие — опустить. В результате обеспечивается максимальная универсальность. Подведем итог.

*Дополнительные заголовки протокола IPv6 напоминают поля параметров дейтаграммы IPv4. В каждую дейтаграмму включаются только те дополнительные заголовки, которые необходимы для ее передачи.*

### **33.11. Анализ дейтаграммы протокола IPv6**

И в основном, и в дополнительных заголовках есть поле типа следующего заголовка. Значения этих полей используются программным обеспечением, которое установлено на промежуточных маршрутизаторах и у конечного получателя,

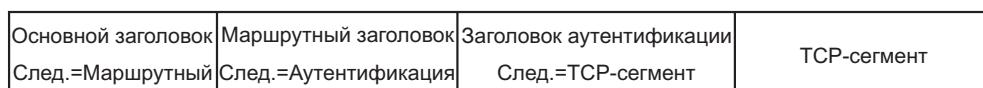
для анализа содержимого дейтаграммы во время ее обработки. Для извлечения информации, содержащейся в заголовках дейтаграммы протокола IPv6, необходимо последовательно проанализировать все заголовки. На рис. 33.3 показаны значения полей типа следующего заголовка для трех дейтаграмм, одна из которых не содержит дополнительного заголовка, вторая и третья содержат один и два дополнительных заголовка, соответственно.



(а)



(б)



(в)

*Рис. 33.3. Три дейтаграммы: только с основным заголовком (а); с основным и одним дополнительным заголовками (б); с основным заголовком и двумя дополнительными (в). Определить тип следующего заголовка можно по значению одноименного поля текущего заголовка*

Очевидно, что анализ дейтаграммы протокола IPv6, состоящей только из основного заголовка и данных, не займет много времени; он столь же эффективен, как и анализ дейтаграммы протокола IPv4. Более того, промежуточным маршрутизаторам нужно анализировать только дополнительный заголовок, который был добавлен предыдущим маршрутизатором (*hop-by-hop*), а остальные дополнительные заголовки обрабатываются только конечными получателями.

### 33.12. Фрагментация и сборка дейтаграммы в протоколе IPv6

Как и в протоколе IPv4, в IPv6 сборка фрагментированной дейтаграммы выполняется на машине конечного получателя. Тем не менее разработчики внесли изменения, которые позволяют избежать фрагментации дейтаграмм маршрутизаторами. Напомним, что в протоколе IPv4 любая дейтаграмма может быть разбита промежуточным маршрутизатором на фрагменты, если ее размер превышает значение MTU сети, через которую она будет передаваться. В протоколе IPv6 фрагментация является сквозной (*end-to-end*); промежуточным маршрутизаторам нет необходимости производить фрагментацию. Отправитель, выполняющий фрагментацию, может воспользоваться двумя возможностями. Во-первых, он может разбить дейтаграмму на фрагменты в соответствии с *минимальным гарантированным значением MTU*, которое равно 1280 октетам. Во-вторых, он может выполнить поиск *минимального значения MTU* по пути следования пакетов. В любом случае дейтаграмма разбивается отправителем таким образом, чтобы каждый фрагмент был меньше ожидаемого значения MTU по пути следования дейтаграммы.

В основном заголовке протокола IPv6 не содержится полей, аналогичных тем, которые используются для поддержки фрагментации в заголовке IPv4. При выполнении фрагментации отправитель помещает небольшой дополнительный заголовок после основного заголовка в каждый фрагмент. На рис. 33.4 показан формат *дополнительного заголовка фрагментации* (*Fragment Extension Header*).

	0	8	16	29	31
+4	Тип след. заголовка	Зарезервировано	Смещение фрагмента	RS	M
	Идентификация дейтаграммы				

Рис. 33.4. Формат дополнительного заголовка фрагментации

В протоколе IPv6 сохранились основные функциональные возможности протокола IPv4 в плане поддержки фрагментации. Как и раньше, смещение каждого фрагмента выражается числом, кратным 8 октетам (т.е. для определения истинного смещения фрагмента это число надо умножить на 8). Для поиска последнего фрагмента предназначено однобитовое поле *M*, аналогичное биту *More fragments* (дополнительные фрагменты) протокола IPv4. В поле *идентификации дейтаграммы* содержится уникальное значение, используемое получателем для группировки фрагментов<sup>4</sup>. Наконец, поле *RS* в настоящее время зарезервировано; его два бита устанавливаются в ноль во время передачи и игнорируются получателем.

### 33.13. Важность сквозной фрагментации

Введение сквозной фрагментации позволяет снизить нагрузку на промежуточные маршрутизаторы. При этом каждый из них может обрабатывать большее количество дейтаграмм за единицу времени. Действительно, для фрагментации дейтаграммы протокола IPv4 маршрутизатор должен затратить некоторое количество процессорного времени. Если большая часть получаемых дейтаграмм должна быть фрагментирована, степень использования центрального процессора маршрутизатора может достигать 100%. Введение сквозной фрагментации имеет одно важное следствие. Оно означает, что при этом не выполняется основополагающее предположение протокола IPv4 о том, что маршруты могут изменяться динамически.

Чтобы стала понятна важность сквозной фрагментации, напомним, что при разработке протокола IPv4 специально предусматривалась возможность изменения маршрута дейтаграмм. Например, если часть сети или один из маршрутизаторов вышли из строя, то трафик может направляться по другому пути. Основное преимущество такой системы — гибкость: изменение направления трафика может осуществляться без разрыва TCP-соединения и без уведомления об этом отправителя и получателя. Однако в протоколе IPv6 маршруты так легко изменяться не могут, поскольку при этом меняется значение MTU маршрута. Если MTU нового маршрута будет меньше исходного, то либо дейтаграммы должны разбиваться промежуточным маршрутизатором на фрагменты, либо отправитель должен быть проинформирован об этом. Проблему можно сформулировать следующим образом.

*При использовании сквозной фрагментации требуется, чтобы отправитель заранее определял значение MTU маршрута, ведущего к каждому получателю. Если размер дейтаграммы больше, чем значение MTU пути,*

<sup>4</sup> В протоколе IPv6 размер этого поля увеличен до 32 бит в связи с поддержкой высокоскоростных сетей.

*то исходящая дейтаграмма должна быть разбита отправителем на фрагменты. При сквозной фрагментации не допускается динамическое изменение маршрута следования дейтаграмм.*

Для решения проблемы, связанной с изменением значения MTU при изменении маршрутов, в протокол IPv6 добавлено новое ICMP-сообщение об ошибке. Когда маршрутизатор обнаруживает, что необходима фрагментация, отправителю посылается это сообщение. Получив такое сообщение, отправитель вновь определяет минимальное значение MTU маршрута. Затем дейтаграммы разбивают-ся на фрагменты в соответствии с новым значением.

### 33.14. Маршрутизация от источника в протоколе IPv6

В протоколе IPv6 за отправителем сохранена возможность определять неточный маршрут от источника. Однако в отличие от протокола IPv4, в котором маршрутизация от источника задается с помощью параметров дейтаграммы, в протоколе IPv6 используется отдельный дополнительный заголовок. Как показано на рис. 33.5, первые четыре поля маршрутного заголовка фиксированы. Значение поля *типа маршрута* определяет тип предоставляемой маршрутной информации. Единственный определенный на данный момент тип 0 соответствует неточной маршрутизации от источника. В области *данных, зависящих от типа маршрута*, указывается список адресов маршрутизаторов, через которые должна пройти дейтаграмма. Общее количество оставшихся в списке адресов маршрутизаторов указывается в специальном фиксированном поле заголовка. Общая длина дополнительного заголовка указывается в одноименном фиксированном поле маршрутного заголовка.

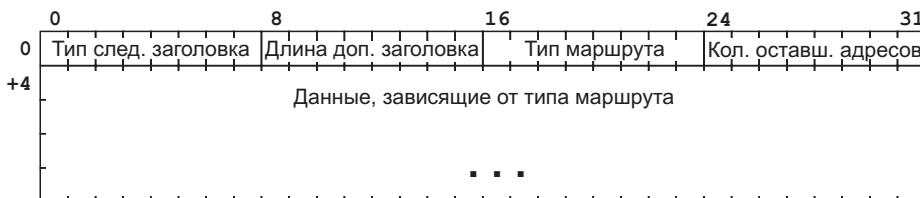


Рис. 33.5. Формат маршрутного заголовка протокола IPv6. В настоящее время определен только тип 0 (неточная маршрутизация от источника)

### 33.15. Параметры протокола IPv6

Может показаться, что в протоколе IPv6 параметры протокола IPv4 полностью заменены дополнительными заголовками. Однако разработчиками были предложены еще два дополнительных заголовка, содержащие различную информацию, не вошедшую в другие дополнительные заголовки. Этими заголовками являются *переходной (Hop By Hop) дополнительный заголовок* и *сквозной (End To End) дополнительный заголовок*. Из их названий следует, что в одном из них содержатся параметры, которые должны анализироваться при каждом переходе, а в другом — те, которые обрабатываются в конечном пункте назначения.

Несмотря на то что коды типов этих заголовков различаются, оба они имеют идентичный формат (рис. 33.6).

Как обычно, тип следующего заголовка указывается в специальном поле, расположенном в начале дополнительного заголовка. Поскольку размер заголовка, содержащего параметры протокола, имеет переменную длину, то она указывается

в специальном поле. Набор различных параметров протокола указывается в области данных дополнительного заголовка. На рис. 33.6 она обозначена как *Один или несколько параметров протокола*. На рис. 33.7 показано, как кодируется тип, длина и значение каждого отдельного параметра протокола<sup>5</sup>. Поля параметров никак не выравниваются и не дополняется пустыми символами.

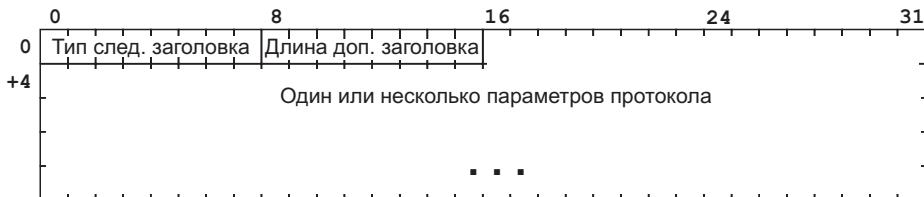


Рис. 33.6. Формат дополнительного заголовка, содержащий параметры протокола IPv6. Формат обоих типов заголовков (переходного и сквозного) одинаков. Различить их можно только по значению поля типа следующего заголовка, содержащегося в предыдущем заголовке



Рис. 33.7. Формат кодирования отдельного параметра в дополнительном заголовке протокола IPv6. Каждый параметр состоит из 1-октетного поля типа и 1-октетного поля длины, за которыми следует ноль или более октетов данных

Как показано на рис. 33.7, параметры протокола IPv6 имеют такой же формат, как и параметры протокола IPv4. В начале каждого параметра располагается 1-октетное поле типа, за которым следует 1-октетное поле длины. Если для параметра предусмотрены дополнительные данные, они помещаются после поля длины.

Два старших бита поля типа каждого параметра определяют реакцию узла сети или маршрутизатора на получение дейтаграммы, содержащей неизвестный параметр (табл. 33.1).

**Таблица 33.1. Значение двух старших битов поля типа параметра**

Значение	Описание
00	Пропустить этот параметр
01	Аннулировать дейтаграмму, не посыпать ICMP-сообщение
10	Аннулировать дейтаграмму, послать ICMP-сообщение отправителю
11	Аннулировать дейтаграмму, послать ICMP-сообщение для немногоадресатного адреса

Кроме того, от значения третьего бита поля типа зависит, будет ли параметр изменяться в процессе передачи дейтаграммы. Такая информация очень важна при аутентификации. Если содержимое параметра может изменяться в процессе передачи дейтаграммы, то при аутентификации значение ее полей полагаются равными нулю.

<sup>5</sup> В литературе такой тип кодирования называют TLV-кодированием. TLV — это сокращение от Type–Length–Value (Тип–Длина–Значение).

### **33.16. Размер адресного пространства протокола IPv6**

В соответствии с протоколом IPv6, каждый адрес занимает 16 октетов, что в четыре раза больше размера адреса, определяемого протоколом IPv4. Такой гигантский размер адресного пространства позволяет создать практически любую рациональную схему распределения адресов. Действительно, если разработчики позднее решат изменить схему адресации, то при таком размере подобное перераспределение сделать будет достаточно легко.

Чтобы представить себе размер этого адресного пространства, попробуем сравнить его с численностью населения Земли. Адресное пространство, определяемое протоколом IPv6, настолько велико, что каждому человеку на планете можно выделить такое количество адресов, что он сможет организовать собственную глобальную сеть, по величине не уступающую современной Internet. Другой путь — соотнести количество адресов с площадью поверхности Земли, которая примерно равна  $5,1 \times 10^8$  квадратных километров. Таким образом, на каждый квадратный метр земной поверхности приходится более  $10^{24}$  адресов. Еще один способ представить себе этот размер — оценить время, через которое может исчерпаться такое количество адресов. Рассмотрим, например, в течение какого времени можно будет распределить все возможные адреса. Известно, что 16-октетное число может принимать  $2^{128}$  значений. Таким образом, размер адресного пространства превышает  $3,4 \times 10^{38}$  адресов. Если каждую микросекунду распределять один миллион адресов, то адресное пространство исчерпается через  $10^{20}$  лет.

### **33.17. Шестнадцатеричная запись с разделением двоеточиями**

Применение адресов больших размеров решает проблему нехватки адресного пространства, но при этом возникает другая проблема — человеческий фактор. Дело в том, что люди, сопровождающие глобальные сети, должны анализировать, вводить и изменять адреса. Очевидно, что двоичная форма записи для этих целей непригодна. Запись же таких адресов в виде десятичных чисел, разделенных точками, которая использовалась в протоколе IPv4, также недостаточно компактна. Чтобы понять, почему, рассмотрим в качестве примера 128-битовое число, записанное в таком виде:

104.230.140.100.255.255.255.255.0.0.17.128.150.10.255.255

Чтобы сделать длинный адрес несколько компактнее и легче для ввода, разработчики протокола IPv6 предложили использовать шестнадцатеричную форму записи с разделением двоеточиями. Адрес представляется в виде последовательности 16-битовых чисел, представленных в шестнадцатеричной системе счисления и разделенных двоеточиями. Например, если значение адреса, записанное выше десятичными числами, перевести в шестнадцатеричную форму записи с разделением двоеточиями, то получится вот что:

68E6:8C64:FFFF:FFFF:0:1180:96A:FFFF

Очевидное преимущество шестнадцатеричной формы записи по сравнению с десятичной состоит в том, что при этом используется меньше цифр и разделительных знаков. Кроме того, в шестнадцатеричной форме записи с разделением двоеточиями предусмотрены два приема, повышающие удобочитаемость адреса. Первый прием называется *сжатие нулей*. Суть его состоит в том, что ряд повторяющихся нулей можно заменить парой двоеточий. Например, адрес:

FF05:0:0:0:0:0:0:B3

можно записать как:

FF05::B3

В стандарте указывается, что к любому адресу процедура сжатия нулей может применяться только один раз. В противном случае возможна неоднозначная интерпретация адреса. Сжатие нулей особенно эффективно при использовании с предложенной в стандарте схемой распределения адресов, поскольку при этом во многих адресах будут содержаться длинные цепочки нулей. Второй прием — включение в шестнадцатеричную форму записи с разделением двоеточиями десятичных суффиксов, разделенных точками. Ниже мы покажем, что во время перехода от протокола IPv4 к IPv6 предполагается использование такой комбинированной формы записи. Например, следующая строка является корректной формой записи адреса протокола IPv6:

0:0:0:0:0:0:128.10.2.1

Обратите внимание, что каждое из чисел, разделенных двоеточием, является 16-битовой величиной, тогда как разделяемые точкой десятичные числа представляют величину длиной в один октет. К вышеуказанному числу, конечно же, можно применить процедуру сжатия нулей, в результате чего получится эквивалентная строка, которая будет очень похожа на запись обычного адреса протокола IPv4:

::128.10.2.1

И наконец, протокол IPv6 позволяет применять для представления адреса расширенную форму записи CIDR-адресов. Другими словами, за строкой адреса через косую черту указывается десятичное число, которое определяет количество бит, составляющих адрес. Например, запись

12AB::CD30:0:0:0:0/60

определяет первые 60 бит адреса, или в шестнадцатеричной системе 12AB00000000CD3.

### 33.18. Три основных типа адресов протокола IPv6

Как и в протоколе IPv4, в IPv6 адрес соответствует определенному сетевому соединению, а не компьютеру. Поэтому процесс назначения адресов в протоколе IPv6 такой же, как и в IPv4. Маршрутизатору выделяется два или более адреса IPv6 (по количеству его сетевых интерфейсов), а узлу с одним сетевым соединением необходим только один адрес IPv6. В IPv6 также сохраняется (и расширяется) иерархия адресов IPv4, в которой часть адреса (префикс) идентифицирует физическую сеть. Однако, чтобы сделать распределение и изменение адресов более простым, в протоколе IPv6 можно назначить для одной сети несколько префиксов, а один компьютер может иметь несколько различных адресов, соответствующих одному и тому же интерфейсу. Кроме того, в протоколе IPv6 расширены, а в некоторых случаях и унифицированы, специальные адреса протокола IPv4. В общем случае адрес получателя дейтаграммы можно отнести к одной из трех категорий (табл. 33.2).

Таблица 33.2. Три типа адресов протокола IPv6

Тип адреса	Описание
Одноадресатный (unicast)	Определяет одного получателя в сети (компьютер или маршрутизатор). Дейтаграмма должна передаваться к месту назначения по кратчайшему маршруту

Окончание табл. 33.2

Тип адреса	Описание
Альтернативный (anycast)	Определяет группу компьютеров, возможно, расположенных в разных местах объединенной сети, которым назначен один общий адрес. Дейтаграмма должна передаваться по кратчайшему маршруту только одному (ближайшему) члену группы <sup>6</sup>
Многоадресатный (multicast)	Определяет группу компьютеров, возможно, расположенных в различных местах объединенной сети. Дейтаграмма, посланная по многоадресатному адресу будет доставлена каждому члену группы. При этом используется многоадресатный или широковещательный режим передачи сетевого оборудования (если, конечно он поддерживается)

### 33.19. Взаимозаменяемость широковещательной и многоадресатной передачи

В протоколе IPv6 не используются термины *широковещательная* (*broadcast*) и *направленная широковещательная* (*directed broadcast*) передача по отношению к передаче информации всем компьютерам физической сети или логической IP-подсети. Вместо этого используется термин *многоадресатная* (*multicast*) передача, а широковещательная передача рассматривается как частный случай многоадресатной передачи. Такой подход может показаться странным каждому, кто хоть немного знаком с принципами работы сетевого оборудования, поскольку режим широковещательной передачи поддерживается на уровне оборудования чаще, чем многоадресатной. На самом деле разработчики оборудования обычно рассматривают многоадресатную передачу как ограниченную форму широковещательной передачи. На аппаратном уровне многоадресатный пакет рассыпается всем компьютерам локальной сети точно так же, как и широковещательный пакет. После этого все многоадресатные пакеты, кроме тех, о которых со стороны программного обеспечения компьютера поступило указание принять, отфильтровываются интерфейсным оборудованием сетевой платы.

Теоретически выбор между многоадресатной передачей и ограниченными формами широковещательной передачи не принципиален, поскольку одна может эмулироваться с помощью другой. Другими словами, они в некоторой степени дублируют друг друга, обеспечивая одинаковые функциональные возможности. Чтобы понять это, рассмотрим эмуляцию одного режима передачи с помощью другого. Если поддерживается режим широковещательной передачи, то пакет может быть доставлен группе получателей следующим образом. Он посыпается всем машинам, на каждой из которых программным обеспечением принимается решение о том, принять или аннулировать пришедший пакет. Когда поддерживается многоадресатная передача, пакет может быть доставлен всем машинам, если предварительно объединить их в одну многоадресатную группу (группу всех компьютеров данной подсети), которая была рассмотрена в главе 17, “Режим многоадресатной передачи в объединенной сети”.

### 33.20. Проблема выбора, стоящая перед разработчиками

То что широковещательный и многоадресатный режимы передачи теоретически взаимозаменяемы, никак не облегчает разработчикам задачу выбора одного из них. Чтобы стало понятно, почему разработчики протокола IPv6 в качестве

<sup>6</sup> Альтернативные адреса ранее назывались *клUSTERНЫМИ* (*cluster*) адресами.

основной модели выбрали режим многоадресатной, а не широковещательной передачи, рассмотрим эту проблему с точки зрения создания приложений, а не на уровне используемого оборудования. Приложению необходимо обмениваться данными либо с другим приложением, либо с группой приложений. Непосредственный обмен данными с одним приложением наилучшим образом обеспечивается посредством одноадресатной передачи, а с группой приложений — с помощью широковещательной или многоадресатной передачи. Для обеспечения максимальной универсальности принадлежность к группе не должна определяться по адресу сетевого интерфейса, поскольку компьютеры, входящие в группу, могут располагаться в любом месте объединенной сети. Режим широковещательной передачи нельзя использовать для обмена данными со всеми компьютерами группы, которые разбросаны по глобальной сети, особенно когда размер сети со-поставим с размером современной сети Internet.

Поэтому неудивительно, что разработчики заранее зарезервировали некоторые многоадресатные адреса для использования вместо сетевых широковещательных адресов протокола IPv4. Таким образом, от каждого маршрутизатора требуется, чтобы он кроме собственного адреса принимал все дейтаграммы, адресованные локальной многоадресатной группе, в которую входят *все маршрутизаторы*.

### 33.21. Проект распределения адресного пространства протокола IPv6

Вопрос о том, как разделить на части адресное пространство протокола IPv6, вызвал много дискуссий. Существует два основных спорных вопроса: как управлять распределением адресов и как определять по адресу маршрут. При обсуждении первого вопроса внимание уделяется практической проблеме планирования иерархии административных единиц. В отличие от современной сети Internet, в которой используется двухуровневая иерархия (сетевые префиксы распределяются центральным административным органом Internet, а суффиксы сетевых адресов назначаются локальными организациями), огромное адресное пространство протокола IPv6 подразумевает создание одной или нескольких многоуровневых иерархий. При обсуждении второго вопроса основное внимание уделяется эффективности вычислительных алгоритмов. Дело в том, что маршрутизаторы должны анализировать каждую проходящую через них дейтаграмму и выбирать для нее кратчайший маршрут к получателю независимо от того, к какой из иерархий административных органов принадлежит этот адрес. Чтобы снизить стоимость высокоскоростных маршрутизаторов, необходимо оптимизировать алгоритм выбора кратчайшего пути. Тогда время обработки каждой дейтаграммы в маршрутизаторе будет минимальным.

Разработчики протокола IPv6 предложили разбить все адресное пространство на классы, по аналогии со схемой, используемой в протоколе IPv4 (табл. 33.3). Как и в протоколе IPv4 тип адреса определяется по его префиксу. Следует отметить, что, хотя первых восьми битов адреса вполне хватило бы для определения его типа, разработчики не стали разбивать адресное пространство на блоки одинакового размера.

Таблица 33.3. Проект распределения адресного пространства протокола IPv6

Двоичный префикс	Тип адреса	Часть адресного пространства
0000 0000	Зарезервировано (для совместимости с IPv4)	1/256
0000 0001	Не распределено	1/256

*Окончание табл. 33.3*

<i>Двоичный префикс</i>	<i>Тип адреса</i>	<i>Часть адресного пространства</i>
0000 001	Адреса NSAP	1/128
0000 010	Адреса IPX	1/128
0000 011	Не распределено	1/128
0000 1	Не распределено	1/32
0001	Не распределено	1/16
001	Сгруппированные глобальные одноадресатные адреса	1/8
010	Не распределено	1/8
011	Не распределено	1/8
100	Не распределено	1/8
101	Не распределено	1/8
110	Не распределено	1/8
1110	Не распределено	1/16
1111 0	Не распределено	1/32
1111 10	Не распределено	1/64
1111 110	Не распределено	1/128
1111 1110 0	Не распределено	1/512
1111 1110 10	Одноадресатные адреса, локальные в пределах канала передачи данных	1/1024
1111 1110 11	Одноадресатные адреса, локальные в пределах сетевого центра	1/1024
1111 1111	Многоадресатные адреса	1/256

Как видно из табл. 33.3, на сегодняшний день распределено только 15% адресного пространства. Оставшаяся часть будет задействоваться IETF по мере необходимости. Несмотря на кажущуюся разбросанность, при распределении блоков адресов учитывалась возможность ускорения их обработки в маршрутизаторе. Например, многоадресатный адрес легко отличить от одноадресатного адреса по значению старшего октета. В многоадресатном адресе все биты старшего октета равны единице, тогда как в одноадресатном адресе в старшем октете содержится смесь нулей и единиц.

### **33.22. Кодирование адресов протокола IPv4 и переход к адресам IPv6**

Хотя в табл. 33.3 против префикса 0000 0000 отмечено, что он *зарезервирован*, небольшую часть адресов из этого блока разработчики планируют использовать для кодирования адресов, определенных протоколом IPv4. В частности, любой адрес, начинающийся с 80 нулевых бит, за которыми следуют 16 бит из единиц или 16 бит из нулей, содержит адрес IPv4 в младших 32 битах. Значение соответствующего 16-битового поля показывает, есть ли у данного узла еще и одноадресатный адрес протокола IPv6. На рис. 33.8 показаны оба возможных варианта.

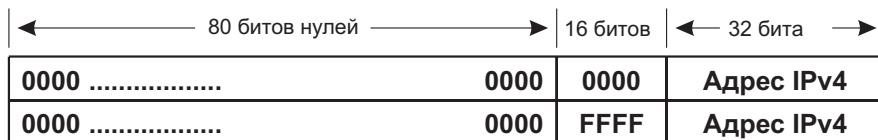


Рис. 33.8. Кодирование адресов IPv4 в протоколе IPv6. В 16-битовом поле содержится 0000, если узел имеет также одноадресатный адрес IPv6, и FFFF — в противном случае

Описанное выше кодирование будет использоваться при переходе от протокола IPv4 к IPv6 по двум причинам. Во-первых, программное обеспечение компьютера, использующее протокол IPv4, может быть обновлено на программное обеспечение, использующее протокол IPv6, еще до того, как машина получит адрес IPv6. Во-вторых, у компьютера с действующим программным обеспечением на основе протокола IPv6 может возникнуть необходимость связаться с компьютером, на котором используется только программное обеспечение на основе протокола IPv4.

Кодирование адресов IPv4 в адреса протокола IPv6 не решает проблему взаимодействия программного обеспечения, поддерживающего разные версии протокола, необходим еще механизм их преобразования. При использовании транслятора адресов компьютер, использующий протокол IPv6, должен сгенерировать дейтаграмму, в которой содержится закодированный IPv4 адрес получателя. Затем компьютер посылает дейтаграмму транслятору адресов, который, используя протокол IPv4, отправляет ее получателю. Получив ответ, транслятор адресов преобразовывает дейтаграмму протокола IPv4 в дейтаграмму протокола IPv6 и отсылает ее отправителю с помощью протокола IPv6.

На первый взгляд может показаться, что преобразование протокольных адресов будет приводить к некорректной работе системы, поскольку протоколы более высокого уровня проверяют неизменность адреса. В частности, в протоколах TCP и UDP при вычислении контрольной суммы используется псевдоголовка. В нем содержатся адреса как отправителя, так и получателя, поэтому изменение этих адресов может повлиять на вычисление контрольной суммы. Поэтому разработчики должны были предусмотреть возможность взаимодействия протоколов TCP и UDP на машинах, использующих IPv4, с соответствующими транспортными протоколами на машинах, использующими IPv6. Чтобы избежать несовпадения контрольных сумм, при кодировании адресов протокола IPv4 в протоколе IPv6 алгоритм вычисления контрольной суммы выбирается так, чтобы 16-битовое дополнение контрольной суммы до единицы было одинаково для обоих типов адресов. Выделяя главное, можно сказать вот что.

*Кроме выбора технических деталей нового протокола Internet, работа проблемной группы IEFT над протоколом IPv6 была сосредоточена на поиске способа перехода от текущего протокола к новому. В частности, в текущем варианте протокола IPv6 предусмотрена возможность кодирования адреса протокола IPv4 в адрес IPv6 таким образом, что преобразование адреса не изменяет контрольной суммы псевдоголовка.*

### 33.23. Неопределенные адреса и адреса петли обратной связи

Как и в протоколе IPv4, некоторым адресам протокола IPv6 придается специальное значение. На пример, адрес, состоящий из одних 0:

0:0:0:0:0:0:0:0

является неопределенным. Его нельзя назначить ни одному компьютеру или использовать в качестве адреса получателя. Он используется как адрес отправителя только во время начальной загрузки компьютера, которому еще не назначен реальный адрес.

Подобно протоколу IPv4, у IPv6 есть *адрес петли обратной связи*, который используется для тестирования программного обеспечения. В протоколе IPv6 таким адресом является

0:0:0:0:0:0:1

Любая дейтаграмма, посланная по адресу петли обратной связи, не покидает пределов локальной машины. Этот адрес никогда не следует использовать как адрес получателя в исходящей дейтаграмме.

### 33.24. Иерархия одноадресатных адресов

Одним из наиболее важных изменений в протоколе IPv6, по сравнению с IPv4, является стратегия распределения одноадресатный (unicast) адресов и полученная в результате иерархия адресов. Напомним, что в оригинальной системе адресации протокола IPv4 использовалась двухуровневая иерархия, согласно которой адрес делился на глобально уникальный префикс и суффикс. В протоколе IPv6 данная концепция расширена за счет принятия иерархии адресов с тремя концептуальными уровнями, как показано в табл. 33.4. На практике эта иерархическая структура адреса содержит дополнительные уровни.

Таблица 33.4. Три концептуальных уровня иерархии одноадресатных адресов протокола IPv6

Уровень	Назначение
1	Глобальная открытая топология
2	Отдельный сетевой центр
3	Отдельный сетевой интерфейс

Два нижних уровня концептуальной иерархии наиболее легки для понимания, поскольку они соответствуют реально существующим объектам. Самому нижнему уровню иерархии соответствует одиночное соединение между компьютером и сетью. Среднему уровню иерархии соответствует несколько компьютеров и сетей, объединенных в один *сетевой центр (site)*, подразумевающий как тесную физическую взаимосвязь, так и организацию, которая владеет и управляет оборудованием. Как будет показано ниже, такую схему адресации используют как большие, так и малые сетевые центры, поскольку она подходит для сложной внутренней структуры.

Для обеспечения гибкости верхний уровень иерархии, обозначенный в табл. 33.4 как *глобальная открытая топология*, точно не определяется. В общем, его можно понимать как “сегмент” глобальной сети Internet, открытый для общего доступа. Вырисовываются два типа открытой топологии. Один из них соответствует крупному провайдеру Internet, который предоставляет своим потребителям, называемым *абонентами*, услуги по передаче данных на большие расстояния. Второй тип соответствует недавно образованной структуре, называемый *коммутатором (exchange)*. По замыслу разработчиков, коммутаторами будут выполняться две функции: выступать в качестве точки доступа к сети (NAP), т.е. связывать крупных провайдеров Internet и обеспечивать передачу трафика между ними, и обслуживать отдельных абонентов (в отличие от современных NAP,

обслуживающих только крупных провайдеров). Это означает, что коммутаторами будут выделяться адреса абонентам. Главное преимущество выделения адресов коммутатором в том, что такие адреса не зависят от провайдера Internet, т.е. абоненты смогут свободно менять провайдеров.

### **33.25. Структура сгруппированных глобальных одноадресатных адресов**

Права на распределение адресов в протоколе IPv6 плавно переходят от одного уровня к другому вниз по иерархии. Каждой организации верхнего уровня (например, провайдеру Internet или коммутатору) выделяется уникальный префикс. Когда некоторая организация становится абонентом провайдера верхнего уровня, ее сетевому центру присваивается уникальный номер. В конце концов, сетевой администратор должен назначить этот номер каждому сетевому соединению. Чтобы добиться эффективной маршрутизации, для каждого назначения в адресе резервируются последовательные наборы битов. На рис. 33.9 показан формат *группированных глобальных одноадресатных адресов* (*aggregatable global unicast address*).



*Рис. 33.9. Разделение группированного глобального одноадресатного адреса, соответствующего протоколу IPv6, на поля с указанием их соответствия трехуровневой модели иерархии*

Трехбитовое поле, обозначенное на рис. 33.9 как  $P$ , соответствует префиксу формата, значение которого для сгруппированного глобального одноадресатного адреса равно 001. Восьмибитовое поле  $RES$  зарезервировано для дальнейшего использования, и в нем содержатся нули. Оставшиеся поля адреса предназначаются для обеспечения эффективной маршрутизации. В частности, поля, соответствующие самому высокому уровню иерархии, сгруппированы, и в них включены старшие биты адреса. В поле  $TLA\ ID$  содержится идентификатор *группировки верхнего уровня* (*Top-Level Aggregation*), т.е. уникальный идентификатор, назначаемый владельцам адреса (т.е. провайдеру Internet или коммутатору). Владелец адреса использует поле  $NLA$  для обозначения *группировки следующего уровня* (*Next-Level Aggregation*), т.е. для идентификации отдельного абонента.

Шестнадцатибитовое поле SLA ID (*Site-Level Aggregation*, или *группировка на уровне сетевого центра*) предназначено для использования отдельным сетевым центром. Разработчики предполагают, что оно будет использоваться подобно полю подсети, предусмотренному протоколом IPv4. Таким образом, сетевой центр, состоящий из небольшого количества сетей, может трактовать значение этого поля как идентификатор сети. Сетевой центр с большим количеством сетей может использовать его для разбиения сетей на группы, которые затем могут быть организованы в иерархию. Для создания одноуровневой иерархии в сетевом центре организация должна использовать префикс для идентификации группы, и суффикс — для идентификации отдельной сети в этой группе. Как и в случае организации подсетей протокола IPv4, деление на группы улучшает эффективность маршрутизации, поскольку в таблице маршрутизации указываются только маршруты к другим группам, а не к индивидуальным сетям.

### 33.26. Идентификаторы интерфейса

Как показано на рис. 33.9, младшие 64 бита сгруппированного одноадресатного адреса протокола IPv6 идентифицируют конкретный сетевой интерфейс. В отличие от протокола IPv4, суффикс в IPv6 выбран достаточно большим, чтобы можно было непосредственно закодировать в протокольном адресе физический адрес платы сетевого интерфейса. Кодирование физического адреса в протокольном адресе IPv6 имеет два преимущества. Во-первых, в протоколе IPv6 для преобразования IP-адреса в физический адрес больше не нужно использовать протокол ARP. Вместо него IPv6 предусматривает использование *протокола обнаружения соседних узлов (neighbor discovery protocol)*, который входит в новую версию протокола ICMP (*ICMPv6*). Он позволяет узлу сети определить, какие компьютеры подключены вместе с ним к одной физической сети. Во-вторых, для обеспечения взаимодействия всеми компьютерами должен использоваться один и тот же алгоритм кодирования физических адресов. Следовательно, в стандарте протокола IPv6 должно быть четко указано, как кодируются различные формы физических адресов. В простейшем случае физический адрес непосредственно размещается в адресе протокола IPv6. В отдельных случаях для кодирования адреса используются более сложные алгоритмы.

Ниже приведены два примера кодирования физических адресов, которые помогут понять принцип. Например, Институтом инженеров по электротехнике и электронике (IEEE) определен стандартный 64-битовый формат глобально уникального адреса, называемый *EUI-64*. Единственное изменение, которое необходимо выполнить при кодировании адреса стандарта EUI-64 в адрес протокола IPv6 состоит в изменении 6-го бита в старшем октете адреса. Он показывает, является ли адрес глобально уникальным.

Для обычного 48-битового адреса Ethernet необходимы более сложные преобразования (рис. 33.10). Как видно из рисунка, биты исходного физического адреса в закодированной форме не располагаются рядом. Вместо этого посередине результирующего адреса помещается 16-битовое шестнадцатеричное значение 0xFFFF. Кроме того, 6-ой бит, который указывает, является ли адрес глобальным, изменен с 0 на 1. Оставшиеся биты адреса, включая бит группы (обозначенный *g*), идентификатор компании — производителя интерфейса (обозначенный *c*) и поле расширения производителя, копируются, как показано на рис. 33.10. Расширение, выбранное производителем, для однозначной идентификации устройства помещается в младшие 24 бита адреса.

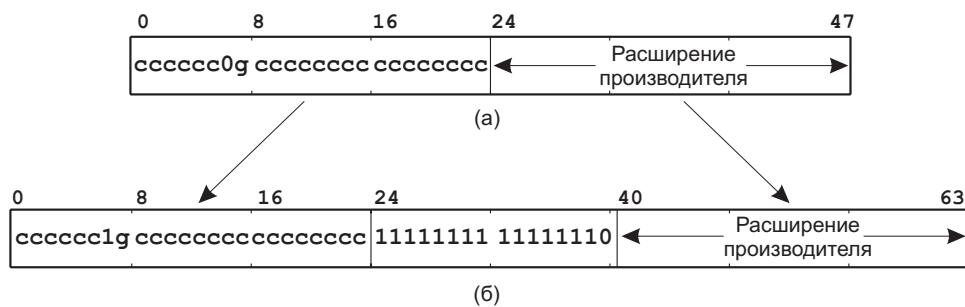


Рис. 33.10. Формат 48-битового адреса стандарта IEEE 802, используемого в сети Ethernet (а); кодирование этого адреса в младших 64-х битах одноадресатного адреса в соответствии с требованиями протокола IPv6 (б)

### **33.27. Дополнительный уровень иерархии**

Несмотря на то что в формате одноадресатного адреса, показанного на рис. 33.9, уже заложена строгая иерархия адресного пространства, в ней можно создать много дополнительных уровней. Например, биты поля NLA ID могут использоваться для создания иерархии провайдеров. Аналогично, 16 бит поля SLA ID можно разделить на части для создания иерархии в пределах организации. Поскольку эти поля имеют достаточно большой размер, то по сравнению с механизмом подсетей протокола IPv4, обеспечивается большая универсальность. Организация может принять решение создать двухуровневую иерархию, состоящую из зон и подсетей, входящих в каждую зону. В другом случае организация может выбрать трехуровневую иерархию, состоящую из зон, подзон и подсетей, образующих подзоны.

### **33.28. Локальные адреса**

Кроме глобальных одноадресатных адресов, описанных выше, в протоколе IPv6 предусмотрены дополнительные префиксы для одноадресатных адресов, имеющих локальную область действия. Как показано в табл. 33.3, в стандарте определены два типа локальных адресов: *по отношению к каналу передачи данных* (т.е. ограниченные одной физической сетью) и *по отношению к сетевому центру* (т.е. ограниченные одним сетевым центром). Область действия этих адресов учитывается маршрутизаторами: дейтаграммы, содержащие локальные адреса, не передаются ими за пределы указанной области.

Использование локальных адресов способствует решению двух проблем. Адреса, локальные по отношению к каналу передачи данных, обеспечивают взаимодействие машин в одной физической сети. Тем самым устраняется опасность передачи дейтаграммы через внешнюю объединенную сеть. Например, при обнаружении соседних узлов маршрутизатор, поддерживающий протокол IPv6, должен использовать локальный адрес по отношению к каналу связи. Правила области действия адресов определяют, что сообщения протокола определения соседних узлов будут получать только компьютеры, находящиеся в одной сети с отправителем. Точно так же локальные по отношению к каналу связи адреса могут использоваться для связи компьютеров, подключенных к *изолированной сети* (т.е. сети, к которой не подключены маршрутизаторы).

В отличие от дейтаграммы, содержащей локальный по отношению к каналу связи адрес, маршрутизаторы могут передавать дейтаграммы, содержащие локальные по отношению к сетевому центру адреса, по всем сетям организации. Однако маршрутизаторы препятствуют передаче таких дейтограмм в глобальную сеть Internet. Таким образом, эти адреса соответствуют тому, что в протоколе IPv4 называется *частными, или немаршрутизуемыми* адресами. Организация может назначать и использовать локальные по отношению к сетевому центру адреса в пределах всех своих внутренних сетей без официального получения глобально уникального префикса.

### **33.29. Автоконфигурация и перенумерация сетей**

При разработке протокола IPv6 был предусмотрен режим *автоконфигурации, не требующий сервера*<sup>7</sup>. Благодаря этому сетевым администраторам нет необходимости предварительно назначать адреса компьютерам для того, чтобы они

---

<sup>7</sup> Такой режим также называется автоконфигурацией без учета состояния.

могли начать взаимодействовать. Режим автоконфигурации оказался возможным и эффективным благодаря описанным выше двум механизмам: локальной адресации относительно канала связи и кодированию физических адресов в протокольном адресе IPv6. Перед посылкой первого пакета в сеть компьютер генерирует свой локальный относительно канала связи адрес путем комбинирования локального префикса 1111 1110 10 с 54-мя нулевыми битами и 64-битовым идентификатором своей платы сетевого интерфейса.

После проверки локального адреса на уникальность он используется компьютером для отправки специального пакета маршрутизатору, в котором запрашивается дополнительная информация. Если к данной физической сети подключен маршрутизатор, то в ответ на запрос он пришлет *извещение о маршрутизаторе*. В этом сообщении указываются префиксы, которые могут быть использованы в локальных относительно сетевого центра и глобальных адресах. После получения извещения о маршрутизаторе, компьютер назначает его отправителю своим стандартным маршрутизатором. Наконец, в извещении есть специальный флагок, указывающий на то, может ли компьютер воспользоваться полученными данными для выполнения автоконфигурации, либо он должен продолжить настройку своих параметров с помощью обычного режима *управляемого конфигурирования* (т.е. протокола DHCP).

Для упрощения процесса перенумерации сетей в протоколе IPv6 маршрутизатор может ограничить время использования префикса компьютером. Поэтому в извещении для каждого префикса указываются два значения: действительное и предпочтительное время жизни. При этом узел сети должен анализировать дополнительные сообщения об извещении маршрутизатора. По истечении предпочтительного времени жизни префикса он остается работоспособным, но узел по возможности должен воспользоваться другим префиксом для продолжения сеанса связи. По истечении действительного времени жизни префикса узел сети должен немедленно прекратить его использование, даже если в этот момент происходит обмен данными с другими компьютерами.

### 33.30. Резюме

Проблемной группой IETF разработано новое поколение протокола IP. Этот протокол называется IPv6, потому что ему был присвоен шестой номер версии. В IPv6 сохранены основные особенности современного протокола IPv4, однако внесено множество изменений. Как и IPv4, протокол IPv6 обеспечивает негарантированную доставку дейтаграмм без установки соединения с получателем. Однако формат дейтаграммы протокола IPv6 отличается от формата IPv4. Кроме того, протоколом IPv6 обеспечиваются такие новые возможности, как аутентификация и поддержка идентификаторов потока.

В протоколе IPv6 каждая дейтаграмма состоит из набора заголовков, за которыми следуют данные. Дейтаграмма всегда начинается с 40-октетного основного заголовка, в котором содержится адрес отправителя и получателя, класс трафика и идентификатор потока. За основным заголовком могут следовать дополнительные заголовки, за которыми расположены данные. Дополнительные заголовки необязательны — в протоколе IPv6 они используются для передачи большей части информации, которая кодируется в виде параметров в протоколе IPv4.

Длина адреса в протоколе IPv6 равна 128 битам. Это обеспечивает настолько большое адресное пространство, что оно не может быть исчерпано в обозримом будущем. По префиксу адреса протокола IPv6 можно определить его тип и значение остальных частей адреса. Кроме традиционных одно- и многоадресатных адресов, в IPv6 определены также альтернативные адреса. Один такой адрес может быть

назначен группе компьютеров; посланная по этому адресу дейтаграмма передается только одному компьютеру из группы (т.е. ближайшему к отправителю).

В протоколе IPv6 поддерживаются режимы автоконфигурации компьютера и перенумерации сети. Каждым узлом изолированной сети генерируется уникальный локальный относительно канала передачи данных адрес, который затем может им использоваться для обмена данными с другими компьютерами, подключенными к той же сети. Такой адрес может также использоваться узлом сети для обнаружения маршрутизаторов и получения информации о локальных относительно сетевого центра и глобальных префиксах сетей. Для упрощения процесса перенумерации сетей, для всех префиксов указывается время жизни, по истечении которого узел должен воспользоваться новым префиксом.

## Материал для дальнейшего изучения

Информация, имеющая отношение к протоколу IPv6, приведена во многих документах RFC. Дилинг (Deering) и Хайнден (Hinden) в [RFC 2460] описывают основной протокол. Томсон (Thomson) и Нартен (Narten) в [RFC 2462] описывают режим автоконфигурации без учета состояния. Нартен, Нордмарк (Nordmark) и Симпсон (Simpson) в [RFC 2461] обсуждают протокол обнаружения соседних узлов. Конта (Conta) и Дилинг [RFC 2463] анализируют сопутствующий протоколу IPv6 протокол ICMPv6. Кроуфорд (Crawford) в [RFC 2464] описывает процесс инкапсуляции дейтаграмм протокола IPv6 для передачи по сетям Ethernet.

Многие документы RFC посвящены проблемам адресации протокола IPv6. Хайнден и Дилинг в [RFC 2373] описывают структуру основной системы адресации и приводят описание префиксов сетей. Хайнден, О’Делл (O’Dell) и Дилинг в [RFC 2374] рассматривают формат глобального одноадресатного адреса. Хайнден и Дилинг в [RFC 2375] анализируют процесс назначения многоадресатных адресов. Джонсон (Johnson) и Дилинг в [RFC 2526] описывают зарезервированные альтернативные адреса. Информацию о 64-битовом формате EUI можно найти по адресу: <http://standards.ieee.org/regauth/oui/tutorials/EUI64.html>.

## Упражнения

- 33.1. В современном стандарте протокола IPv6 не предусмотрено поле контрольной суммы заголовка. Какие преимущества и недостатки связаны с этим?
- 33.2. Как должны быть упорядочены дополнительные заголовки, чтобы минимизировать время обработки данных?
- 33.3. Несмотря на то что адреса протокола IPv6 назначаются согласно определенной иерархии, маршрутизатор не должен полностью анализировать адрес для выбора маршрута. Придумайте алгоритм и соответствующую структуру данных для выполнения эффективной маршрутизации. (*Подсказка.* Рассмотрите метод самого длинного пути.)
- 33.4. Докажите, что нет необходимости в 128-битовом IP-адресе и что 96-битовый адрес обеспечивает достаточный размер адресного пространства.
- 33.5. Предположим, ваша организация собирается перейти на использование протокола IPv6. Разработайте схему адресации, которая будет использоваться в организации для назначения адреса каждому узлу сети. Стоит ли в пределах вашей организации использовать иерархическую структуру адресов? Поясните свой ответ.

- 33.6.** Назовите основное преимущество кодирования адресов Ethernet непосредственно в адреса протокола IPv6?
- 33.7.** Предположим, узел сети генерирует свой локальный относительно канала связи адрес путем объединения закодированного 48-битового адреса платы Ethernet со стандартным префиксом локального относительно канала связи адреса. Будет ли получившийся адрес уникальным в сети? Поясните свой ответ.
- 33.8.** Используя условия предыдущего упражнения, ответьте, указано ли в стандарте, что для проверки уникальности адреса узел сети должен использовать протокол обнаружения соседних узлов (NDP)? Поясните свой ответ.
- 33.9.** Если бы вас попросили выбрать размеры полей сгруппированного глобального одноадресатного адреса протокола IPv6 (см. рис. 33.9), соответствующих верхнему, среднему и нижнему уровням иерархии, как бы вы это сделали? Поясните свой ответ.
- 33.10.** Ознакомьтесь с подробной информацией о формате дополнительных заголовков аутентификации и системы безопасности протокола IPv6. Почему было предложено два заголовка?
- 33.11.** Каким образом минимальное значение MTU, равное 1280 октетам, определяет универсальность протокола IPV6.

# Приложение 1

## Справочник по документам RFC

### Введение

Большая часть информации о семействе протоколов TCP/IP и объединенным сетям, составляющим глобальную сеть Internet, включая их структуру, используемые протоколы и историю возникновения, приведена в наборе специальных документов, названных *запросами на комментарии* (*Request For Comments*, или *RFC*). Несмотря на то что RFC представляют собой набор плохо структурированных документов, написанных неформальным языком, в них, как правило, содержится бесценная (а иногда и забавная) информация, найти которую где бы то ни было еще весьма проблематично. Перед тем как рассмотреть более серьезный аспект RFC, мы уделим немного внимания именно забавной стороне. Хорошой отправной точкой можно считать стихотворение Винтона Церфа *Twas the Night Before Start-up* (*Бессонная ночь накануне пуска*), приведенное в [RFC 968]. В нем в юмористической форме описывается ряд проблем, которые возникают при развертывании новой сети. Чувство юмора поможет читателю преодолеть все проблемы. Каждый, кто помнит свои первые впечатления от участия в конференции Internet, буквально “напичканной” сетевыми жargonными словечками, и от прочтения книги Льюиса Кэрролла *Зазеркалье* (Lewis Carroll, *Jabberwocky*), насыщенной рифмованной абракадаброй, поймет почему Д.Л. Ковилл (D. L. Covill) назвал свой опус, опубликованный в [RFC 527] *ARPAWOCKY*.

Ряд других документов RFC покажутся вам такими же легкомысленными. Описание разнообразных идей, которые привели к резкому изменению сетевых технологий можно найти в документах наподобие [RFC 416], написанном в начале ноября 1972 года. Он озаглавлен *The ARC System will be Unavailable for Use During Thanksgiving Week* (*Система ARC не будет работать во время недели благодарения*). Это название говорит само за себя. А как вам ироничный юмор Криспина (Crispin), который в [RFC 748] описывает предлагаемый параметр протокола TELNET, позволяющий удалять символы случайным образом? По сути любой документ RFC, составленный первого апреля, следует рассматривать как очередную шутку. Если подобные документы кажутся вам совершенно бессмысленными, то как вам список из 75 RFC, которые *никогда не были изданы*? У каждого документа есть автор, каждому из документов назначен номер, но ни один из них так и не вышел в свет. Оставшиеся в результате этого пропуски в системе нумерации документов будут постоянно напоминать о так и нереализованных идеях или о работе, которая так и не была завершена.

Даже если удалить все шутливые, бессмысленные и бесполезные RFC, оставшиеся документы все равно не будут соответствовать принятым строгим стандартам научных работ. В отличие от заумных научных журналов, в которых обычно публикуются тщательно отобранные статьи, представляющие в основном

архивный интерес для потомков, в RFC отражены происходившие когда-то дебаты по поводу принципов разработки и создания того или иного компонента сети, результаты проведенных измерений и опыт его эксплуатации в глобальной сети Internet. Читатель вдруг начинает осознавать, что в RFC сконцентрированы идеи исследователей, занимающихся разработкой передовых технических решений, а не мысли классиков, полностью постигших предмет. Авторы документов RFC не всегда делают правильные выводы и излагают верные мысли, но они прекрасно понимают сложность изучаемых ими проблем и то, что решить их можно только совместными усилиями, проведя открытое обсуждение. Например, [RFC 1173] является документом, в котором зафиксированы “неписанные правила”, которым должны следовать сетевые администраторы (весь парадокс состоит в том, что с момента опубликования этого документа “неписанные правила” стали “писанными”).

Несмотря на противоречия, содержащиеся в документах RFC, которые иногда затрудняют их понимание, особенно для новичков, механизм документов RFC принят и сейчас выполняет свои функции очень хорошо. Поскольку документы RFC публикуются в электронном виде, информация, содержащаяся в них, быстро распространяется среди всего сообщества Internet. Поскольку в них затрагивается широкий круг вопросов, для создания документов RFC привлекаются как разработчики, так и эксплуатационники. Поскольку в RFC часто фиксируются неофициальные предложения, эти документы можно рассматривать как источник дискуссий, а не как истину в последней инстанции. Пользу могут принести даже те документы, в которых содержатся противоречивые и спорные предложения, поскольку они показывают, какие вопросы рассматривали разработчики перед принятием окончательного решения, относящегося к данному протоколу. Кроме того, те из читателей, кто интересуется историей развития какой-либо идеи или протокола, могут по соответствующим документам RFC проследить все этапы их развития от начала и до текущего состояния.

## **Значение документов, содержащих требования к узлам сети и маршрутизаторам**

В отличие от большинства документов RFC, в которых описывается какая-либо одна идея или протокол, существуют три специальных документа RFC, имеющих отношение к большому числу протоколов. Эти специальные документы озаглавлены *Requirements for Internet Routers* (*Требования к маршрутизаторам Internet*) и *Requirements for Internet Hosts* (*Требования к узлам сети Internet*). Последний документ состоит из двух частей.

Документы с требованиями опубликованы на основании обобщения многолетнего опыта использования семейства протоколов TCP/IP. В них приведены основные изменения, которые были внесены в стандарты протоколов. По существу в этих документах приводится обзор многих протоколов. В них описываются выявленные слабые места протоколов, указываются неопределенности, содержащиеся в документах RFC, определяющие стандарты протоколов, приводятся негласные правила, принятые разработчиками, и проблемы, возникающие на практике, а также возможные решения этих проблем, почерпнутые из практики. Следует учесть, что изменения, вносимые в протоколы в процессе их эволюции, не отражаются в соответствующих документах RFC, где эти протоколы были первоначально описаны. Все изменения фиксируются в документах, содержащих требования. Поэтому читатели должны быть внимательны при изучении конкретного протокола и не забывать обращаться к документам, содержащим требования.

## Магические числа RFC

Размеры документов RFC изменяются в очень широких пределах. Средний размер документа составляет 47504,5 байт. Размер самого большого документа [[RFC 1166](#)], содержащего перечень номеров Internet, составляет 566778 байт, а размер самого маленького документа составляет всего 27 байт. В нем содержится следующий текст:

This RFC was never issued.  
(Этот документ RFC не опубликован)

В процессе анализа документов RFC было обнаружено несколько интересных совпадений. Например, текстовый файл, содержащий документ [[RFC 41](#)], состоит ровно из 41 строки, а файл, содержащий [[RFC 854](#)], состоит из 854 строк. Номер документа [[RFC 1996](#)] соответствует году его опубликования. И это при том, что номера всех других документов никогда не соответствовали году их опубликования.

Количество опубликованных за год документов RFC также изменяется в течение времени в очень широких пределах (рис. П.1). Всплеск в начале 1970-х годов соответствует начальному периоду развития Internet. Кроме того, резкое увеличение количества публикаций, произошедший за 1990-е годы, является результатом начала коммерческого использования Internet.

## Как найти нужный документ RFC в Internet

Как уже упоминалось, документы RFC опубликованы в электронном виде. Их копии содержатся на многих файловых серверах, разбросанных по всему миру. Чтобы найти нужный документ RFC в Internet, обратитесь к своему сетевому администратору. Он укажет адрес ближайшего к вам сервера. Кроме того, начать поиск можно со следующего URL: <http://www.rfc-editor.org>.

## Поиск нужного документа в списке RFC

Для облегчения поиска нужного документа RFC создано несколько индексных файлов. Институт научной информации (Institute for Scientific Information, или ISI) опубликовал индексы всех RFC, упорядоченные в хронологическом порядке. Читателям часто нужно знать, в каком из документов RFC описана последняя версия официального стандарта протокола или какой из протоколов считается официальным, а какой — нет. Поэтому IAB периодически публикует документ RFC, озаглавленный *INTERNET OFFICIAL PROTOCOL STANDARDS* (*Официальные стандарты протоколов Internet*). В нем содержится список всех протоколов, которые были утверждены в качестве официальных стандартов семейства протоколов TCP/IP, а также номера последних опубликованных документов RFC, либо номера документов RFC, содержащих описание каждого протокола. Сам процесс стандартизации также описан в специальном документе [[RFC 1602](#)], озаглавленном *The Internet Standards Process — Revision 2* (*Процесс стандартизации в Internet, 2-е издание*). В этом документе даны определения таким терминам, как *предложенный стандарт* (*proposed standard*), *рабочий стандарт* (*draft standard*), *стандарт Internet* (*Internet standard*), *обязательный* (*required*), *рекомендованный* (*recommended*) и *исторический* (*historic*).

Несмотря на наличие индексов, поиск в них нужного документа RFC может быть затруднен, особенно в случае, когда читателю нужна информация по конкретной теме. Дело усугубляется еще и тем, что эта информация может находиться в разных документах RFC, опубликованных в разное время. Поиск также

частично затрудняется из-за того, что по названию документа не всегда можно определить его содержимое. Как, например, по названию *Leaving Well Enough Alone* (*Лучшее враг — хорошего*) человек может определить, что данный документ имеет отношение к стандарту FTP? И наконец, читателя может сбить с толку то, что у некоторых документов RFC совпадают названия (например, *Internet Numbers*, т.е. *Номера Internet*). Без обращения к архиву читатель не сможет определить, устарел документ или нет.

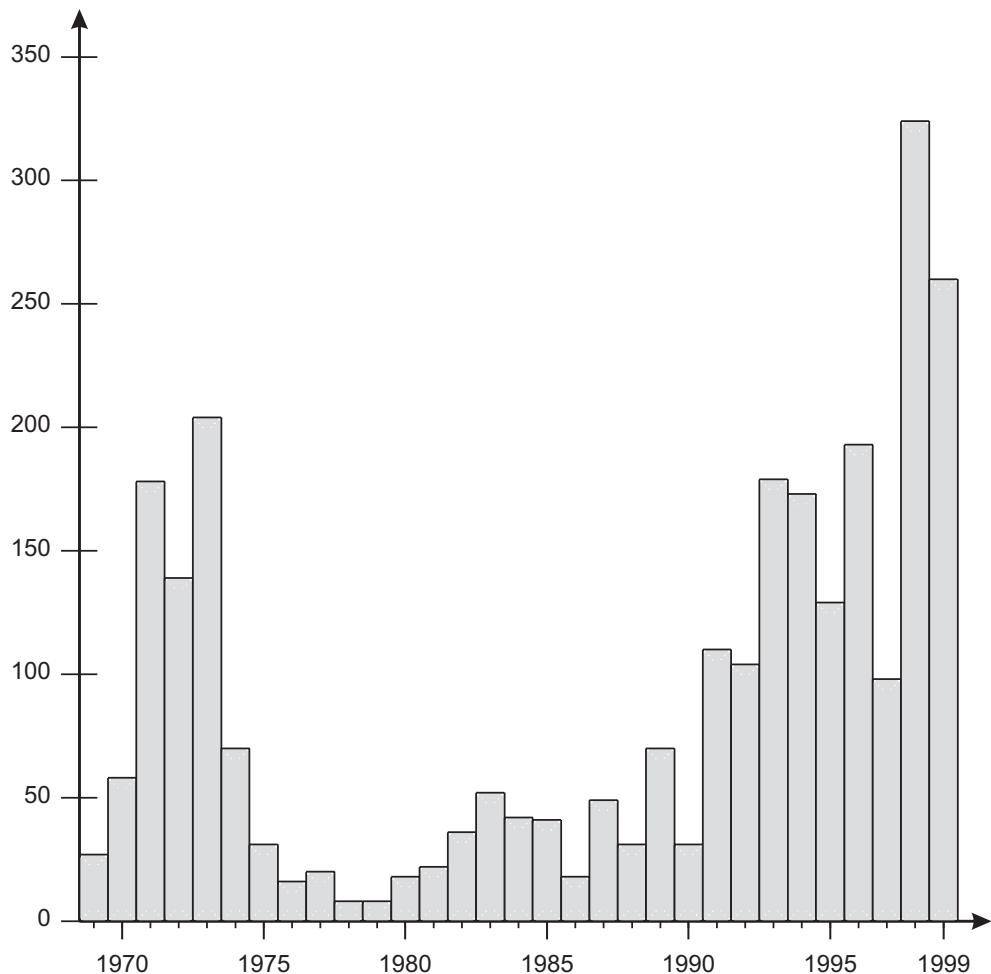


Рис. П.1. График распределения количества опубликованных документов RFC по годам

## Список RFC, упорядоченный по темам

Последний раздел этого приложения предназначен для облегчения поиска информации в документах RFC. В нем содержится список первых 2728 документов RFC, упорядоченный по темам. Читатели также могут обратиться к более раннему тематическому индексу, опубликованному в [RFC 1000], содержащему список первых 1000 документов RFC с необходимыми комментариями. Несмотря

на длину [RFC 1000] можно рекомендовать как альтернативный источник достоверной и ценной информации. Особенно восхищает введение к этому документу. Здесь нужно напомнить, что истоки документов RFC тянутся к истокам сети ARPANET. Введение наполнено духом приключений и силой, которая до сих пор является отличительной чертой Internet.

## **Документы RFC, упорядоченные по основным категориям и подтемам**

### **1. Административные**

#### **1а. Назначенные номера Internet (официальные значения, используемые в протоколах)**

1700, 1340, 1166, 1117, 1062, 1060, 1020, 1010, 997, 990, 960, 943, 923, 900, 870, 820, 790, 776, 770, 762, 758, 755, 750, 739, 717, 604, 503, 433, 349, 322, 317, 204, 179, 175, 167

#### **1б. Официальные стандарты IAB и другие списки протоколов**

2500, 2400, 2300, 2200, 2000, 1920, 1880, 1800, 1780, 1720, 1610, 1600, 1540, 1500, 1410, 1360, 1280, 1250, 1200, 1140, 1130, 1100, 1083, 1011, 991, 961, 944, 924, 901, 880, 840, 694, 661, 617, 582, 580, 552  
774, 766

#### **1с. Материалы конференций и протоколы**

- 2316 — Report of the IAB Security Architecture Workshop (Отчет о семинаре IAB, посвященном проблемам безопасности)
- 2130 — The Report of the IAB Character Set Workshop held 29 February — 1 March, 1996 (Отчет о семинаре IAB, посвященном символьным таблицам, проходившем с 29 февраля по 1 марта 1996 года)
- 1862 — Report of the IAB Workshop on Internet Information Infrastructure, October 12-14, 1994 (Отчет о семинаре IAB, посвященном информационной инфраструктуре Internet, проходившем с 12 по 14 октября 1994 года)
- 1636 — Report of IAB Workshop on Security in the Internet Architecture — February 8-10, 1994 (Отчет о семинаре IAB, посвященном проблемам безопасности в структуре Internet, проходившем с 8 по 10 февраля 1994 года)
- 1588 — White Pages Meeting Report (Отчет о конференции, посвященной белым страницам Internet)
- 1210 — Network and infrastructure user requirements for transatlantic research collaboration: Brussels, July 16-18, and Washington July 24-25, 1990 (Отчет о конференциях, посвященных требованиям пользователей к сетям и инфраструктуре для проведения совместных трансатлантических исследований, проходивших в Брюсселе с 16 по 18 июля и в Вашингтоне с 24 по 25 июля 1990 года)
- 1152 — Workshop report: Internet research steering group workshop on very-highspeed networks (Отчет о семинаре, проведенным группой управления исследованиями в Internet и посвященном сверхвысокоскоростным сетям)
- 1077 — Critical issues in high bandwidth networking (Критические заметки о сетях с высокой пропускной способностью)

- 1019 — Report of the Workshop on Environments for Computational Mathematics  
 (Отчет о семинаре, посвященном среде для вычислительной математики)
- 1017 — Network requirements for scientific research: Internet task force on scientific computing (Требования к сети для научных исследований: специальная комиссия Internet по научным расчетам)
- 910, 807 — Multimedia mail meeting notes (Материалы конференции, посвященной мультимедийной электронной почте)
- 898 — Gateway special interest group meeting notes (Материалы конференции, проводимой специальной группой по интересам и посвященной проблеме шлюзов)
- 808, 805, 469 — Summary of computer mail services meeting held at BBN on 10 January 1979 (Конспект конференции, посвященной службе компьютерной почты, проведенной корпорацией BBN 10 января 1979 года)
- 585 — ARPANET users interest working group meeting (Материалы конференции, проводимой рабочей группой заинтересованных пользователей ARPANET)
- 549, 396, 282, 253 — Minutes of Network Graphics Group meeting, 15-17 July 1973 (Протоколы конференции, проводимой группой сетевой графики с 15 по 17 июля 1973 года)
- 371 — Demonstration at International Computer Communications Conference (Отчет о демонстрации, проведенной на международной конференции по компьютерным коммуникациям)
- 327 — Data and File Transfer workshop notes (Материалы семинара, посвященного передаче данных и файлов)
- 316 — ARPA Network Data Management Working Group (Рабочая группа по управлению данными в сети ARPA)
- 164, 131, 108, 101, 82, 77, 63, 37, 21 — Minutes of Network Working Group meeting, 5/16 through 5/19/71 (Протоколы конференции сетевой рабочей группы, проведенной с 16 по 19 мая 1971 года)

#### **1d. Уведомления о конференциях и обзоры групп**

- 1160, 1120 — Internet Activities Board (Архитектурный совет Internet)
- 828 — Data communications: IFIP's international "network" of experts (Передача данных: международная "сеть" экспертов Международной федерации по обработке информации)
- 631 — International meeting on minicomputers and data communication: Call for papers (Международная конференция по миникомпьютерам и передаче данных: материалы предоставляются по требованию)
- 584 — Charter for ARPANET Users Interest Working Group (Хартия рабочей группы заинтересованных пользователей ARPANET)
- 537 — Announcement of NGG meeting July 16-17 (Уведомление о конференции NGG, проводимой с 16 по 17 июля)
- 526 — Technical meeting: Digital image processing software systems (Промышленная конференция: программные системы для обработки цифровых изображений)
- 504 — Distributed resources workshop announcement (Уведомление о семинаре, посвященном распределенным ресурсам)
- 483 — Cancellation of the resource notebook framework meeting (Отмена семинара, посвященного структуре регистрации ресурсов)
- 474, 314, 246, 232, 134 — Announcement of NGWG meeting: Call for papers (Уведомление о конференции NGWG: материалы предоставляются по требованию)

- 471 — Workshop on multi-site executive programs (Семинар, посвященный многошинным управляющим программам)
- 461 — Telnet Protocol meeting announcement (Уведомление о конференции, посвященной протоколу Telnet)
- 457 — TIPUG
- 456 — Memorandum: Date change of mail meeting (Приказ: изменение даты проведения конференции по системам электронной почты)
- 454 — File Transfer Protocol — meeting announcement and a new proposed document (Протокол передачи файлов — объявление о конференции и новые предложенные документы)
- 453 — Meeting announcement to discuss a network mail system (Объявление о конференции, посвященной проблемам сетевых почтовых систем)
- 374 — IMP System Announcement (Уведомление о системе IMP)
- 359 — Status of the Release of the New IMP System (2600) (Состояние дел по выпуску новой системы IMP 2600)
- 343, 331 — IMP System change notification (Извещение об изменении системы IMP)
- 324 — RJE Protocol meeting (Конференция по протоколу RJE)
- 323 — Formation of Network Measurement Group (NMG) (Создание сетевой измерительной группы)
- 320 — Workshop on Hard Copy Line Printers (Семинар, посвященный построчно печатающим устройствам)
- 309 — Data and File Transfer Workshop Announcement (Уведомление о семинаре, посвященном передаче данных и файлов)
- 299 — Information Management System (Система управления информацией)
- 295 — Report of the Protocol Workshop, 12 October 1971 (Отчет о семинаре, посвященном протоколам, который проходил 12 октября 1971 года)
- 291, 188, 173 — Data Management Meeting Announcement (Уведомление о конференции, посвященной управлению данными)
- 245, 234, 207, 140, 116, 99, 87, 85, 75, 43, 35 — Reservations for Network Group meeting (Бронь для конференции сетевой группы)
- 222 — Subject: System programmer's workshop (Тематика: Семинар системных программистов)
- 212 — NWG meeting on network usage (Конференция компании NWG по использованию сети)
- 157 — Invitation to the Second Symposium on Problems in the Optimization of Data Communications Systems (Приглашение на второй симпозиум посвященный проблемам оптимизации систем передачи данных)
- 149 — Best Laid Plans (Планы на будущее)
- 130 — Response to RFC 111: Pressure from the chairman (Ответ на RFC 111: давление со стороны председателя)
- 111 — Pressure from the Chairman (Давление со стороны председателя)
- 48 — Possible protocol plateau (Возможная платформа для протоколов)
- 46 — ARPA Network protocol notes (Примечания по поводу протоколов а сети ARPA)

#### **1e. Списки рассылок**

- 402, 363, 329, 303, 300, 211, 168, 155 — ARPA Network Mailing Lists (Списки рассылки сети ARPA)
- 69 — Distribution List Change for MIT (Изменения списка рассылки для MIT)
- 52 — Updated distribution list (Обновленный список рассылки)

## **1f. Нормативные документы**

- 2717 — Registration Procedures for URL Scheme Names (Методики регистрации для структуры имен URL)
- 2506 — Media Feature Tag Registration Procedure (Методика регистрации дескриптора описания возможностей среды)
- 2489 — Procedure for Defining New DHCP Options (Методика определения новых параметров протокола DHCP)
- 2418, 1603 — IETF Working Group Guidelines and Procedures (Руководящие принципы и методы рабочей группы IETF)
- 2282, 2027 — IAB and IESG Selection, Confirmation, and Recall Process: Operation of the Nominating and Recall Committees (Процедуры отбора, подтверждения и отзыва, принятые IAB и IESG: деятельность комитетов по выдвижению и отзыву)
- 2278 — IANA Charset Registration Procedures (Процедуры регистрации набора символов в IANA)
- 2277 — IETF Policy on Character Sets and Languages (Политика IETF по отношению к наборам символов и языкам)
- 2146, 1816, 1811 — US Government Internet Domain Names (Доменные имена Internet, принадлежащие государственным организациям США)
- 2135 — Internet Society By-Laws (Устав сообщества Internet)
- 2050 — Internet Registry IP Allocation Guidelines (Руководящие принципы деятельности Реестра Internet по выделению IP-адресов)
- 2042 — Registering New BGP Attribute Types (Регистрация новых типов атрибутов протокола BGP)
- 2014 — IRTF Research Group Guidelines and Procedures (Руководящие принципы и методы исследовательской группы IRTF)
- 1956 — Registration in the MIL Domain (Порядок регистрации имен в домене MIL)
- 1930 — Guidelines for creation, selection, and registration of an Autonomous System (AS) (Правила создания, отбора и регистрации автономных систем)
- 1875 — UNINETT PCA Policy Statements (Перечень правил приемлемого использования UNINETT PCA)
- 1371 — Choosing a Common IGP for the IP Internet (Выбор общего протокола внутреннего шлюза для объединенной сети на основе протокола IP)
- 1124 — Policy issues in interconnecting networks (Правила использования объединенных сетей)
- 1087 — Ethics and the Internet (Этика и Internet)
- 1052 — IAB recommendations for the development of Internet network management standards (Рекомендации IAB по разработке стандартов сетевого управления)
- 1039 — DoD statement on Open Systems Interconnection protocols (Заявление МО США по поводу протоколов эталонной модели взаимодействия открытых систем)
- 980 — Protocol document order information (Информация об упорядочении документов протоколов)
- 952, 810, 608 — DoD Internet host table specification (Спецификация таблицы узлов Internet МО США)
- 945 — DoD statement on the NRC report (Заявление МО США по поводу отчета научно-исследовательского совета)
- 902 — ARPA Internet Protocol policy (Правила использования протоколов Internet в сети ARPA)
- 849 — Suggestions for improved host table distribution (Предложения по улучшению распространения таблицы узлов)
- 678 — Standard file formats (Стандартные форматы файлов)

- 602 — “The stockings were hung by the chimney with care” (Дословный перевод:  
“Аккуратно вешайте чулки над трубой”)
- 115 — Some Network Information Center policies on handling documents (Некоторые правила Сетевого информационного центра по обработке документов)
- 53 — Official protocol mechanism (Механизм создания официальных протоколов)

#### **1g. Административные документы**

- 2648 — A URN Namespace for IETF Documents (Пространство унифицированных имен ресурсов для документов IETF)
- 2629 — Writing I-Ds and RFCs using XML (Написание документации и RFC на языке XML)
- 2499, 2399, 2299, 2199, 2099, 1999, 1899, 1799, 1699, 1599, 1499, 1399, 1299, 999, 899, 800, 699, 598, 200, 170, 160, 100, 84 — Request for Comments Summary (Сводный список документов RFC)
- 2434 — Guidelines for Writing an IANA Considerations Section in RFCs (Правила написания раздела документов RFC, в котором излагается мнение IANA)
- 2360 — Guide for Internet Standards Writers (Руководство для составителей стандартов Internet)
- 2223, 1543, 1111 — Instructions to RFC Authors (Инструкции авторам RFC)
- 2119 — Key words for use in RFCs to Indicate Requirement Levels (Ключевые слова, которые должны использоваться в документах RFC для отображения уровня требований)
- 1818 — Best Current Practices (Лучшие современные методики)
- 1796 — Not All RFCs are Standards (Не все документы RFC являются стандартами)
- 1311 — Introduction to the STD Notes (Предисловие к стандартным комментариям)
- 1150 — FYI on FYI: Introduction to the FYI Notes (Предисловие к информационным комментариям)
- 1000 — Request For Comments reference guide (Справочное руководство по документам RFC)
- 825 — Request for comments on Requests For Comments (Запрос на комментарии по документам RFC)
- 629 — Scenario for using the Network Journal (План использования сетевого журнала)
- 628 — Status of RFC numbers and a note on pre-assigned journal numbers (Состояние номеров документов RFC и комментарии по поводу заранее распределенных номеров журналов)

#### **1h. Другие**

- 2691 — A Memorandum of Understanding for an ICANN Protocol Support Organization (Меморандум о согласии для организации поддержки протокола при ICANN)
- 2690 — A Proposal for an MOU-Based ICANN Protocol Support Organization (Предложение, основанное на меморандуме о согласии для организации поддержки протокола при ICANN)
- 2436 — Collaboration between ISOC/IETF and ITU-T (Сотрудничество между организациями ISOC/IETF и ITU-T)
- 2339, 1790 — An Agreement Between the Internet Society, the IETF, and Sun Microsystems, Inc. (Соглашение между сообществом Internet, IETF и фирмой Sun Microsystems, Inc.)
- 2134 — Articles of Incorporation of Internet Society (Договор об объединении сообщества Internet)
- 2053 — The AM (Armenia) Domain (Домен AM (Армения))

- 2031 — IETF-ISOC relationship (Взаимодействие между IETF и сообществом Internet)
- 2028 — The Organizations Involved in the IETF Standards Process (Список организаций, вовлеченных в процесс стандартизации группы IETF)
- 2026, 1871, 1602, 1310 — The Internet Standards Process — Revision 3 (Процесс стандартизации в Internet, 3-я редакция)
- 1988 — Conditional Grant of Rights to Specific Hewlett-Packard Patents In Conjunction With the Internet Engineering Task Force's Internet-Standard Network Management Framework (Субсидия, предусматривающая соблюдение получателем определенных прав на некоторые патенты фирмы Hewlett-Packard, во взаимодействии со структурой сетевого управления, созданной рабочей группой IETF)
- 1984 — IAB and IESG Statement on Cryptographic Technology and the Internet (Заявление IAB и исполнительного комитета IETF по поводу использования криптографических технологий в Internet)
- 1917 — An Appeal to the Internet Community to Return Unused IP Networks (Prefixes) to the IANA (Обращение к сообществу Internet с просьбой возвращать неиспользуемые IP-адреса сетей (префиксов) в IANA)
- 1822 — A Grant of Rights to Use a Specific IBM patent with Photuris (Перечень прав на использование некоторых патентов фирмы IBM совместно с Photuris)
- 1718, 1539, 1391 — The Tao of IETF — A Guide for New Attendees of the Internet Engineering Task Force (Святая святых IETF — руководство для новых членов этой организации)
- 1690 — Introducing the Internet Engineering and Planning Group (IEPG) (Общие сведения об Группе разработки и планирования Internet)
- 1689 — A Status Report on Networked Information Retrieval: Tools and Groups (Данные о состоянии сетевой системы выборки информации: средства и группы)
- 1640 — The Process for Organization of Internet Standards Working Group (POISED) (Предписание для рабочей группы организации по стандартизации в Internet)
- 1601, 1358 — Charter of the Internet Architecture Board (IAB) (Хартия Архитектурного совета Internet)
- 1527 — What Should We Plan Given the Dilemma of the Network? (Что планируется предпринять для выхода из затруднительного положения в Сети?)
- 1481 — IAB Recommendation for an Intermediate Strategy to Address the Issue of Scaling (Рекомендация IAB по поводу промежуточной политики в отношении адресации при росте сети)
- 1401 — Correspondence between the IAB and DISA on the use of DNS (Переписка между IAB и DISA по поводу использования DNS)
- 1396 — The Process for Organization of Internet Standards Working Group (POISED) (Предписание для рабочей группы организации по стандартизации в Internet)
- 1380 — IESG Deliberations on Routing and Addressing (Рассуждения IESG по поводу маршрутизации и адресации)
- 1297 — NOC Internal Integrated Trouble Ticket System Functional Specification Wish list ("NOC TT REQUIREMENTS") (Функциональные требования ко внутренней интегрированной системе аварийных сертификатов сетевого операционного центра (Требования NOC TT))
- 1287 — Towards the Future Internet Architecture (Еще раз о будущем структуры Internet)

- 1272 — Internet Accounting: Background (Учет использования ресурсов в Internet: предпосылки)
- 1261 — Transition of Nic Services (Развитие служб Nic)
- 1174 — IAB recommended policy on distributing internet identifier assignment and IAB recommended policy change to internet “connected” status (Рекомендованные IAB правила распространения назначений идентификаторов объединенной сети и рекомендованные IAB правила изменены на состояние “подключения” к объединенной сети)
- 637 — Change of network address for SU-DSL (Изменение сетевого адреса для SU-DSL)
- 634 — Change in network address for Haskins Lab (Изменение сетевого адреса для Haskins Lab)
- 616 — Latest network maps (Свежие карты сети)
- 609 — Statement of upcoming move of NIC/NLS service (Заявление по поводу стремительного развития служб NIC/NLS)
- 590 — MULTICS address change (Изменение адресов системы MULTICS)
- 588 — London node is now up (Запущен в строй Лондонский узел)
- 551 — NYU, ANL, and LBL Joining the Net (К Internet подключены NYU, ANL и LBL)
- 544 — Locating on-line documentation at SRI-ARC (Поиск электронной документации на SRI-ARC)
- 543 — Network journal submission and delivery (О подчинении и распространении сетевого журнала)
- 518 — ARPANET accounts (Бюджет ARPANET)
- 511 — Enterprise phone service to NIC from ARPANET sites (Корпоративная телефонная служба, связывающая узлы ARPANET с NIC)
- 510 — Request for network mailbox addresses (Запрос на сетевые адреса почтовых ящиков)
- 440 — Scheduled network software maintenance (Плановое обслуживание сетевого программного обеспечения)
- 432 — Network logical map (Сетевая логическая карта)
- 423, 389 — UCLA Campus Computing Network Liaison Staff for ARPANET (Штат служащих по связям с ARPANET сети Калифорнийского университета в Лос-Анджелесе)
- 421 — Software Consulting Service for Network Users (Программные консультационные службы для пользователей сети)
- 419 — To: Network liaisons and station agents (Сетевым посредникам и станционным агентам)
- 416 — ARC System Will Be Unavailable for Use During Thanksgiving Week (Система ARC не будет работать во время недели, на которую приходится День Благодарения)
- 405 — Correction to RFC 404 (Изменения в RFC 404)
- 404 — Host Address Changes Involving Rand and ISI (Изменение адресов узлов включая Rand и ISI)
- 403 — Desirability of a network 1108 service (Желательность службы сети 1108)
- 386 — Letter to TIP users-2 (Второе письмо к пользователям TIP)
- 384 — Official site ident for organizations in the ARPA Network (Официальные сайты для представления организаций в сети ARPA)
- 381 — Three aids to improved network operation (Три рекомендации для улучшения работы сети)
- 365 — Letter to All TIP Users (Письмо ко всем пользователям TIP)
- 356 — ARPA Network Control Center (Центр управления сетью ARPA)
- 334 — Network Use on May 8 (О пользовании сетью 8 мая)

- 305 — Unknown Host Numbers (Неизвестные номера узлов)  
301 — BBN IMP (#5) and NCC Schedule March 4, 1971 (Место проведения и расписание конференции BBN и NCC на 4 Марта 1971 года)  
289 — What we hope is an official list of host names (Свершилось: официальный список имен узлов)  
276 — NIC course (Курс NIC)  
249 — Coordination of equipment and supplies purchase (Координация использования оборудования и закупки принадлежностей)  
223 — Network Information Center schedule for network users (Расписание работы сетевого информационного центра для пользователей сети)  
185 — NIC distribution of manuals and handbooks (Распространение справочников и учебников через NIC)  
154 — Exposition Style (Выставочный стиль)  
136 — Host accounting and administrative procedures (Учет использования ресурсов узлами сети и методы администрирования)  
118 — Recommendations for facility documentation (Рекомендации по написанию понятной документации)  
95 — Distribution of NWG/RFC's through the NIC (Распространение документов RFC от NWG через NIC)  
16 — M.I.T

## **2. Документы с требованиями и пересмотры основных протоколов**

### **2a. Требования к узлам сети**

- 1127 — Perspective on the Host Requirements RFCs (Перспективы документов RFC, посвященных требованиям к узлам сети)  
1123 — Requirements for Internet hosts — application and support (Требованиям к узлам сети: приложения и поддержка)  
1122 — Requirements for Internet hosts — communication layers (Требованиям к узлам сети: уровни взаимодействия)

### **2b. Требования к шлюзам**

- 2644 — Changing the Default for Directed Broadcasts in Routers (Изменение стандартного пути для направленных широковещательных дейтаграмм в маршрутизаторах)  
1812, 1009 — Requirements for IP Version 4 Routers (Требования к маршрутизаторам протокола IP, версии 4)

## **3. Уровень сетевого интерфейса (см. также раздел 8)**

### **3a. Привязка адресов (протоколы ARP и RARP)**

- 2390, 1293 — Inverse Address Resolution Protocol (Инверсный протокол обратного преобразования адресов)  
1931 — Dynamic RARP Extensions for Automatic Network Address Acquisition (Расширения динамического протокола RARP для автоматического получения сетевых адресов)  
1868 — ARP Extension — UNARP (Расширение протокола ARP — UNARP)  
1433 — Directed ARP (Направленный ARP)  
1329 — Thoughts on Address Resolution for Dual MAC FDDI Networks (Соображения по поводу преобразования адресов для дублированных сетей MAC FDDI)  
1027 — Using ARP to implement transparent subnet gateways (Использование протокола ARP для реализации прозрачных шлюзов подсетей)

- 925 — Multi-LAN address resolution (Преобразование адресов в многосегментной локальной сети)
- 903 — Reverse Address Resolution Protocol (Протокол обратного преобразования адресов)
- 826 — Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48 bit Ethernet address for transmission on Ethernet hardware (Протокол преобразования адресов в сети Ethernet: преобразование сетевого протокольного адреса в 48-битовый физический адрес Ethernet для последующей передачи по сети Ethernet)

### **3b. Использование протокола IP в других сетях (инкапсуляция)**

- 2728 — The Transmission of IP Over the Vertical Blanking Interval of a Television Signal (Передача IP-трафика во время импульса обратного хода кадровой развертки телевизионного сигнала)
- 2625 — IP and ARP over Fibre Channel (Протоколы IP и ARP в оптоволоконных сетях)
- 2176 — IPv4 over MAPOS Version 1 (Использование протокола IPv4 на основе протокола множественного доступа версии 1 в сетях SONET)
- 2143 — Encapsulating IP with the Small Computer System Interface (Инкапсуляция IP-дейтаграмм для передачи по протоколу системного интерфейса малых компьютеров (SCSI))
- 2067, 1374 — IP over HIPPI (Использование протокола IP на основе высокоскоростного параллельного интерфейса)
- 2004, 2003, 1853 — Minimal Encapsulation within IP (Минимально необходимая инкапсуляция в рамках протокола IP)
- 1390, 1188, 1103 — Transmission of IP and ARP over FDDI Networks (Передача дейтаграмм и протокол ARP в сетях FDDI)
- 1241 — Scheme for an internet encapsulation protocol: Version 1 (Проект протокола инкапсуляции объединенной сети, версия 1)
- 1226 — Internet protocol encapsulation of AX.25 frames (Инкапсуляция IP-дейтаграмм в фреймы AX.25)
- 1221, 907 — Host Access Protocol (HAP) specification: Version 2 (Спецификация протокола доступа к узлам сети (HAP), версия 2)
- 1209 — Transmission of IP datagrams over the SMDS Service (Передача IP-дейтаграмм через службу высокоскоростной коммутации данных)
- 1201, 1051 — Transmitting IP traffic over ARCnet networks (Передача IP-трафика по сетям ARCnet)
- 1088 — Standard for the transmission of IP datagrams over NetBIOS networks (Стандарт для передачи IP-дейтаграмм по протоколу NetBIOS)
- 1055 — Nonstandard for transmission of IP datagrams over serial lines: SLIP (Передача IP-дейтаграмм по последовательным каналам связи: протокол SLIP (неофициальный документ))
- 1044 — Internet Protocol on Network System's HYPERchannel: Protocol specification (Протокол IP в сетевых системах HYPERchannel: спецификация протокола)
- 1042 — Standard for the transmission of IP datagrams over IEEE 802 networks (Стандарт передачи IP-дейтаграмм по сетям IEEE 802)
- 948 — Two methods for the transmission of IP datagrams over IEEE 802.3 networks (Два способа передачи IP-дейтаграмм по сетям IEEE 802.3)
- 895 — Standard for the transmission of IP datagrams over experimental Ethernet networks (Стандарт передачи IP-дейтаграмм по экспериментальным сетям Ethernet)

- 894 — Standard for the transmission of IP datagrams over Ethernet networks  
(Стандарт передачи IP-дейтаграмм по сетям Ethernet)
- 893 — Trailer encapsulations (Инкапсуляция трейлера)
- 877 — Standard for the transmission of IP datagrams over public data networks  
(Стандарт передачи IP-дейтаграмм по открытым сетям передачи данных)

### **3с. Сети с множественным доступом, не поддерживающие режим широковещания (ATM, IP-коммутация, MPLS)**

- 2702 — Requirements for Traffic Engineering Over MPLS (Требования для передачи трафика через MPLS)
- 2684 — MultiProtocol Encapsulation over ATM Adaptation Layer 5 (Многопротокольная инкапсуляция для протокола адаптации ATM уровня 5)
- 2682 — Performance Issues in VC-Merge Capable ATM LSRs (Вопросы производительности коммутирующего маршрутизатора меток в виртуальных каналах ATM, поддающихся слиянию)
- 2643 — Cabletron's SecureFast VLAN Operational Model (Действующая модель виртуальной сети SecureFast фирмы Cabletron)
- 2642 — Cabletron's VLS Protocol Specification (Спецификация протокола VLS фирмы Cabletron)
- 2641 — Cabletron's VlanHello Protocol Specification Version 4 (Спецификация протокола VlanHello версии 4 фирмы Cabletron)
- 2603 — ILMI-Based Server Discovery for NHRP (Обнаружение серверов на основе интерфейса интегрированного локального управления для протокола определения ближайшего адресата)
- 2602 — ILMI-Based Server Discovery for MARS (Обнаружение серверов на основе интерфейса интегрированного локального управления для MARS)
- 2601 — ILMI-Based Server Discovery for ATMARP (Обнаружение серверов на основе интерфейса интегрированного локального управления для протокола ATMARP)
- 2583 — Guidelines for Next Hop Client (NHC) Developers (Общие правила для разработчиков клиентов протокола определения ближайшего адресата)
- 2520 — NHRP with Mobile NHCs (Протокол определения ближайшего адресата в мобильном протоколе NHC)
- 2443 — A Distributed MARS Service Using SCSP (Распределенная служба MARS на основе использования протокола синхронизации кэш-памяти сервера)
- 2383 — ST2+ over ATM Protocol Specification — UNI 3.1 Version (Спецификация протокола ST2+ в сети ATM: версия интерфейса “пользователь-сеть” 3.1)
- 2340 — Nortel's Virtual Network Switching (VNS) Overview (Обзор виртуальной сетевой коммутации фирмы Nortel)
- 2337 — Intra-LIS IP multicast among routers over ATM using Sparse Mode PIM (Реализация многоадресатной передачи IP-пакетов по внутренней логической IP-подсети в сетях ATM с использованием разреженного режима PIM)
- 2336 — Classical IP to NHRP Transition (Классический переход с IP к NHRP)
- 2335 — A Distributed NHRP Service Using SCSP (Реализация распределенной службы NHRP на основе SCSP)
- 2334 — Server Cache Synchronization Protocol (SCSP) (Протокол синхронизации кэш-памяти сервера)
- 2333 — NHRP Protocol Applicability Statement (Заявление по поводу применимости протокола NHRP)
- 2332 — NBMA Next Hop Resolution Protocol (NHRP) (Протокол определения ближайшего адресата в нешироковещательных сетях с множественным доступом)

- 2331 — ATM Signaling Support for IP over ATM — UNI Signaling 4.0 Update (Поддержка режима сигнализации ATM для протокола IP в сети ATM: обновление версии 4.0 механизма сигнализации “пользователь-сеть”)
- 2297, 1987 — Ipsilon's General Switch Management Protocol Specification Version 2.0 (Спецификация протокола общего управления коммутаторами версии 2.0 фирмы Ipsilon)
- 2269 — Using the MARS Model in non-ATM NBMA Networks (Использование модели MARS в нешироковещательных не ATM-сетях с множественным доступом)
- 2226 — IP Broadcast over ATM Networks (Широковещательный режим передачи IP-дейтаграмм в сетях ATM)
- 2225, 1577 — Classical IP and ARP over ATM (Реализация классических протоколов IP и ARP в сетях ATM)
- 2191 — VENUS — Very Extensive Non-Unicast Service (VENUS — чрезвычайно протяженная неодноадресатная служба)
- 2170 — Application REQuested IP over ATM (AREQUIPA) (Установка сквозных IP-соединений между машинами в сети ATM)
- 2149 — Multicast Server Architectures for MARS-based ATM multicasting (Структура многоадресатного сервера для системы многоадресатной рассылки в сети ATM на основе MARS)
- 2129 — Toshiba's Flow Attribute Notification Protocol (FANP) Specification (Спецификация протокола извещения об атрибуте потока фирмы Toshiba)
- 2124 — Cabletron's Light-weight Flow Admission Protocol Specification Version 1.0 (Спецификация упрощенного протокола доступа к потоку версии 1.0 фирмы Cabletron)
- 2121 — Issues affecting MARS Cluster Size (Факторы, влияющие на размер кластера службы MARS)
- 2105 — Cisco Systems' Tag Switching Architecture Overview (Обзор структуры системы коммутации тэгов фирмы Cisco)
- 2098 — Toshiba's Router Architecture Extensions for ATM: Overview (Обзор расширений структуры маршрутизатора для сети ATM)
- 2022 — Support for Multicast over UNI 3.0/3.1 based ATM Networks (Поддержка режима многоадресатного вещания в сетях ATM на основе UNI 3.0/3.1)
- 1954 — Transmission of Flow Labeled IPv4 on ATM Data Links Ipsilon Version 1.0 (Передача потока, помеченного как IPv4, по каналам передачи данных ATM фирмы Ipsilon, версия 1.0)
- 1953 — Ipsilon Flow Management Protocol Specification for IPv4 Version 1.0 (Спецификация протокола управления потоком для IPv4 версии 1.0 фирмы Ipsilon)
- 1932 — IP over ATM: A Framework Document (Протокол IP в сети ATM: базовый документ)
- 1755 — ATM Signaling Support for IP over ATM (Поддержка режима сигнализации ATM для протокола IP в сети ATM)
- 1754 — IP over ATM Working Group's Recommendations for the ATM Forum's MultiProtocol BOF Version 1 (Рекомендация рабочей группы по поддержке протокола IP в сети ATM версии 1, сделанная на неформальном заседании, посвященном проблеме поддержки многих протоколов)
- 1735 — NBMA Address Resolution Protocol (NARP) (Протокол преобразования адресов в нешироковещательных сетях с множественным доступом)
- 1626 — Default IP MTU for use over ATM AAL5 (Стандартное значение MTU для протокола IP для использования с протоколом адаптации ATM уровня 5)
- 1483 — MultiProtocol Encapsulation over ATM Adaptation Layer 5 (Многопротокольная инкапсуляция в протоколе адаптации ATM уровня 5)

### **3d. Другие**

- 2469 — A Caution On The Canonical Ordering Of Link-Layer Addresses (Проблемы при каноническом упорядочении адресов канального уровня)  
2427, 1490, 1294 — MultiProtocol Interconnect over Frame Relay (Многопротокольное взаимодействие в сети Frame Relay)  
2341 — Cisco Layer Two Forwarding (Protocol) “L2F” (Протокол пересылки пакетов второго уровня фирмы Cisco)  
2175 — MAPOS 16 — Multiple Access Protocol over SONET/SDH with 16 Bit Addressing (Протокол множественного доступа в сетях SONET/SDH с 16-битовой адресацией)  
2174 — A MAPOS version 1 Extension — Switch-Switch Protocol (Расширение протокола MAPOS версии 1: протокол “коммутатор-коммутатор”)  
2173 — A MAPOS version 1 Extension — Node Switch Protocol (Расширение протокола MAPOS версии 1 — протокол узла коммутации)  
2172 — MAPOS Version 1 Assigned Numbers (Выделенные номера для протокола MAPOS версии 1)  
2171 — MAPOS — Multiple Access Protocol over SONET/SDH Version 1 (Протокол множественного доступа в сетях SONET/SDH версии 1)  
1326 — Mutual Encapsulation Considered Dangerous (Опасности взаимной инкапсуляции)

## **4. Уровень Internet**

### **4a. Протокол IP**

- 2113 — IP Router Alert Option (Параметр, запрашивающий внимание маршрутизатора, протокола IP)  
1624, 1141 — Computation of the Internet Checksum via Incremental Update (Вычисление контрольной суммы IP-заголовка при инкрементальном обновлении)  
1191, 1063 — Path MTU Discovery (Поиск значения MTU по пути следования пакетов)  
1071 — Computing the Internet checksum (Вычисление контрольной суммы IP-заголовка)  
1025 — TCP and IP bake off (Конкурс на лучшую реализацию протоколов TCP и IP)  
815 — IP datagram reassembly algorithms (Алгоритмы сборки IP-дейтаграмм)  
791, 760 — Internet Protocol (Протокол Internet)  
781 — Specification of the Internet Protocol (IP) timestamp option (Спецификация параметра, содержащего временную метку протокола IP)

### **4b. Протокол межсетевых управляемых сообщений (ICMP)**

- 2521 — ICMP Security Failures Messages (ICMP-сообщения об ошибках протокола IPsec)  
1788 — ICMP Domain Name Messages (ICMP-сообщения для системы доменных имен)  
1256 — ICMP Router Discovery Messages (ICMP-сообщения для системы обнаружения маршрутизаторов)  
1018 — Some comments on SQuID (Некоторые комментарии по поводу задержки, вносимой механизмом подавления источника)  
1016 — Something a host could do with source quench: The Source Quench Introduced Delay (SQuID) (Как узел сети может использовать механизм подавления источника: задержка, вносимая механизмом подавления источника)

792, 777 — Internet Control Message Protocol (Протокол межсетевых управляющих сообщений)

#### **4c. Многоадресатная рассылка (IGMP)**

- 2588 — IP Multicast and Firewalls (Многоадресатная рассылка в протоколе IP и брандмауэры)
- 2502 — Limitations of Internet Protocol Suite for Distributed Simulation the Large Multicast Environment (Ограничения семейства протоколов Internet для распределенной эмуляции большой среды многоадресатной передачи)
- 2365 — Administratively Scoped IP Multicast (Административно управляемое пространство многоадресатной передачи протокола IP)
- 2357 — IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols (Критерий IETF для оценки надежной транспортной службы для многоадресатной рассылки и протоколов прикладного уровня)
- 2236 — Internet Group Management Protocol, Version 2 (Межсетевой протокол управления группами, версия 2)
- 1768 — Host Group Extensions for CLNP Multicasting (Расширения ведущей группы для режима многоадресатной передачи в сетевых протоколах без установки соединения)
- 1469 — IP Multicast over Token-Ring Local Area Networks (Режим многоадресатной передачи протокола IP в локальных сетях Token-Ring)
- 1458 — Requirements for Multicast Protocols (Требования к протоколам многоадресатной передачи)
- 1301 — Multicast Transport Protocol (Транспортный протокол многоадресатной передачи)
- 1112, 1054, 988, 966 — Host extensions for IP multicasting (Расширения для узлов сети для многоадресатной передачи протокола IP)

#### **4d. Маршрутизация и алгоритмы шлюза (BGP, GGP, RIP, OSPF)**

- 2715 — Interoperability Rules for Multicast Routing Protocols (Правила взаимодействия для протоколов многоадресатной маршрутизации)
- 2676 — QoS Routing Mechanisms and OSPF Extensions (Механизмы маршрутизации с поддержкой качества обслуживания и расширения протокола OSPF)
- 2650 — Using RPSL in Practice (Практическое использование языка RPSL)
- 2622, 2280 — Routing Policy Specification Language (RPSL) (Спецификация языка описания маршрутной политики)
- 2519 — A Framework for Inter-Domain Route Aggregation (Структура для объединения междоменных маршрутов)
- 2453, 1723, 1388 — RIP Version 2 (Протокол RIP, версия 2)
- 2439 — BGP Route Flap Damping (Демпфирование самопроизвольно изменяющихся маршрутов протокола BGP)
- 2385 — Protection of BGP Sessions via the TCP MD5 Signature Option (Защита BGP-сессий с помощью параметра протокола TCP, содержащего сигнатуру MD5)
- 2370 — The OSPF Opaque LSA Option (Трудный для понимания параметр LSA протокола OSPF)
- 2362, 2117 — Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification (Спецификация не зависящего от протокола разреженного режима многоадресатной передачи PIM-SM)
- 2338 — Virtual Router Redundancy Protocol (Протокол отказоустойчивого виртуального маршрутизатора)
- 2329 — OSPF Standardization Report (Отчет по стандартизации протокола OSPF)
- 2328, 2178, 1583, 1247, 1131 — OSPF Version 2 (Протокол OSPF версии 2)

- 2283 — MultiProtocol Extensions for BGP-4 (Многопротокольные расширения для BGP-4)
- 2281 — Cisco Hot Standby Router Protocol (HSRP) (Протокол резервного маршрутизатора компании Cisco)
- 2270 — Using a Dedicated AS for Sites Homed to a Single Provider (Использование выделенных автономных систем для сетевых центров, подключенных к одному провайдеру)
- 2260 — Scalable Support for Multi-homed Multi-provider Connectivity (Поддержка расширяемости для сетей, имеющих несколько физических соединений, ведущих к разным провайдерам)
- 2201, 2189 — Core Based Trees (CBT) Multicast Routing Architecture (Протокол распределенного связующего дерева для систем многоадресатной маршрутизации)
- 2154 — OSPF with Digital Signatures (Протокол OSPF с цифровой подписью)
- 2103 — Mobility Support for Nimrod: Challenges and Solution Approaches (Поддержка мобильности для систем Nimrod: проблемы и методы их решения)
- 2102 — Multicast Support for Nimrod : Requirements and Solution Approaches (Поддержка многоадресатной передачи для систем Nimrod: требования и методы их решения)
- 2092 — Protocol Analysis for Triggered RIP (Анализ протоколов для ждущего RIP)
- 2091 — Triggered Extensions to RIP to Support Demand Circuits (Ждущие расширения протокола RIP для соединений, устанавливаемых по требованию)
- 2082 — RIP-2 MD5 Authentication (Аутентификация MD5 в протоколе RIP-2)
- 2009 — GPS-Based Addressing and Routing (Адресация и маршрутизация на основе GPS)
- 1998 — An Application of the BGP Community Attribute in Multi-home Routing (Применение атрибута сообщества BGP в процессе многоадресной маршрутизации)
- 1997 — BGP Communities Attribute (Атрибут сообщества BGP)
- 1992 — The Nimrod Routing Architecture (Система маршрутизации Nimrod)
- 1966 — BGP Route Reflection An alternative to full mesh IBGP (Отражение маршрутов BGP: альтернатива полносеточной IBGP)
- 1965 — Autonomous System Confederations for BGP (Конфедерация автономных систем для BGP)
- 1923 — RIPv1 Applicability Statement for Historic Status (Заявление по поводу применимости протокола RIPv1 для восстановления исторической справедливости)
- 1863 — A BGP/IDRP Route Server alternative to a full mesh routing (Сервер маршрутизации BGP/IDRP — альтернатива полносеточной маршрутизации)
- 1817 — CIDR and Classful Routing (CIDR и классовая маршрутизация)
- 1793 — Extending OSPF to Support Demand Circuits (Расширение протокола OSPF для поддержки соединений, устанавливаемых по требованию)
- 1787 — Routing in a Multi-provider Internet (Маршрутизация в объединенной сети, связанной с несколькими провайдерами)
- 1786 — Representation of IP Routing Policies in a Routing Registry (ripe-81++) (Представление правил IP-маршрутизации в реестре маршрутизации (ripe-81++))
- 1774 — BGP-4 Protocol Analysis (Анализ протокола BGP-4)
- 1773, 1656 — Experience with the BGP-4 protocol (Опыт использования протокола BGP-4)
- 1772, 1655, 1268, 1164 — Application of the Border Gateway Protocol in the Internet (Применение протокола BGP в Internet)

- 1771, 1654, 1267, 1163 — A Border Gateway Protocol 4 (BGP-4) (Протокол граничного шлюза версии 4 (BGP-4))
- 1765 — OSPF Database Overflow (Переполнение базы данных OSPF)
- 1745 — BGP4/IDRP for IP — OSPF Interaction (Протоколы BGP4/IDRP для выполнения IP-маршрутизации: взаимодействие с протоколом OSPF)
- 1722 — RIP Version 2 Protocol Applicability Statement (Заявление по поводу применимости протокола RIP версии 2)
- 1721, 1387 — RIP Version 2 Protocol Analysis (Анализ протокола RIP версии 2)
- 1702, 1701 — Generic Routing Encapsulation over IPv4 networks (Общая инкапсуляция для маршрутизации в сетях IPv4)
- 1587 — The OSPF NSSA Option (Параметр NSSA протокола OSPF)
- 1586 — Guidelines for Running OSPF Over Frame Relay Networks (Инструкции по запуску протокола OSPF в сетях Frame Relay)
- 1585 — MOSPF: Analysis and Experience (Протокол MOSPF: анализ и опыт использования)
- 1584 — Multicast Extensions to OSPF (Многоадресатные расширения протокола OSPF)
- 1582 — Extensions to RIP to Support Demand Circuits (Расширения протокола RIP для поддержки соединений, устанавливаемых по требованию)
- 1581 — Protocol Analysis for Extensions to RIP to Support Demand Circuits (Анализ протоколов для расширений RIP для поддержки соединений, устанавливаемых по требованию)
- 1520 — Exchanging Routing Information Across Provider Boundaries in the CIDR Environment (Обмен маршрутной информацией между границами провайдеров в CIDR-окружении)
- 1519, 1338 — Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy (Бесклассовая междоменная маршрутизация (CIDR): назначение адресов и стратегия объединения)
- 1517 — Applicability Statement for the Implementation of Classless Inter-Domain Routing (CIDR) (Заявление по поводу применимости бесклассовой междоменной маршрутизации (CIDR))
- 1504 — AppleTalk Update-Based Routing Protocol: Enhanced AppleTalk Routing (Протокол маршрутизации на основе обновлений AppleTalk: расширение протокола маршрутизации AppleTalk)
- 1482 — Aggregation Support in the NSFNET Policy-Based Routing Database (Поддержка объединений в базе данных маршрутизации на основе правил NSFNET)
- 1479 — Inter-Domain Policy Routing Protocol Specification: Version 1 (Спецификация протокола междоменной маршрутизации с помощью правил (IDPR))
- 1478 — An Architecture for Inter-Domain Policy Routing (Структура системы междоменной маршрутизации с помощью правил)
- 1477 — IDPR as a Proposed Standard (Предложенный стандарт протокола междоменной маршрутизации с помощью правил)
- 1465 — Routing Coordination for X.400 MHS Services Within a Multi Protocol /Multi Network Environment. Table Format V3 for Static Routing (Согласование маршрутизации для служб MHS X.400, работающих в много-протокольной и многосетевой среде. Формат таблицы для статической маршрутизации версии 3)
- 1403, 1364 — BGP OSPF Interaction (Взаимодействие протоколов BGP и OSPF)
- 1397 — Default Route Advertisement In BGP2 and BGP3 Version of The Border Gateway Protocol (Анонсирование стандартного маршрута в версиях 2 и 3 протокола граничного шлюза)

- 1383 — An Experiment in DNS Based IP Routing (Опыт проведения IP-маршрутизации на основе DNS)
- 1370 — Applicability Statement for OSPF (Заявление по поводу применимости протокола OSPF)
- 1322 — A Unified Approach to Inter-Domain Routing (Унифицированный подход к проблеме междоменной маршрутизации)
- 1266 — Experience with the BGP Protocol (Опыт использования протокола BGP)
- 1265 — BGP Protocol Analysis (Анализ протокола BGP)
- 1264 — Internet Engineering Task Force Internet Routing Protocol Standardization Criteria (Критерий IETF стандартизации протокола маршрутизации Internet)
- 1254 — Gateway Congestion Control Survey (Перегрузка шлюза управляющими сообщениями)
- 1246 — Experience with the OSPF Protocol (Опыт использования протокола OSPF)
- 1245 — OSPF Protocol Analysis (Анализ протокола OSPF)
- 1222 — Advancing the NSFNET routing architecture (Усовершенствованная система маршрутизации NSFNET)
- 1195 — Use of OSI IS-IS for routing in TCP/IP and dual environments (Использование протокола OSI IS-IS для маршрутизации в среде протокола TCP/IP и сдвоенных средах)
- 1142 — OSI IS-IS Intra-domain Routing Protocol (Протокол междоменной маршрутизации OSI IS-IS)
- 1136 — Administrative Domains and Routing Domains: A model for routing in the Internet (Административные домены и домены маршрутизации: модель системы маршрутизации в Internet)
- 1133 — Routing between the NSFNET and the DDN (Маршрутизация между NSFNET и DDN)
- 1126 — Goals and functional requirements for inter-autonomous system routing (Задачи и функциональные требования для маршрутизации между автономными системами)
- 1125 — Policy requirements for inter Administrative Domain routing (Система требований и правил для маршрутизации между административными доменами)
- 1105 — Border Gateway Protocol (BGP) (Протокол граничного шлюза (BGP))
- 1104 — Models of policy based routing (Модели маршрутизации на основе правил)
- 1102 — Policy routing in Internet protocols (Маршрутизация на основе правил в протоколах Internet)
- 1093 — NSFNET routing architecture (Структура системы маршрутизации в NSFNET)
- 1092 — EGP and policy based routing in the new NSFNET backbone (Протокол внешней маршрутизации и маршрутизация на основе правил в новой магистральной сети NSFNET)
- 1075 — Distance Vector Multicast Routing Protocol (Дистанционно-векторный протокол многоадресатной маршрутизации)
- 1074 — NSFNET backbone SPF based Interior Gateway Protocol (Протокол внутреннего шлюза на основе метода поиска кратчайшего пути магистральной сети NSFNET)
- 1058 — Routing Information Protocol (Протокол маршрутной информации)
- 1046 — Queuing algorithm to provide type-of-service for IP links (Алгоритм на основе очередей, обеспечивающий заданный тип обслуживания для IP-каналов передачи данных)
- 985 — Requirements for Internet gateways — draft (Рабочий вариант требований для шлюзов Internet)

- 975 — Autonomous confederations (Автономные конфедерации)
- 970 — On packet switches with infinite storage (Коммутаторы пакетов с неограниченным объемом памяти)
- 911 — EGP Gateway under Berkeley UNIX 4.2 (Шлюз EGP в Berkeley UNIX 4.2)
- 904, 890, 888, 827 — Exterior Gateway Protocol formal specification (Формализованное описание протокола внешнего шлюза)
- 875 — Gateways, architectures, and heffalumps (Шлюзы, структура и слонопотамы)
- 823 — DARPA Internet gateway (Шлюз Internet DARPA)

#### **4е. Протокол IP следующего поколения (IPng, IPv6)**

- 2711 — IPv6 Router Alert Option (Параметр, запрашивающий внимание маршрутизатора, протокола IPv6)
- 2710 — Multicast Listener Discovery (MLD) for IPv6 (Обнаружение приемника многоадресатных пакетов для протокола IPv6)
- 2675, 2147 — IPv6 Jumbograms (“Слонограммы” протокола IPv6)
- 2590 — Transmission of IPv6 Packets over Frame Relay (Передача пакетов протокола IPv6 через сеть Frame Relay)
- 2553, 2133 — Basic Socket Interface Extensions for IPv6 (Расширения основного интерфейса сокетов для протокола IPv6)
- 2546 — 6Bone Routing Practice (Правила маршрутизации в сетевой магистрали на основе протокола IPv6 (6Bone))
- 2545 — Use of BGP-4 MultiProtocol Extensions for IPv6 Inter-Domain Routing (Использование многопротокольных расширений BGP-4 для выполнения междоменной маршрутизации протокола IPv6)
- 2529 — Transmission of IPv6 over IPv4 Domains without Explicit Tunnels (Передача трафика протокола IPv6 по доменам IPv4 без явного туннелирования)
- 2526 — Reserved IPv6 Subnet Anycast Addresses (Зарезервированные альтернативные адреса подсетей протокола IPv6)
- 2497 — Transmission of IPv6 Packets over ARCnet Networks (Передача пакетов IPv6 по сетям ARCnet)
- 2492 — IPv6 over ATM Networks (Передача трафика IPv6 по сетям ATM)
- 2491 — IPv6 over Non-Broadcast Multiple Access (NBMA) networks (Передача трафика IPv6 по нешироковещательным сетям с множественным доступом)
- 2473 — Generic Packet Tunneling in IPv6 Specification (Спецификация универсального туннелирования пакетов протокола IPv6)
- 2472, 2023 — IP Version 6 over PPP (Использование протокола IPv6 при PPP-соединении)
- 2471, 1897 — IPv6 Testing Address Allocation (Тестирование распределения адресов протокола IPv6)
- 2470 — Transmission of IPv6 Packets over Token Ring Networks (Передача пакетов IPv6 по сетям Token Ring)
- 2467, 2019 — Transmission of IPv6 Packets over FDDI Networks (Передача пакетов IPv6 по сетям FDDI)
- 2466, 2465 — Management Information Base for IP Version 6: ICMPv6 Group (База управляющей информации для протокола IPv6: группа ICMPv6)
- 2464, 1972 — Transmission of IPv6 Packets over Ethernet Networks (Передача пакетов IPv6 по сетям Ethernet)
- 2463, 1885 — Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification (Спецификация протокола межсетевых управляющих сообщений (ICMPv6) для протокола IPv6)
- 2462, 1971 — IPv6 Stateless Address Autoconfiguration (Автоконфигурирование адреса в протоколе IPv6 без учета состояния)

- 2461, 1970 — Neighbor Discovery for IP Version 6 (IPv6) (Обнаружение соседних устройств в протоколе IPv6)
- 2460, 1883 — Internet Protocol, Version 6 (IPv6) Specification (Спецификация протокола IPv6)
- 2454 — IP Version 6 Management Information Base for the User Datagram Protocol (База управляющей информации протокола IPv6 для протокола передачи пользовательских дейтаграмм (UDP))
- 2452 — IP Version 6 Management Information Base for the Transmission Control Protocol (База управляющей информации протокола IPv6 для протокола управления передачей (TCP))
- 2450 — Proposed TLA and NLA Assignment Rule (Предложенные правила назначения идентификаторов верхнего и следующего уровней агрегирования)
- 2375 — IPv6 Multicast Address Assignments (Распределение многоадресатных адресов протокола IPv6)
- 2374, 2073 — An IPv6 Aggregatable Global Unicast Address Format (Формат сгруппированного глобального одноадресатного адреса)
- 2373, 1884 — IP Version 6 Addressing Architecture (Структура адресации протокола IPv6)
- 2292 — Advanced Sockets API for IPv6 (Усовершенствованный API сокетов для протокола IPv6)
- 2185 — Routing Aspects of IPv6 Transition (Особенности маршрутизации при переходе на протокол IPv6)
- 2081 — RIPng Protocol Applicability Statement (Заявление по поводу применимости протокола RIPng)
- 2080 — RIPng for IPv6 (Протокол RIPng для IPv6)
- 1981 — Path MTU Discovery for IP version 6 (Поиск значения MTU по пути следования пакетов для протокола IPv6)
- 1955 — New Scheme for Internet Routing and Addressing (ENCAPS) for IPng (Новый проект маршрутизации и адресации в Internet (ENCAPS) для IPng)
- 1933 — Transition Mechanisms for IPv6 Hosts and Routers (Механизмы перехода на протокол IPv6 для узлов сети и маршрутизаторов)
- 1888 — OSI NSAPs and IPv6 (Точки доступа к сетевым службам OSI и протокол IPv6)
- 1887 — An Architecture for IPv6 Unicast Address Allocation (Структура распределения одноадресатных адресов протокола IPv6)
- 1886 — DNS Extensions to support IP version 6 (Расширения DNS для поддержки протокола IPv6)
- 1881 — IPv6 Address Allocation Management (Управление распределением адресов протокола IPv6)
- 1809 — Using the Flow Label Field in IPv6 (Использование поля, содержащего метку потока в протоколе IPv6)
- 1753 — IPng Technical Requirements Of the Nimrod Routing and Addressing Architecture (Технические требования протокола IPng для маршрутизации Nimrod и структуры адресации)
- 1752 — The Recommendation for the IP Next Generation Protocol (Рекомендации для протокола IP следующего поколения)
- 1726 — Technical Criteria for Choosing IP The Next Generation (IPng) (Технические критерии выбора протокола IP следующего поколения(IPng))
- 1719 — A Direction for IPng (Инструкция по IPng)
- 1710 — Simple Internet Protocol Plus White Paper (Официальный документ, содержащий описание простого протокола Internet плюс (SIPP))
- 1707 — CATNIP: Common Architecture for the Internet (Общая структура для протокола IP следующего поколения)

- 1705 — Six Virtual Inches to the Left: The Problem with IPng (Небольшое отступление от темы: проблемы, связанные с протоколом IPng)
- 1688 — IPng Mobility Considerations (Заметки по поводу мобильности протокола IPng)
- 1687 — A Large Corporate User's View of IPng (Точка зрения пользователей большой корпорации на протокол IPng)
- 1686 — IPng Requirements: A Cable Television Industry Viewpoint (Требования к протоколу IPng с точки зрения индустрии кабельного телевидения)
- 1683 — MultiProtocol Interoperability In IPng (Многопротокольное взаимодействие в IPng)
- 1682 — IPng BSD Host Implementation Analysis (Анализ реализации протокола IPng на узлах сети под управлением BSD Unix)
- 1680 — IPng Support for ATM Services (Поддержка служб ATM в протоколе IPng)
- 1679 — HPN Working Group Input to the IPng Requirements Solicitation (Вклад рабочей группы по разработке высокопроизводительных сетей (HPN) в разработку вводных требований к протоколу IPng)
- 1678 — IPng Requirements of Large Corporate Networks (Требования больших корпоративных сетей к протоколу IPng)
- 1677 — Tactical Radio Frequency Communication Requirements for IPng (Требования тактической радиосвязи к протоколу IPng)
- 1676 — INFN Requirements for an IPng (Требования сети INFN к протоколу IPng)
- 1675 — Security Concerns for IPng (Вопросы безопасности протокола IPng)
- 1674 — A Cellular Industry View of IPng (Мнение представителей индустрии мобильной связи по поводу протокола IPng)
- 1673 — Electric Power Research Institute Comments on IPng (Комментарии представителей института исследования электрической энергии (EPRI) по поводу протокола IPng)
- 1672 — Accounting Requirements for IPng (Требования к учету использования ресурсов для протокола IPng)
- 1671 — IPng White Paper on Transition and Other Considerations (Официальный документ по поводу перехода к протоколу IPng и других проблем)
- 1670 — Input to IPng Engineering Considerations (Заметки по поводу вклада в разработку IPng)
- 1669 — Market Viability as a IPng Criteria (Жизнеспособность на рынке как критерий IPng)
- 1668 — Unified Routing Requirements for IPng (Унифицированные требования к маршрутизации для протокола IPng)
- 1667 — Modeling and Simulation Requirements for IPng (Требования по моделированию и эмуляции протокола IPng)
- 1622 — Pip Header Processing (Обработка заголовка протокола PIP)
- 1621 — Pip Near-term Architecture (Структура перспективного протокола PIP)
- 1550 — IP: Next Generation (IPng) White Paper Solicitation (Протокол IP следующего поколения: ходатайство к официальному документу)
- 1526 — Assignment of System Identifiers for TUBA/CLNP Hosts (Назначение системных идентификатора для узлов TUBA/CLNP)
- 1475 — TP/IX: The Next Internet (TP/IX: Internet ближайшего будущего)
- 1454 — Comparison of Proposals for Next Version of IP (Сравнительные характеристики предложенных новых версий протокола IP)
- 1385 — EIP: The Extended Internet Protocol (EIP: расширенный протокол Internet)
- 1375 — Suggestion for New Classes of IP Addresses (Предложение по созданию новых классов IP-адресации)
- 1365 — An IP Address Extension Proposal (Предложение по расширению IP-адресации)

- 1347 — TCP and UDP with Bigger Addresses (TUBA), A Simple Proposal for Internet Addressing and Routing (Протоколы TCP и UDP, поддерживающие большое адресное пространство, несложное предложение для IP-адресации и маршрутизации)
- 1335 — A Two-Tier Address Structure for the Internet: A Solution to the Problem of Address Space Exhaustion (Двухуровневая структура адресации для Internet: решение проблемы исчерпания адресного пространства)

#### **4f. Выделение IP-адресов и сетевых номеров**

- 2391 — Load Sharing using IP Network Address Translation (LSNAT) (Распределение нагрузки с помощью трансляции сетевых IP-адресов)
- 2101 — IPv4 Address Behavior Today (Современное состояние адресного пространства протокола IPv4)
- 2072 — Router Renumbering Guide (Руководство по перенумерации маршрутизатора)
- 2071 — Network Renumbering Overview: Why would I want it and what is it anyway? (Перенумерация сети: что это такое и зачем она нужна?)
- 2036 — Observations on the use of Components of the Class A Address Space within the Internet (Замечания по использованию компонентов адресного пространства класса А в Internet)
- 2008 — Implications of Various Address Allocation Policies for Internet Routing (Влияние различных стратегий распределения адресов на процесс маршрутизации в Internet)
- 1918, 1597 — Address Allocation for Private Internets (Распределение адресов для частных объединенных сетей)
- 1916 — Enterprise Renumbering: Experience and Information Solicitation (Перенумерация корпоративных сетей: опыт и сбор информации)
- 1900 — Renumbering Needs Work (Перенумерация должна работать)
- 1879, 1797 — Class A Subnet Experiment Results and Recommendations (Результаты экспериментов с подсетями класса А и рекомендации по их использованию)
- 1878, 1860 — Variable Length Subnet Table For IPv4 (Таблица подсетей переменной длины для протокола IPv4)
- 1814 — Unique Addresses are Good (Уникальные адреса — это хорошо)
- 1744 — Observations on the Management of the Internet Address Space (Замечания по управлению адресным пространством Internet)
- 1715 — The H Ratio for Address Assignment Efficiency (Коэффициент Н для эффективного распределения адресов)
- 1681 — On Many Addresses per Host (По поводу назначения нескольких адресов одному узлу)
- 1627 — Network 10 Considered Harmful (Some Practices Shouldn't be Codified) (Десять считающихся вредными сетевых заповедей (Ряд методик, которые не должны использоваться))
- 1466, 1366 — Guidelines for Management of IP Address Space (Инструкции по управлению адресным пространством протокола IP)
- 1219 — On the assignment of subnet numbers (По поводу назначения номеров подсетям)
- 950 — Internet Standard Subnetting Procedure (Стандартная процедура разбиения на подсети в Internet)
- 940, 936, 932, 917 — Toward an Internet standard scheme for subnetting (По поводу будущего стандартного проекта Internet разбиения на подсети)

#### **4g. Изоляция сети (VPN, брандмауэры, NAT)**

- 2694 — DNS extensions to Network Address Translators (DNS\_ALG) (Расширения системы DNS для трансляции сетевых адресов (DNS\_ALG))
- 2685 — Virtual Private Networks Identifier (Идентификатор виртуальных частных сетей)
- 2663 — IP Network Address Translator (NAT) Terminology and Considerations (Преобразование сетевых адресов (NAT): терминология и анализ)
- 2647 — Benchmarking Terminology for Firewall Performance (Терминология, посвященная производительности брандмауэров)
- 2637 — Point-to-Point Tunneling Protocol (Протокол туннелирования “точка-точка”)
- 2547 — BGP/MPLS VPNs (Создание виртуальных частных сетей на основе протоколов BGP/MPLS)
- 1961 — GSS-API Authentication Method for SOCKS Version 5 (Метод аутентификации GSS-API для SOCKS версии 5)
- 1929, 1928 — Username/Password Authentication for SOCKS V5 (Аутентификация по имени пользователя и паролю для SOCKS версии 5)
- 1858 — Security Considerations for IP Fragment Filtering (Анализ безопасности для фильтрации IP-фрагментов)
- 1631 — The IP Network Address Translator (NAT) (Преобразование сетевых IP-адресов (NAT))

#### **4h. Остальное**

- 2698 — A Two Rate Three Color Marker (Двухскоростной трехцветный маркер)
- 2697 — A Single Rate Three Color Marker (Односкоростной трехцветный маркер)
- 2638 — A Two-bit Differentiated Services Architecture for the Internet (Структура двухбитового поля класса дифференцированного обслуживания IP-пакета)
- 2598 — An Expedited Forwarding PHB (per hop forwarding behavior) (Быстрая пересылка по методу от узла к узлу)
- 2597 — Assured Forwarding PHB Group (Гарантиированная пересылка по методу от узла к узлу)
- 2508 — Compressing IP/UDP/RTP Headers for Low-Speed Serial Links (Сжатие заголовков протоколов IP/UDP/RTP для передачи по низкоскоростным последовательным каналам связи)
- 2507 — IP Header Compression (Сжатие заголовка протокола IP)
- 2481 — A Proposal to add Explicit Congestion Notification (ECN) to IP (Предложение по добавлению в протокол IP явного уведомления о перегрузке)
- 2475 — An Architecture for Differentiated Service (Структура системы дифференцированного обслуживания)
- 2474, 1349 — Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers (Описание поля класса дифференцированного обслуживания в заголовках протоколов IPv4 и IPv6)
- 2395 — IP Payload Compression Using LZS (Сжатие содержимого IP-пакета по методу LZS)
- 2394 — IP Payload Compression Using DEFLATE (Сжатие содержимого IP-пакета по методу DEFLATE)
- 2393 — IP Payload Compression Protocol (IPComp) (Протокол сжатия содержимого IP-пакета)
- 2075 — IP Echo Host Service (Служба эха узла сети протокола IP)
- 1946 — Native ATM Support for ST2+ (Естественная поддержка ATM для ST2+)

- 1940 — Source Demand Routing: Packet Format and Forwarding Specification (Version 1) (Маршрутизация по запросу от источника: формат пакетов и правила перенаправления)
- 1937 — “Local/Remote” Forwarding Decision in Switched Data Link Subnetworks (Методы пересылки пакетов между локальным и удаленным узлом в подсетях с коммутируемыми каналами передачи данных)
- 1936 — Implementing the Internet Checksum in Hardware (Аппаратная реализация устройства вычисления контрольной суммы протокола IP)
- 1919 — Classical versus Transparent IP Proxies (Сравнение классического и “прозрачного” типов proxy-серверов протокола IP)
- 1819, 1190 — Internet Stream Protocol Version 2 (ST2) Protocol Specification — Version ST2+ (Спецификация протокола потоковой передачи данных в Internet версии 2 — версия ST2+)
- 1770 — IPv4 Option for Sender Directed Multi-Destination Delivery (Параметр протокола IPv4 для управляемой отправителем доставки пакетов многим получателям)
- 1716 — Towards Requirements for IP Routers (Еще раз о требованиях к IP-маршрутизаторам)
- 1620 — Internet Architecture Extensions for Shared Media (Расширения структуры для сетей с разделяемой пропускной способностью)
- 1560 — The MultiProtocol Internet (Многопротокольный Internet)
- 1518 — An Architecture for IP Address Allocation with CIDR (Структура системы распределения IP адресов при использовании CIDR)
- 1476 — RAP: Internet Route Access Protocol (Протокол доступа к маршрутам в Internet)
- 1467, 1367 — Status of CIDR Deployment in the Internet (Заявление по поводу развертывания CIDR в Internet)
- 1393 — Traceroute Using an IP Option (Трассировка маршрутов с помощью параметров протокола IP)
- 1363 — A Proposed Flow Specification (Описание предлагаемого процесса обработки данных)
- 986 — Guidelines for the use of Internet-IP addresses in the ISO Connectionless-Mode Network Protocol (Инструкции по использованию IP-адресов в сетевом протоколе ISO, не требующем установки соединения между получателями)
- 981 — Experimental multiple-path routing algorithm (Экспериментальный алгоритм маршрутизации по нескольким маршрутам)
- 963 — Some problems with the specification of the Military Standard Internet Protocol (Некоторые проблемы, связанные со спецификацией стандарта IP-протокола для военных целей)
- 947 — Multi-network broadcasting within the Internet (Широковещательный режим передачи в нескольких сетях, подключенных к Internet)
- 922, 919 — Broadcasting Internet datagrams in the presence of subnets (Широковещательный режим передачи IP-дейтаграмм при наличии подсетей)
- 871 — Perspective on the ARPANET reference model (Перспективы эталонной модели ARPANET)
- 831 — Backup access to the European side of SATNET (Резервный канал доступа к Европейской части сети SATNET)
- 817 — Modularity and efficiency in protocol implementation (Модульность и эффективность при реализации протоколов)
- 816 — Fault isolation and recovery (Локализация ошибок и восстановление работоспособности сети)

- 814 — Name, addresses, ports, and routes (Имя, адреса, порты и маршруты)
- 796 — Address mappings (Привязка адресов)
- 795 — Service mappings (Привязка служб)
- 730 — Extensible field addressing (Открытая адресация полей)

## 5. Уровень узла сети

### 5a. Протокол передачи пользовательских дейтаграмм (UDP)

- 768 — Протокол передачи пользовательских дейтаграмм

### 5b. Протокол управления передачей (TCP)

- 2582 — The NewReno Modification to TCP's Fast Recovery Algorithm (Модификация NewReno алгоритма быстрого восстановления протокола TCP)
- 2581, 2001 — TCP Congestion Control (Управление перегрузкой в протоколе TCP)
- 2525 — Known TCP Implementation Problems (Известные проблемы реализации протокола TCP)
- 2488 — Enhancing TCP Over Satellite Channels using Standard Mechanisms (Расширение протокола TCP для спутниковых каналов связи использующих стандартные механизмы)
- 2416 — When TCP Starts Up With Four Packets Into Only Three Buffers (Режим запуска протокола TCP: прием четырех пакетов при наличии только трех буферов)
- 2415 — Simulation Studies of Increased Initial TCP Window Size (Моделирование последствий увеличения начального размера окна протокола TCP)
- 2414 — Increasing TCP's Initial Window (Увеличение начального размера окна протокола TCP)
- 2398 — Some Testing Tools for TCP Implementers (Некоторые средства тестирования для тех, кто занимается реализацией протокола TCP)
- 2140 — TCP Control Block Interdependence (Взаимосвязь управляющих блоков TCP)
- 2018 — TCP Selective Acknowledgement Options (Параметр протокола TCP, отвечающий за избирательное подтверждение приема)
- 1693 — An Extension to TCP: Partial Order Service (Расширение протокола TCP: служба упорядочивания частей)
- 1644 — T/TCP — TCP Extensions for Transactions Functional Specification (T/TCP — расширения протокола TCP для функциональных требований к транзакциям)
- 1379 — Extending TCP for Transactions — Concepts (Расширение протокола TCP для поддержки транзакций — общие представления)
- 1337 — TIME-WAIT Assassination Hazards in TCP (Опасность отмены состояния TIME-WAIT в протоколе TCP)
- 1323, 1185 — TCP Extensions for High Performance (Расширения протокола TCP для повышения производительности)
- 1263 — TCP Extensions Considered Harmful (Считающиеся вредными расширения протокола TCP)
- 1146, 1145 — TCP alternate checksum options (Параметры протокола TCP, относящиеся к альтернативным методам вычисления контрольной суммы)
- 1144 — Compressing TCP/IP headers for low-speed serial links (Сжатие заголовков протокола TCP/IP для передачи по низкоскоростным последовательным каналам связи)
- 1110 — Problem with the TCP big window option (Проблемы, связанные с большим размером окна протокола TCP)
- 1106 — TCP big window and NAK options (Большой размер окна протокола TCP и отрицательное подтверждение приема)

- 1078 — TCP port service Multiplexer (TCPMUX) (Мультиплексор служб портов протокола TCP)
- 1072 — TCP extensions for long-delay paths (Расширения протокола TCP для маршрутов с большой задержкой)
- 964 — Some problems with the specification of the Military Standard Transmission Control Protocol (Некоторые проблемы, связанные со спецификацией протокола TCP для применений в военных целях)
- 962 — TCP-4 prime (Начальные сведения о протоколе TCP-4)
- 896 — Congestion control in IP/TCP internetworks (Управление перегрузкой в объединенных сетях на основе протокола IP/TCP)
- 889 — Internet delay experiments (Опыты с измерением задержки в Internet)
- 879 — TCP maximum segment size and related topics (Максимальный размер сегмента протокола TCP и связанные с этим темы)
- 872 — TCP-on-a-LAN (Протокол TCP в локальной сети)
- 813 — Window and Acknowledgement Strategy in TCP (Окно и стратегия подтверждения приема в протоколе TCP)
- 794 — Pre-emption (Упреждение)
- 793, 761, 675 — Transmission Control Protocol (Протокол управления передачей)
- 721 — Out-of-Band Control Signals in a Host-to-Host Protocol (Внеполосные управляющие сигналы в протоколе типа “узел-узел”)
- 700 — Protocol experiment (Эксперимент с протоколом)

### **5с. Протоколы типа “точка-точка” (PPP)**

- 2701 — Nortel Networks Multi-link Multi-node PPP Bundle Discovery Protocol (Многоканальный многоузловой протокол обнаружения связок PPP сетей Nortel)
- 2687 — PPP in a Real-time Oriented HDLC-like Framing (Протокол PPP, ориентированный на фреймирование в реальном масштабе времени наподобие протокола HDLC)
- 2686 — The Multi-Class Extension to Multi-Link PPP (Расширение многоканального протокола PPP для поддержки нескольких классов)
- 2615, 1619 — PPP over SONET/SDH (Протокол PPP в сетях SONET/SDH)
- 2516 — Method for Transmitting PPP Over Ethernet (PPPoE) (Метод для передачи данных протокола PPP по сети Ethernet)
- 2509 — IP Header Compression over PPP (Сжатие заголовков протокола IP при передаче по протоколу PPP)
- 2484 — PPP LCP Internationalization Configuration Option (Параметр настройки локализации протокола PPP LCP)
- 2433 — Microsoft PPP CHAP Extensions (Расширения CHAP протокола PPP фирмы Microsoft)
- 2420 — The PPP Triple-DES Encryption Protocol (3DESE) (Протокол шифрования по алгоритму тройного DES протокола PPP)
- 2419, 1969 — The PPP DES Encryption Protocol, Version 2 (DESE-bis) (Протокол шифрования по алгоритму DES протокола PPP, версия 2)
- 2364 — PPP Over AAL5 (Применение протокола PPP при передаче данных через протокол адаптации ATM уровня 5)
- 2363 — PPP Over FUNI (Применение протокола PPP при передаче данных через интерфейс “пользователь-сеть” с кадрированием)
- 2284 — PPP Extensible Authentication Protocol (EAP) (Расширяемый протокол аутентификации PPP)
- 2153 — PPP Vendor Extensions (Расширения производителей для протокола PPP)
- 2125 — The PPP Bandwidth Allocation Protocol (BAP) / The PPP Bandwidth Allocation Control Protocol (BACP) (Протокол выделения полосы пропуска-

- ния PPP/Протокол управления выделением полосы пропускания протокола PPP)
- 2118 — Microsoft Point-To-Point Compression (MPPC) Protocol (Протокол сжатия “точка-точка” фирмы Microsoft)
- 2097 — The PPP NetBIOS Frames Control Protocol (NBFCP) (Протокол управления фреймами NetBIOS протокола PPP)
- 2043 — The PPP SNA Control Protocol (SNACP) (Протокол управления системной сетевой структурой протокола PPP)
- 1994, 1334 — PPP Challenge Handshake Authentication Protocol (CHAP) (Протокол аутентификации с предварительным согласованием вызова протокола PPP)
- 1993 — PPP Gandalf FZA Compression Protocol (Протокол сжатия данных по методу Gandalf FZA протокола PPP)
- 1990, 1717 — The PPP Multilink Protocol (MP) (Многоканальный протокол протокола PPP)
- 1989, 1333 — PPP Link Quality Monitoring (Контроль качества канала передачи данных протокола PPP)
- 1979 — PPP Deflate Protocol (Протокол сжатия данных по методу Deflate протокола PPP)
- 1978 — PPP Predictor Compression Protocol (Протокол упреждающего сжатия протокола PPP)
- 1977 — PPP BSD Compression Protocol (Протокол сжатия данных по методу BSD протокола PPP)
- 1976 — PPP for Data Compression in Data Circuit-Terminating Equipment (DCE) (Применение протокола PPP для сжатия данных в оконечном оборудовании канала передачи данных)
- 1975 — PPP Magnalink Variable Resource Compression (Сжатие данных по методу переменного ресурса Magnalink протокола PPP)
- 1974 — PPP Stac LZS Compression Protocol (Протокол сжатия данных по методу Stac LZS протокола PPP)
- 1973 — PPP in Frame Relay (Применение протокола PPP в сетях Frame Relay)
- 1968 — The PPP Encryption Control Protocol (ECP) (Протокол управления шифрованием протокола PPP)
- 1967 — PPP LZS-DCP Compression Protocol (LZS-DCP) (Протокол сжатия данных по методу LZS-DCP протокола PPP)
- 1963 — PPP Serial Data Transport Protocol (SDTP) (Транспортный протокол последовательной передачи данных протокола PPP)
- 1962 — The PPP Compression Control Protocol (CCP) (Протокол управления сжатием протокола PPP)
- 1934 — Ascend’s Multilink Protocol Plus (MP+) (Многоканальный протокол+ фирмы Ascend)
- 1915 — Variance for The PPP Connection Control Protocol and The PPP Encryption Control Protocol (Различия между протоколами управления соединением и шифрованием протокола PPP)
- 1877 — PPP Internet Protocol Control Protocol Extensions for Name Server Addresses (Расширения для адресов серверов имен, предназначенное для управления передачей IP-дейтаграмм по протоколу PPP)
- 1841 — PPP Network Control Protocol for LAN Extension (Протокол управления сетью для сегментов локальной сети протокола PPP)
- 1764 — The PPP XNS IDP Control Protocol (XNSCP) (Протокол управления протоколом передачи межсетевых дейтаграмм сетевых систем фирмы Xerox для протокола PPP)

- 1763 — The PPP Banyan Vines Control Protocol (BVCP) (Протокол управления протоколом Banyan Vines для протокола PPP)
- 1762, 1376 — The PPP DECnet Phase IV Control Protocol (DNCP) (Протокол управления 4-й фазой протокола маршрутизации DECnet для протокола PPP)
- 1663 — PPP Reliable Transmission (Надежная передача дейтаграмм по протоколу PPP)
- 1662, 1549 — PPP in HDLC-like Framing (Применение протокола PPP при фреймировании типа HDLC)
- 1661, 1548 — The Point-to-Point Protocol (PPP) (Протокол “точка-точка”)
- 1638, 1220 — PPP Bridging Control Protocol (BCP) (Протокол управления мостом для протокола PPP)
- 1618 — PPP over ISDN (Применение протокола PPP при передаче данных через сеть ISDN)
- 1598 — PPP in X.25 (Протокол PPP в сети X.25)
- 1570 — PPP LCP Extensions (Расширения протокола управления каналом связи для протокола PPP)
- 1552 — The PPP Internetworking Packet Exchange Control Protocol (IPXCP) (Протокол управления межсетевым обменом пакетов (IPX) для протокола PPP)
- 1547 — Requirements for an Internet Standard Point-to-Point Protocol (Требования к стандарту Internet для протокола PPP)
- 1378 — The PPP AppleTalk Control Protocol (ATCP) (Протокол управления протоколом AppleTalk для протокола PPP)
- 1377 — The PPP OSI Network Layer Control Protocol (OSINLCP) (Протокол управления сетевым уровнем OSI для протокола PPP)
- 1332, 1172 — The PPP Internet Protocol Control Protocol (IPCP) (Протокол управления протоколом Internet для протокола PPP)
- 1331, 1171, 1134 — The Point-to-Point Protocol (PPP) for the Transmission of Multi-protocol Datagrams over Point-to-Point Links (Использование протокола PPP для передачи многопротокольных дейтаграмм через двухточечные соединения)

#### **5e. Протоколы управления транзакциями и распределенные операционные системы**

- 2372 — Transaction Internet Protocol — Requirements and Supplemental Information (Транзакционный протокол Internet: требования и дополнительная информация)
- 2371 — Transaction Internet Protocol Version 3.0 (Транзакционный протокол Internet, версия 3)
- 955 — Towards a transport service for transaction processing applications (Еще раз о транспортной службе для приложений, выполняющих обработку транзакций)
- 938 — Internet Reliable Transaction Protocol functional and interface specification (Функциональное описание и спецификация интерфейса надежного транзакционного протокола Internet)
- 722 — Thoughts on Interactions in Distributed Services (Размышления по поводу взаимодействия между распределенными службами)
- 713 — MSDTP — Message Services Data Transmission Protocol (Протокол передачи данных служб сообщений)
- 712 — Distributed Capability Computing System (DCCS) (Распределенные вычислительные комплексы)
- 708 — Elements of a Distributed Programming System (Элементы распределенных систем программирования)

- 707 — High-level framework for network-based resource sharing (Высокоуровневая структура для совместного использования ресурсов на сетевом уровне)
- 684 — Commentary on procedure calling as a network protocol (Пояснение к процедуре вызова сетевых протоколов)
- 677 — Maintenance of duplicate databases (Обслуживание дубликатов баз данных)
- 674 — Procedure call documents: Version 2 (Документы по вызову процедур, версия 2)
- 672 — Multi-site data collection facility (Возможность сбора данных на нескольких машинах)
- 671 — Note on Reconnection Protocol (Комментарии по поводу протокола восстановления соединения)
- 645 — Network Standard Data Specification syntax (Сетевой стандарт синтаксиса определения данных)
- 615 — Proposed Network Standard Data Pathname syntax (Предложенный сетевой стандарт синтаксиса полных имен данных)
- 610 — Further datalanguage design concepts (Дальнейшее развитие идей по разработке языков описания данных)
- 592 — Some thoughts on system design to facilitate resource sharing (Ряд мыслей по поводу разработки системы, облегчающей совместное использование ресурсов)
- 578 — Using MIT-Mathlab MACSYMA from MIT-DMS Muddle (Использование MIT-Mathlab MACSYMA из компании MIT-DMS)
- 515 — Specifications for datalanguage: Version 0/9 (Спецификация языка описания данных: версия 0/9)
- 500 — Integration of data management systems on a computer network (Интеграция системы управления данными с компьютерной сетью)
- 441 — Inter-Entity Communication — an experiment (Связь между объектами: эксперимент)
- 437 — Data Reconfiguration Service at UCSB (Служба реконфигурации данных Калифорнийского университета в Санта-Барбаре)
- 203 — Achieving reliable communication (Достижение надежного соединения)
- 76 — Connection by name: User oriented protocol (Подключение по имени: протокол, ориентированный на пользователя)
- 62 — Systems for Interprocess Communication in a Resource Sharing Computer Network (Системы, предназначенные для межпроцессного взаимодействия в компьютерных сетях, поддерживающих совместное использование ресурсов)
- 61 — Note on Interprocess Communication in a Resource Sharing Computer Network (Комментарии по поводу межпроцессного взаимодействия в компьютерных сетях, поддерживающих совместное использование ресурсов)
- 51 — Proposal for a Network Interchange Language (Предложение по языку сетевого обмена)
- 31 — Binary Message Forms in Computer (Формы двоичных сообщений в компьютере)

#### **5f. Протоколы для локальных сетей (NetBIOS)**

- 1002 — Protocol standard for a NetBIOS service on a TCP/UDP transport: Detailed specifications (Стандарт протокола для службы NetBIOS, использующей транспортные протоколы TCP/UDP: подробная спецификация)
- 1001 — Protocol standard for a NetBIOS service on a TCP/UDP transport: Concepts and methods (Стандарт протокола для службы NetBIOS, использующей транспортные протоколы TCP/UDP: общее представление и методы)

## **5g. Мобильность протокола IP и роуминг**

- 2607 — Proxy Chaining and Policy Implementation in Roaming (Цепочка proxy-серверов и выполнение правил в роуминге)
- 2548, 2138, 2058 — Microsoft Vendor-specific RADIUS Attributes (Атрибуты RADIUS фирмы Microsoft)
- 2501 — Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations (Специальные сети мобильной связи: проблема производительности протоколов маршрутизации и оценочный анализ)
- 2486 — The Network Access Identifier (Идентификатор доступа к сети)
- 2477 — Criteria for Evaluating Roaming Protocols (Критерий оценки протоколов роуминга)
- 2356 — Sun's SKIP Firewall Traversal for Mobile IP (Обход брандмауэра SKIP фирмы Sun для мобильного протокола IP)
- 2344 — Reverse Tunneling for Mobile IP (Обратное туннелирование для мобильного протокола IP)
- 2290 — Mobile-IPv4 Configuration Option for PPP IPCP (Параметры конфигурации мобильного протокола IPv4 для протокола PPP IPCP)
- 2194 — Review of Roaming Implementations (Обзор методов реализации роуминга)
- 2139, 2059 — RADIUS Accounting (Протокол обмена учетной информацией RADIUS)
- 2041 — Mobile Network Tracing (Мониторинг мобильной сети)
- 2005 — Applicability Statement for IP Mobility Support (Заявление по поводу применимости и поддержки мобильного протокола IP)
- 2002 — IP Mobility Support (Поддержка мобильного протокола IP)

## **5h. Остальное**

- 1546 — Host Anycasting Service (Служба альтернативной рассылки узла)
- 1312, 1159 — Message Send Protocol 2 (Протокол отправки сообщений, версия 2)
- 1151, 908 — Version 2 of the Reliable Data Protocol (RDP) (Протокол надежной передачи данных, версия 2)
- 1045 — VMTP: Versatile Message Transaction Protocol: Protocol specification (Спецификация многоцелевого протокола передачи сообщений о транзакциях)
- 998, 969 — NETBLT: A bulk data transfer protocol (NETBLT: протокол массовой пересылки данных)
- 979 — PSN End-to-End functional specification (Спецификация функциональных возможностей сквозного протокола узла коммутации пакетов)
- 869 — Host Monitoring Protocol (Протокол мониторинга узла сети)
- 643 — Network Debugging Protocol (Протокол сетевой отладки)
- 162 — NETBUGGER3 (Программа NETBUGGER3)

## **6. Уровень приложений**

### **6a. Протокол Telnet (TELNET)**

- 2355, 1647 — TN3270 Enhancements (Расширения TN3270)
- 1921 — TNVIP Protocol (Протокол TELNET для поддержки VIP-терминалов)
- 1646 — TN3270 Extensions for LName and Printer Selection (Расширения TN3270 для запроса указанного устройства и выбора принтера)
- 1576 — TN3270 Current Practices (Текущие реализации TELNET для терминала TN3270)
- 1205 — 5250 Telnet interface (Интерфейс TELNET терминала IBM 5250)
- 1184 — Telnet Linemode Option (Параметр Linemode протокола Telnet)

- 854, 764 — Telnet Protocol Specification (Спецификация протокола Telnet)  
818 — Remote User Telnet service (Служба Telnet для дистанционной работы пользователей)  
782 — Virtual Terminal management model (Модель управления виртуальным терминалом)  
728 — Minor pitfall in the Telnet Protocol (Незначительные ошибки в протоколе Telnet)  
703, 702, 701, 679, 669 — July, 1975, survey of New-Protocol Telnet Servers (Отчет за июль 1975 года по серверам нового протокола Telnet)  
688 — Tentative schedule for the new Telnet implementation for the TIP (Предварительный график работы команды разработки нового протокола Telnet)  
681 — Network UNIX (Сетевые средства UNIX)  
600 — Interfacing an Illinois plasma terminal to the ARPANET (Подключение плазменного терминала Illinois к ARPANET)  
596 — Second thoughts on Telnet Go-Ahead (Несколько мыслей “задним числом” по поводу команды Telnet Go-Ahead)  
595 — Second thoughts in defense of the Telnet Go-Ahead (Несколько мыслей “задним числом” в защиту команды Telnet Go-Ahead)  
593 — Telnet and FTP implementation schedule change (Изменение в графике работы команды разработки протоколов Telnet и FTP)  
576 — Proposal for modifying linking (Предложение по изменению связывания)  
570 — Experimental input mapping between NVT ASCII and UCSB On Line System (Опытное преобразование входящей информации между кодировками NVT ASCII и диалоговой системой Калифорнийского университета в Санта-Барбара)  
562 — Modifications to the Telnet specification (Изменения в спецификации протокола Telnet)  
559 — Comments on The New Telnet Protocol and its Implementation (Комментарии по поводу нового протокола Telnet и его реализации)  
529 — Note on protocol synch sequences (Заметки по поводу синхронизирующих последовательностей протокола)  
513 — Comments on the new Telnet specifications (Комментарии по поводу спецификации нового протокола Telnet)  
495 — Telnet Protocol specifications (Спецификация протокола Telnet)  
466 — Telnet logger/server for host LL-67 (Сервер регистрации Telnet для узла LL-67)  
452 — TELENET Command at Host LL (Команда TELENET на узле LL)  
435 — Telnet issues (Вопросы Telnet)  
426 — Reconnection Protocol (Протокол восстановления соединений)  
393 — Comments on Telnet Protocol Changes (Комментарии по поводу изменений к протоколу Telnet)  
377 — Using TSO via ARPA Network Virtual Terminal (Использование системы разделения времени на виртуальных терминалах в сети ARPA)  
357 — Echoing strategy for satellite links (Стратегия эхо-повтора для спутниковых каналов связи)  
355, 346 — Response to NWG/RFC 346 (Ответ на документ NWG/RFC 346)  
340 — Proposed Telnet Changes (Предложенные изменения в стандарт Telnet)  
339 — MLTNET: A Multi Telnet Subsystem for Tenex (MLTNET: подсистема для Tenex с поддержкой многих сеансов работы с Telnet)  
328 — Suggested Telnet Protocol Changes (Выдвигаемые предложения по изменению протокола Telnet)  
318 — Telnet Protocols (Протоколы Telnet)

- 216 — Telnet access to UCSB's On-Line System (Доступ через Telnet к диалоговой системе Калифорнийского университета в Санта-Барбаре)
- 215 — NCP, ICP, and Telnet: The Terminal IMP implementation (Протоколы NCP, ICP и Telnet: реализация для терминала IMP)
- 206 — User Telnet — description of an initial implementation (Пользовательский протокол Telnet — описание начальной реализации)
- 205 — NETCRT — a character display protocol (NETCRT — протокол отображения символов)
- 190 — DEC PDP-10-IMLAC communications system (Коммуникационная система DEC PDP-10-IMLAC)
- 158 — Telnet Protocol: A Proposed Document (Протокол Telnet: предлагаемый документ)
- 139 — Discussion of Telnet Protocol (Обсуждение протокола Telnet)
- 137 — Telnet Protocol — a proposed document (Протокол Telnet — предлагаемый документ)
- 135, 110 — Response to NWG/RFC 110 (Ответ на документ NWG/RFC 110)
- 103 — Implementation of Interrupt Keys (Реализация клавиш прерывания)
- 97 — First Cut at a Proposed Telnet Protocol (Первый отзыв на предложенный стандарт протокола Telnet)
- 91 — Proposed User-User Protocol (Предложенный протокол “пользователь-пользователь”)

#### **6b. Параметры протокола Telnet**

- 2217 — Telnet Com Port Control Option (Параметр управления COM-портом протокола Telnet)
- 2066 — TELNET CHARSET Option (Параметр CHARSET протокола Telnet)
- 1572, 1408 — Telnet Environment Option (Параметр среды протокола Telnet)
- 1571 — Telnet Environment Option Interoperability Issues (Проблемы использования параметра среды протокола Telnet на разных платформах)
- 1416, 1409 — Telnet Authentication Option (Параметр аутентификации протокола Telnet)
- 1412 — Telnet Authentication: SPX (Аутентификация Telnet: протокол SPX)
- 1411 — Telnet Authentication: Kerberos Version 4 (Аутентификация Telnet: протокол Kerberos Version 4)
- 1372, 1080 — Telnet Remote Flow Control Option (Параметр дистанционного управления потоком Telnet)
- 1143 — The Q Method of Implementing TELNET Option Negotiation (Q-метод реализации режима согласования параметров протокола Telnet)
- 1116 — Telnet Linemode option (Параметр Linemode протокола Telnet)
- 1096 — Telnet X display location option (Параметр обнаружения X-терминала протокола Telnet)
- 1091 — Telnet terminal-type option (Параметр определения типа терминала протокола Telnet)
- 1079 — Telnet terminal speed option (Параметр определения скорости передачи данных протокола Telnet)
- 1073 — Telnet window size option (Параметр определения размера окна протокола Telnet)
- 1053 — Telnet X.3 PAD option (Параметр X.3 PAD протокола Telnet)
- 1043 — Telnet Data Entry Terminal option: DODIIS implementation (Параметр протокола Telnet для терминала ввода данных: реализация DODIIS)
- 1041 — Telnet 3270 regime option (Параметр режима 3270 протокола Telnet)
- 946 — Telnet terminal location number option (Параметр обнаружения номера терминала протокола Telnet)

- 933 — Output marking Telnet option (Параметр протокола Telnet для маркировки выходных данных)
- 930 — Telnet terminal type option (Параметр протокола Telnet для определения типа терминала)
- 927 — TACACS user identification Telnet option (Параметр протокола Telnet для идентификации пользователя с помощью системы управления доступом ТАС)
- 885 — Telnet end of record option (Параметр протокола Telnet для определения конца записи)
- 884 — Telnet terminal type option (Параметр протокола Telnet для определения типа терминала)
- 861 — Telnet Extended Options: List Option (Расширенные параметры протокола Telnet: параметр списка)
- 860 — Telnet Timing Mark Option (Параметр протокола Telnet для определения временной метки)
- 859 — Telnet Status Option (Параметр состояния протокола Telnet)
- 858 — Telnet Suppress Go Ahead Option (Параметр подавления команды Go-Ahead протокола Telnet)
- 857 — Telnet Echo Option (Параметр эха протокола Telnet)
- 856 — Telnet Binary Transmission (Передача бинарных данных по протоколу Telnet)
- 855 — Telnet Option Specifications (Спецификация параметров протокола Telnet)
- 779 — Telnet send-location option (Параметр send-location протокола Telnet)
- 749 — Telnet SUPDUP-Output option (Параметр SUPDUP-Output протокола Telnet)
- 747 — Recent extensions to the SUPDUP Protocol (Новые расширения протокола SUPDUP)
- 746 — SUPDUP graphics extension (Графическое расширение протокола SUPDUP)
- 736 — Telnet SUPDUP option (Параметр SUPDUP протокола Telnet)
- 735 — Revised Telnet byte macro option (Параметр макробайта протокола Telnet)
- 732, 731 — Telnet Data Entry Terminal option (Параметр протокола Telnet для терминала ввода данных)
- 729 — Telnet byte macro option (Параметр макробайта протокола Telnet)
- 727 — Telnet logout option (Параметр завершения сеанса работы протокола Telnet)
- 726 — Remote Controlled Transmission and Echoing Telnet option (Параметр дистанционного управления передачей и эха протокола Telnet)
- 719 — Discussion on RCTE (Подробное обсуждение протокола RCTE)
- 718 — Comments on RCTE from the Tenex Implementation Experience (Комментарии по поводу протокола RCTE, основанные на опыте реализации протокола Telnet для системы Tenex)
- 698 — Telnet extended ASCII option (Параметр расширения кодировки ASCII протокола Telnet)
- 659 — Announcing additional Telnet options (Анонсирование дополнительных параметров протокола Telnet)
- 658 — Telnet output linefeed disposition (Параметр, согласующий положение символа перевода строки в выходном потоке данных)
- 657 — Telnet output vertical tab disposition option (Параметр, согласующий положение символа вертикальной табуляции в выходном потоке данных)
- 656 — Telnet output vertical tabstops option (Параметр, согласующий положение символа вертикальной табуляции в выходном потоке данных)
- 655 — Telnet output formfeed disposition option (Параметр, согласующий положение символа прогона страницы в выходном потоке данных)

- 654 — Telnet output horizontal tab disposition option (Параметр, согласующий положение символа горизонтальной табуляции в выходном потоке данных)
- 653 — Telnet output horizontal tabstops option (Параметр, согласующий положение символа горизонтальной табуляции в выходном потоке данных)
- 652 — Telnet output carriage-return disposition option (Параметр, согласующий положение символа перевода строки в выходном потоке данных)
- 651 — Revised Telnet status option (Пересмотренный параметр состояния протокола Telnet)
- 587 — Announcing new Telnet options (Анонсирование новых параметров протокола Telnet)
- 581 — Corrections to RFC 560: Remote Controlled Transmission and Echoing Telnet Option (Исправления в RFC 560: дистанционное управление передачей и эхом в протоколе Telnet)
- 563 — Comments on the RCTE Telnet option (Комментарии по поводу параметра RCTE протокола Telnet)
- 560 — Remote Controlled Transmission and Echoing Telnet option (Параметр дистанционного управления передачей и эхом протокола Telnet)

#### **6с. Передача файлов и протоколы доступа к файлам (FTP, TFTP, SFTP, NFS)**

- 2640 — Internationalization of the File Transfer Protocol (Локализация протокола передачи файлов)
- 2624 — NFS Version 4 Design Considerations (Анализ проекта 4-й версии протокола NFS)
- 2623 — NFS Version 2 and Version 3 Security Issues and the NFS Protocol's Use of RPCSEC\_GSS and Kerberos V5 (Вопросы безопасности протоколов NFS версий 2 и 3 и использование RPCSEC\_GSS и Kerberos V5 в протоколе NFS)
- 2577 — FTP Security Considerations (Анализ безопасности протокола FTP)
- 2428 — FTP Extensions for IPv6 and NATs (Расширения протокола FTP для IPv6 и NAT)
- 2389 — Feature negotiation mechanism for the File Transfer Protocol (Механизм согласования функциональных возможностей для протокола передачи файлов)
- 2349, 1784 — TFTP Timeout Interval and Transfer Size Options (Величина таймаута для протокола TFTP и параметры, влияющие на размер пересылаемых данных)
- 2348, 1783 — TFTP Blocksize Option (Параметр Blocksize протокола TFTP)
- 2347, 1782 — TFTP Option Extension (Расширение параметров протокола TFTP)
- 2228 — FTP Security Extensions (Расширения системы безопасности протокола FTP)
- 2224 — NFS URL Scheme (Схема URL NFS)
- 2204 — ODETTE File Transfer Protocol (Протокол передачи файлов ODETTE)
- 2090 — TFTP Multicast Option (Параметр многоадресатной передачи протокола TFTP)
- 2055 — WebNFS Server Specification (Спецификация сервера WebNFS)
- 2054 — WebNFS Client Specification (Спецификация клиента WebNFS)
- 1986 — Experiments with a Simple File Transfer Protocol for Radio Links using Enhanced Trivial File Transfer Protocol (ETFTP) (Эксперименты с простым протоколом передачи файлов для радиоканалов связи с использованием расширенного протокола TFTP)
- 1813 — NFS Version 3 Protocol Specification (Спецификация протокола NFS версии 3)

- 1785 — TFTCP Option Negotiation Analysis (Анализ механизма согласования параметров протокола TFTCP)
- 1639, 1545 — FTP Operation Over Big Address Records (FOOBAR) (Работа протокола FTP при использовании больших адресов)
- 1635 — How to Use Anonymous FTP (Как пользоваться анонимным FTP)
- 1579 — Firewall-Friendly FTP (Протокол FTP с поддержкой брандмауэров)
- 1440 — SIFT/UFT: Sender-Initiated/Unsolicited File Transfer (SIFT/UFT: передача файлов по инициативе отправителя)
- 1415 — FTP-FTAM Gateway Specification (Спецификация шлюза FTP-FTAM)
- 1350, 783 — The TFTP Protocol (Revision 2) (Протокол TFTP, вторая редакция)
- 1282, 1258 — BSD Rlogin (Описание утилиты Rlogin системы BSD)
- 1235 — Coherent File Distribution Protocol (Протокол распространения связанных файлов)
- 1094 — NFS: Network File System Protocol specification (Спецификация протокола сетевой файловой системы NFS)
- 1068 — Background File Transfer Program (BFTP) (Протокол фоновой передачи файлов)
- 1037 — NFILE — a file access protocol (NFILE — протокол доступа к файлам)
- 959, 765, 542, 354, 265, 172, 114 — File Transfer Protocol (Протокол передачи файлов)
- 949 — FTP unique-named store command (Команда записи unique-named протокола FTP)
- 913 — Simple File Transfer Protocol (Простой протокол передачи файлов)
- 906 — Bootstrap loading using TFTP (Процесс начальной загрузки с помощью протокола TFTP)
- 775 — Directory oriented FTP commands (Команды протокола FTP, предназначенные для работы с каталогами)
- 743 — FTP extension: XRSQ/XRCP (Расширение протокола FTP: команды XRSQ/XRCP)
- 737 — FTP extension: XSEN (Расширение протокола FTP: команда XSEN)
- 697 — CWD command of FTP (Команда CWD протокола FTP)
- 691 — One more try on the FTP (Еще одна попытка разработки протокола FTP)
- 686 — Leaving well enough alone (Лучшее враг — хорошего)
- 683 — FTPSRV — Tenex extension for paged files (FTPSRV — расширения системы Tenex для постраничной разбивки файлов)
- 662 — Performance improvement in ARPANET file transfers from Multics (Улучшение производительности в механизме передачи файлов в сети ARPANET от Multics)
- 640 — Revised FTP reply codes (Пересмотренные коды ответа протокола FTP)
- 630 — FTP error code usage for more reliable mail service (Использование кодов ошибки протокола FTP для повышения надежности почтовой службы)
- 624 — Comments on the File Transfer Protocol (Комментарии по поводу протокола передачи файлов)
- 614 — Response to RFC 607: “Comments on the File Transfer Protocol” (Ответ на документ RFC 607)
- 607 — Comments on the File Transfer Protocol (Комментарии по поводу протокола передачи файлов)
- 571 — Tenex FTP problem (Проблема с протоколом FTP в системе Tenex)
- 535 — Comments on File Access Protocol (Комментарии по поводу протокола доступа к файлам)
- 532 — UCSD-CC Server-FTP facility (Устройство FTP-сервера Калифорнийского университета в Сан-Диего)

- 520 — Memo to FTP group: Proposal for File Access Protocol (Памятка для группы разработки протокола FTP: предложение по протоколу доступа к файлам)
- 506 — FTP command naming problem (Проблема с выбором имен команд протокола FTP)
- 505 — Two solutions to a file transfer access problem (Два решения проблемы доступа к процессу передачи файлов)
- 501 — Un-muddling “free file transfer” (Попытка упорядочения “бесплатной пересылки файлов”)
- 487 — Free file transfer (Бесплатная пересылка файлов)
- 486 — Data transfer revisited (Возвращаясь к передаче данных)
- 480 — Host-dependent FTP parameters (Параметры протокола FTP, зависящие от узла сети)
- 479 — Use of FTP by the NIC Journal (Использование протокола FTP в журнале NIC)
- 478 — FTP server-server interaction — II (Взаимодействие между двумя FTP-серверами — II)
- 468 — FTP data compression (Сжатие данных при пересылке по протоколу FTP)
- 463 — FTP comments and response to RFC 430 (Комментарии по поводу протокола FTP и ответ на RFC 430)
- 448 — Print files in FTP (Печать файлов с помощью протокола FTP)
- 438 — FTP server-server interaction (Взаимодействие между двумя FTP-серверами)
- 430 — Comments on File Transfer Protocol (Комментарии по поводу протокола передачи файлов)
- 418 — Server file transfer under TSS/360 at NASA Ames (Сервер для передачи файлов, работающий под управлением TSS/360 в центре NASA)
- 414 — File Transfer Protocol (FTP) status and further comments (Состояние протокола передачи файлов (FTP) и дополнительные комментарии)
- 412 — User FTP Documentation (Документация для пользователя протокола FTP)
- 385 — Comments on the File Transfer Protocol (Комментарии по поводу протокола передачи файлов)
- 310 — Another Look at Data and File Transfer Protocols (Альтернативный взгляд на протоколы передачи данных и файлов)
- 294 — The Use of “Set Data Type” Transaction in File Transfer Protocol (Использование транзакции “Установить тип данных” в протоколе передачи файлов)
- 281 — Suggested addition to File Transfer Protocol (Предполагаемые дополнения к протоколу передачи файлов)
- 269 — Some Experience with File Transfer (Некоторый опыт использования передачи файлов)
- 264, 171 — The Data Transfer Protocol (Протокол передачи данных)
- 250 — Some thoughts on file transfer (Несколько мыслей по поводу передачи файлов)
- 242 — Data Descriptive Language for Shared Data (Язык описания данных для совместно используемых данных)
- 238 — Comments on DTP and FTP proposals (Комментарии по поводу предлагаемых протоколов DTP и FTP)
- 163 — Data transfer protocols (Протоколы передачи данных)
- 141 — Comments on RFC 114: A File Transfer Protocol (Комментарии по поводу документа RFC 114: протокол передачи файлов)
- 133 — File Transfer and Recovery (Передача файлов и восстановление после сбоев)

#### **6d. Система доменных имен (DNS)**

- 2673 — Binary Labels in the Domain Name System (Бинарные метки в системе доменных имен)
- 2672 — Non-Terminal DNS Name Redirection (Нетерминальная переадресация имен системы DNS)
- 2671 — Extension Mechanisms for DNS (EDNS0) (Механизмы расширения системы DNS (EDNS0))
- 2606 — Reserved Top Level DNS Names (Зарезервированные имена доменов верхнего уровня системы DNS)
- 2541 — DNS Security Operational Considerations (Анализ текущей системы безопасности DNS)
- 2540 — Detached Domain Name System (DNS) Information (Информация по автономной системе доменных имен)
- 2539 — Storage of Diffie-Hellman Keys in the Domain Name System (DNS) (Сохранение ключей Диффи-Хелмана в системе доменных имен)
- 2535 — Domain Name System Security Extensions (Расширения системы безопасности DNS)
- 2517 — Building Directories from DNS: Experiences from WWWSeeker (Создание каталога на основе системы DNS: результаты опытов WWWSeeker)
- 2352, 2240 — A Convention For Using Legal Names as Domain Names (Соглашение по использованию легальных имен в системе доменных имен)
- 2317 — Classless IN-ADDR.ARPA delegation (Бесклассовое делегирование домена IN-ADDR.ARPA)
- 2308 — Negative Caching of DNS Queries (DNS NCACHE) (Кэширование отрицательных DNS-запросов)
- 2230 — Key Exchange Delegation Record for the DNS (Запись о делегировании процедуры обмена ключей для системы DNS)
- 2219 — Use of DNS Aliases for Network Services (Использование псевдонимов системы DNS для сетевых служб)
- 2182 — Selection and Operation of Secondary DNS Servers (Выбор и запуск в работу вторичного сервера DNS)
- 2181 — Clarifications to the DNS Specification (Разъяснения к спецификации DNS)
- 2137 — Secure Domain Name System Dynamic Update (Защищенное динамическое обновление системы доменных имен)
- 2136 — Dynamic Updates in the Domain Name System (DNS UPDATE) (Динамические обновления в системе доменных имен)
- 2065 — Domain Name System Security Extensions (Расширения системы безопасности DNS)
- 2052 — A DNS RR for specifying the location of services (DNS SRV) (Запись ресурса системы DNS, определяющая размещение служб)
- 2010 — Operational Criteria for Root Name Servers (Критерий работоспособности корневых серверов системы доменных имен)
- 1996 — A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY) (Механизм быстрого уведомления об изменениях в зоне)
- 1995 — Incremental Zone Transfer in DNS (Инкрементальная пересылка зоны в DNS)
- 1982 — Serial Number Arithmetic (Вычисление серийных номеров)
- 1912, 1537 — Common DNS Operational and Configuration Errors (Часто встречающиеся ошибки в работе системы DNS и ее конфигурации)
- 1876 — A Means for Expressing Location Information in the Domain Name System (Средства для оперативного поиска информации в системе доменных имен)
- 1794 — DNS Support for Load Balancing (Поддержка механизма выравнивания нагрузки в DNS)

- 1713 — Tools for DNS debugging (Средства для отладки работоспособности системы DNS)
- 1712 — DNS Encoding of Geographical Location (Кодирование географических мест в системе DNS)
- 1706, 1637, 1348 — DNS NSAP Resource Records (Запись ресурса NSAP системы DNS)
- 1591 — Domain Name System Structure and Delegation (Структура системы доменных имен и правила делегирования)
- 1536 — Common DNS Implementation Errors and Suggested Fixes (Часто встречающиеся ошибки в реализации системы DNS и предложения по их устранению)
- 1535 — A Security Problem and Proposed Correction With Widely Deployed DNS Software (Проблема безопасности и предлагаемые методы ее решения для широко используемого программного обеспечения системы DNS)
- 1480, 1386 — The US Domain (Домен US)
- 1464 — Using the Domain Name System To Store Arbitrary String Attributes (Использование системы доменных имен для сохранения произвольных атрибутов текстовой строки)
- 1394 — Relationship of Telex Answerback Codes to Internet Domains (Взаимосвязь кодов ответа системы Telex и доменов Internet)
- 1183 — New DNS RR Definitions (Определения новых записей ресурсов системы DNS)
- 1101 — DNS encoding of network names and other types (Кодирование имен сетей и других типов в системе DNS)
- 1035 — Domain names — implementation and specification (Доменные имена: реализация и спецификация)
- 1034 — Domain names — concepts and facilities (Доменные имена: концепции и возможности)
- 1033 — Domain administrators operations guide (Должностная инструкция администратора домена)
- 1032 — Domain administrators guide (Руководство администратора домена)
- 1031 — MILNET name domain transition (Перемещение домена имен MILNET)
- 973 — Domain system changes and observations (Изменения в системе доменных имен и некоторые замечания)
- 953, 811 — Hostname Server (Сервер имен узлов)
- 921, 897 — Domain name system implementation schedule — revised (Пересмотренный график реализации системы доменных имен)
- 920 — Domain requirements (Требования к доменам)
- 883 — Domain names: Implementation specification (Доменные имена: техническое описание на реализацию)
- 882 — Domain names: Concepts and facilities (Доменные имена: концепции и возможности)
- 881 — Domain names plan and schedule (План и график реализации системы доменных имен)
- 830 — Distributed system for Internet name service (Распределенная система для службы имен Internet)
- 819 — Domain naming convention for Internet user applications (Соглашение по системе доменных имен для пользовательских приложений)
- 799 — Internet name domains (Домены имен Internet)
- 756 — NIC name server: a datagram-based information utility (Сервер имен NIC: информационная утилита для анализа дейтаграмм)
- 752 — Universal host table (Универсальная таблица узлов)

## **6е. Системы электронной почты и рассылки сообщений (SMTP, MIME, POP, IMAP, X.400)**

- 2683 — IMAP4 Implementation Recommendations (Рекомендации по реализации протокола IMAP4)
- 2646 — The Text/Plain Format Parameter (Параметр форматирования Text/Plain)
- 2645 — ON-DEMAND MAIL RELAY (ODMR) SMTP with Dynamic IP Addresses (Ретрансляция электронной почты по требованию: SMTP-сервер с динамически назначаемым IP-адресом)
- 2634 — Enhanced Security Services for S/MIME (Расширение служб безопасности для S/MIME)
- 2633 — S/MIME Version 3 Message Specification (Спецификация формата сообщений S/MIME версии 3)
- 2632 — S/MIME Version 3 Certificate Handling (Обработка сертификатов в S/MIME версии 3)
- 2595 — Using TLS with IMAP, POP3 and ACAP (Использование протокола TLS совместно с протоколами IMAP, POP3 и ACAP)
- 2586 — The Audio/L16 MIME content type (Тип содержимого MIME Audio/L16)
- 2557, 2110 — MIME Encapsulation of Aggregate Documents, such as HTML (MHTML) (Инкапсуляция MIME сводных документов, таких как HTML (MHTML))
- 2554 — SMTP Service Extension for Authentication (Расширение службы SMTP для поддержки аутентификации)
- 2530 — Indicating Supported Media Features Using Extensions to DSN and MDN (Индикация поддерживаемых возможностей среди передачи данных с помощью расширений к DSN и MDN)
- 2524 — Neda's Efficient Mail Submission and Delivery (EMSD) Protocol Specification Version 1.3 (Спецификация эффективного протокола приема и доставки электронной почты фирмы Neda Communication версии 1.3)
- 2505 — Anti-Spam Recommendations for SMTP MTAs (Рекомендации по борьбе со спамом для агентов передачи сообщений протокола SMTP)
- 2503 — MIME Types for Use with the ISO ILL Protocol (Типы MIME, которые используются с протоколом ISO ILL)
- 2487 — SMTP Service Extension for Secure SMTP over TLS (Расширение службы SMTP для создания защищенной версии протокола SMTP на основе протокола TLS)
- 2480 — Gateways and MIME Security Multiparts (Шлюзы и защищенные части MIME-сообщения)
- 2476 — Message Submission (Процесс подачи сообщений)
- 2449 — POP3 Extension Mechanism (Механизм расширений протокола POP3)
- 2442 — The Batch SMTP Media Type (Тип содержимого Batch-SMTP)
- 2426 — vCard MIME Directory Profile (Каталожный профиль MIME vCard)
- 2425 — A MIME Content-Type for Directory Information (Тип содержимого MIME для указания каталожной информации)
- 2424 — Content Duration MIME Header Definition (Определение заголовка MIME Content-Duration)
- 2387, 2112, 1872 — The MIME Multipart/Related Content-type (Тип содержимого MIME Multipart/Related)
- 2384 — POP URL Scheme (Схема URL POP)
- 2359 — IMAP4 UIDPLUS extension (Расширение UIDPLUS протокола IMAP4)
- 2342 — IMAP4 Namespace (Пространство имен протокола IMAP4)
- 2318 — The text/css Media Type (Тип содержимого text/css)
- 2312 — S/MIME Version 2 Certificate Handling (Обработка сертификатов в S/MIME версии 2)

- 2311 — S/MIME Version 2 Message Specification (Спецификация формата сообщений S/MIME версии 2)
- 2302 — Tag Image File Format (TIFF) — image/tiff MIME Sub-type Registration (Регистрация подтипа MIME image/tiff)
- 2298 — An Extensible Message Format for Message Disposition Notifications (Расширяемый формат сообщения для уведомления о положении сообщения)
- 2231, 2184 — MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations (Значения параметров MIME и кодирование расширенных атрибутов слов: наборы символов, языки и продолжения)
- 2221 — IMAP4 Login Referrals (Перенесение регистрации на другие серверы IMAP4)
- 2220 — The Application/MARC Content-type (Тип содержимого Application/MARC)
- 2197, 1854 — SMTP Service Extension for Command Pipelining (Расширение службы SMTP для конвейера команд)
- 2195, 2095 — IMAP/POP AUTHorize Extension for Simple Challenge/Response (Расширение протокола IMAP/POP AUTHorize для простой аутентификации по методу запрос/ответ)
- 2193 — IMAP4 Mailbox Referrals (Перенесение почтовых ящиков на другие серверы IMAP4)
- 2192 — IMAP URL Scheme (Схема URL IMAP)
- 2180 — IMAP4 Multi-Accessed Mailbox Practice (Практика работы с почтовыми ящиками множественного доступа по протоколу IMAP4)
- 2177 — IMAP4 IDLE command (Команда IDLE протокола IMAP4)
- 2164, 1838 — Use of an X.500/LDAP directory to support MIXER address mapping (Использование каталога X.500/LDAP для поддержки соответствия адресов MIXER)
- 2163, 1664 — Using the Internet DNS to Distribute MIXER Conformant Global Address Mapping (MCGAM) (Использование системы DNS сети Internet для распространения матрицы отображения глобальных сетевых адресов MIXER)
- 2162, 1405 — MaXIM-11 — Mapping between X.400 / Internet mail and Mail-11 mail (Отображение между почтовыми системами X.400 / Internet и Mail-11)
- 2161 — A MIME Body Part for ODA (Тип содержимого MIME application/oda)
- 2160 — Carrying PostScript in X.400 and MIME (Передача данных в формате PostScript в системах X.400 и MIME)
- 2158 — X.400 Image Body Parts (Описание частей тела сообщения, предназначенных для пересылки изображений в системе X.400)
- 2157 — Mapping between X.400 and RFC-822/MIME Message Bodies (Отображение тела сообщения между форматами X.400 и RFC-822/MIME)
- 2156, 1495, 1327, 1148, 1138 — MIXER (Mime Internet X.400 Enhanced Relay): Mapping between X.400 and RFC 822/MIME (MIXER (Расширенный ретранслятор сообщений MIME системы X.400: соответствие стандартов X.400 и RFC 822/MIME))
- 2152, 1642 — UTF-7. A Mail-Safe Transformation Format of Unicode (UTF-7. Надежный формат преобразования электронной почты на основе Unicode)
- 2142 — Mailbox Names for Common Services, Roles and Functions (Имена почтовых ящиков для общих служб, ролей и функций)
- 2088 — IMAP4 non-synchronizing literals (Несинхронизирующиеся литералы протокола IMAP4)
- 2087 — IMAP4 QUOTA extension (Расширение QUOTA протокола IMAP4)

- 2086 — IMAP4 ACL extension (Расширение ACL протокола IMAP4)  
2077 — The Model Primary Content Type for Multipurpose Internet Mail Extensions (Модель основного типа содержимого для многоцелевых расширений электронной почты в сети Internet)  
2076 — Common Internet Message Headers (Общие заголовки для сообщений Internet)  
2062 — Internet Message Access Protocol — Obsolete Syntax (Протокол доступа к сообщениям Internet — устаревший синтаксис)  
2061, 2060, 1730 — IMAP4 Compatibility with IMAP2bis (Совместимость протоколов IMAP4 и IMAP2bis)  
2049 — Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples (Многоцелевые расширения электронной почты в сети Internet (MIME), часть V: критерий соответствия и примеры)  
2048 — Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures (Многоцелевые расширения электронной почты в сети Internet (MIME), часть IV: процедуры регистрации)  
2047, 1522, 1342 — MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text (Многоцелевые расширения электронной почты в сети Internet (MIME), часть III: расширения заголовков сообщений для текста, представленного не в кодировке ASCII)  
2046 — Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types (Многоцелевые расширения электронной почты в сети Internet (MIME), часть II: типы носителя)  
2045, 1521, 1341 — Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies (Многоцелевые расширения электронной почты в сети Internet (MIME), часть I: формат тела сообщения Internet)  
2034 — SMTP Service Extension for Returning Enhanced Error Codes (Расширение службы SMTP, возвращающее расширенные коды ошибок)  
2033 — Local Mail Transfer Protocol (Протокол передачи локальной почты)  
2017 — Definition of the URL MIME External-Body Access-Type (Определение MIME URL, параметры External-Body и Access-Type)  
2015 — MIME Security with Pretty Good Privacy (PGP) (Безопасность MIME с системой PGP (набор алгоритмов и программ для высоконадежного шифрования сообщений с использованием открытых ключей))  
1985 — SMTP Service Extension for Remote Message Queue Starting (Расширение службы SMTP для инициирования начала обработки очереди сообщений сервером, предназначенных для передачи конкретному удаленному узлу)  
1957 — Some Observations on Implementations of the Post Office Protocol (POP3) (Некоторые замечания по поводу реализации протокола POP3)  
1939, 1725, 1460, 1225, 1082, 1081 — Post Office Protocol — Version 3 (Почтовый протокол, версия 3)  
1896, 1563, 1523 — The text/enriched MIME Content-type (Тип содержимого MIME text/enriched)  
1895 — The Application/CALS-1840 Content-type (Тип содержимого MIME Application/CALS-1840)  
1894 — An Extensible Message Format for Delivery Status Notifications (Расширяемый формат сообщений для уведомлений о доставке)  
1893 — Enhanced Mail System Status Codes (Коды состояния расширенной почтовой системы)  
1892 — The Multipart/Report Content Type for the Reporting of Mail System Administrative Messages (Тип содержимого MIME Multipart/Report для

- рассылки сообщений, содержащих уведомления об административных сообщениях почтовой системы)
- 1891 — SMTP Service Extension for Delivery Status Notifications (Расширение службы SMTP для уведомлений о доставке)
- 1873 — Message/External-Body Content-ID Access Type (Тип доступа Content-ID для параметра Message/External-Body)
- 1870, 1653, 1427 — SMTP Service Extension for Message Size Declaration (Расширение службы SMTP для объявления о размере сообщения)
- 1869, 1651, 1425 — SMTP Service Extensions (Расширения службы SMTP)
- 1864, 1544 — The Content-MD5 Header Field (Поле заголовка Content-MD5)
- 1848 — MIME Object Security Services (Службы безопасности объектов MIME)
- 1847 — Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted (Защищенные части сообщения MIME: параметры Multipart/Signed и Multipart/Encrypted)
- 1846 — SMTP 521 Reply Code (Код ответа 521 протокола SMTP)
- 1845 — SMTP Service Extension for Checkpoint/Restart (Расширение службы SMTP для команд Checkpoint/Restart)
- 1844, 1820 — Multimedia E-mail (MIME) User Agent Checklist (Контрольный список, облегчающий оценку агента пользователя, предназначенного для передачи мультимедийных (MIME) сообщений электронной почты)
- 1830 — SMTP Service Extensions for Transmission of Large and Binary MIME Messages (Расширения службы SMTP для передачи больших и двоичных MIME-сообщений)
- 1807, 1357 — A Format for Bibliographic Records (Формат для передачи библиографических записей)
- 1806 — Communicating Presentation Information in Internet Messages: The Content-Disposition Header (Коммуникационная презентационная информация в сообщениях Internet: заголовок Content-Disposition)
- 1767 — MIME Encapsulation of EDI Objects (Инкапсуляция MIME объектов EDI)
- 1741 — MIME Content Type for BinHex Encoded Files (Тип содержимого MIME для файлов, закодированных утилитой BinHex)
- 1740 — MIME Encapsulation of Macintosh Files — MacMIME (Инкапсуляция MIME файлов Macintosh)
- 1734 — POP3 AUTHentication command (Команда аутентификации протокола POP3)
- 1733 — Distributed Electronic Mail Models in IMAP4 (Распределенные модели электронной почты в протоколе IMAP4)
- 1732 — IMAP4 Compatibility with IMAP2 and IMAP2bis (Совместимость протокола IMAP4 с IMAP2 и IMAP2bis)
- 1731 — IMAP4 Authentication Mechanisms (Механизмы аутентификации протокола IMAP4)
- 1711 — Classifications in E-mail Routing (Классификации в системе маршрутизации электронной почты)
- 1685 — Writing X.400 O/R Names (Запись O/R имен системы X.400)
- 1652, 1426 — SMTP Service Extension for 8bit-MIMETransport (Расширение службы SMTP для передачи 8-битовых символов в кодировке MIME)
- 1649 — Operational Requirements for X.400 Management Domains in the GO-MHS Community (Эксплуатационные требования для управления доменами системы X.400 в сообществе GO-MHS (Глобальная открытая система управления сообщениями))
- 1648 — Postmaster Convention for X.400 Operations (Соглашения по использованию адреса postmaster в X.400)
- 1641 — Using Unicode with MIME (Использование юникода в кодировке MIME)

- 1616 — X.400(1988) for the Academic and Research Community in Europe (Отчет об использовании системы X.400(1988) для учебных заведений и исследовательских организаций в Европе)
- 1615 — Migrating from X.400(84) to X.400(88) (Переход с X.400(84) на X.400(88))
- 1590 — Media Type Registration Procedure (Процедура регистрации параметра Media Type)
- 1556 — Handling of Bi-directional Texts in MIME (Двунаправленная обработка текстов в MIME)
- 1524 — A User Agent Configuration Mechanism For Multimedia Mail Format Information (Механизм информирования агента пользователя о формате мультимедийного сообщения)
- 1506 — A Tutorial on Gateways between X.400 and Internet Mail (Учебное пособие по созданию почтовых шлюзов между X.400 и Internet)
- 1505, 1154 — Encoding Header Field for Internet Messages (Кодирование полей заголовка для сообщений Internet)
- 1502 — X.400 Use of Extended Character Sets (Использование расширенного набора символов в X.400)
- 1496 — Rules for downgrading messages from X.400/88 to X.400/84 when MIME content-types are present in the messages (Правила для обратного конвертирования сообщений системы X.400/88 в систему X.400/84 при условии, что MIME-заголовок content-types присутствует в сообщении)
- 1494 — Equivalences between 1988 X.400 and RFC-822 Message Bodies (Эквивалентные части тела сообщения систем 1988 X.400 и RFC-822)
- 1428 — Transition of Internet Mail from Just-Send-8 to 8bit-SMTP/MIME (Переход почтовых систем Internet от несанкционированной рассылки 8-битовых сообщений к 8-битовому протоколу SMTP/MIME)
- 1344 — Implications of MIME for Internet Mail Gateways (Использование стандарта MIME для создания шлюзов электронной почты Internet)
- 1343 — A User Agent Configuration Mechanism for Multimedia Mail Format Information (Механизм информирования агента пользователя о формате мультимедийного сообщения)
- 1339 — Remote Mail Checking Protocol (Протокол дистанционной проверки электронной почты)
- 1328 — X.400 1988 to 1984 downgrading (Обратный переход от системы X.400 1988 к системе X.400 1984)
- 1211 — Problems with the maintenance of large mailing lists (Проблемы обслуживания больших списков рассылки)
- 1204 — Message Posting Protocol (MPP) (Протокол отправки сообщений)
- 1203, 1176, 1064 — Interactive Mail Access Protocol: Version 3 (Протокол интерактивного доступа к электронной почте, версия 3)
- 1168 — Intermail and Commercial Mail Relay services (Бесплатные и коммерческие службы пересылки электронной почты)
- 1153 — Digest message format (Формат сообщений для дайджеста)
- 1137 — Mapping between full RFC 822 and RFC 822 with restricted encoding (Соответствие между системами электронной почты с полной и ограниченной поддержкой RFC 822)
- 1090 — SMTP on X.25 (Протокол SMTP в сети X.25)
- 1056, 993, 984 — PCMAIL: A distributed mail system for personal computers (PCMAIL: распределенная почтовая система для персональных компьютеров)
- 1049 — Content-type header field for Internet messages (Поле заголовка Content-type для сообщений Internet)

- 1047 — Duplicate messages and SMTP (Дублирование сообщений и протокол SMTP)
- 1026, 987 — Addendum to RFC 987: (Mapping between X.400 and RFC-822) (Приложение к документу RFC 987: соответствие между стандартами X.400 и RFC-822)
- 977 — Network News Transfer Protocol (Протокол передачи сетевых новостей)
- 976 — UUCP mail interchange format standard (Стандарт формата UUCP для обмена электронной почты)
- 974 — Mail routing and the domain system (Маршрутизация электронной почты и система доменных имен)
- 937, 918 — Post Office Protocol: Version 2 (Почтовый протокол, версия 2)
- 934 — Proposed standard for message encapsulation (Предложенный стандарт для инкапсуляции сообщений)
- 915 — Network mail path service (Служба сетевых маршрутов электронной почты)
- 886 — Proposed standard for message header munging (Предложенный стандарт для внесения изменений в заголовок сообщения)
- 841 — Specification for message format for Computer Based Message Systems (Спецификация формата сообщений для компьютерной системы обмена сообщениями)
- 822 — Standard for the format of ARPA Internet text messages (Стандартный формат текстовых сообщений, принятый для объединенной сети ARPA)
- 821, 788 — Simple Mail Transfer Protocol (Простой протокол передачи электронной почты)
- 806 — Proposed Federal Information Processing Standard: Specification for message format for computer based message systems (Предложенный стандарт обработки заявлений федеральным властям: спецификация формата сообщения для компьютерной системы обмена сообщениями)
- 786 — Mail Transfer Protocol: ISI TOPS20 MTP-NIMAIL interface (Протокол передачи электронной почты: интерфейс ISI TOPS20 MTP-NIMAIL)
- 785 — Mail Transfer Protocol: ISI TOPS20 file definitions (Протокол передачи электронной почты: определения файлов ISI TOPS20)
- 784 — Mail Transfer Protocol: ISI TOPS20 implementation (Протокол передачи электронной почты: реализация ISI TOPS20)
- 780, 772 — Mail Transfer Protocol (Протокол передачи электронной почты)
- 771 — Mail transition plan (Проект развития системы электронной почты)
- 763 — Role mailboxes (Ролевые почтовые ящики)
- 757 — Suggested solution to the naming, addressing, and delivery problem for ARPANET message systems (Предполагаемое решение проблемы именования, адресации и доставки для системы обмена сообщениями ARPANET)
- 754 — Out-of-net host addresses for mail (Адреса узлов для обмена электронной почтой, не имеющих прямого подключения к Internet)
- 753 — Internet Message Protocol (Протокол сообщений Internet)
- 751 — Survey of FTP mail and MLFL (Обзор службы электронной почты на основе протокола FTP и MLFL)
- 744 — MARS — a Message Archiving and Retrieval Service (MARS — Служба архивного хранения и поиска сообщений)
- 733 — Standard for the format of ARPA network text messages (Стандартный формат текстовых сообщений, принятый для объединенной сети ARPA)
- 724 — Proposed official standard for the format of ARPA Network messages (Предложенный стандарт формата текстовых сообщений для объединенной сети ARPA)

- 720 — Address Specification Syntax for Network Mail (Синтаксис определения адреса для сетевой почтовой системы)
- 706 — On the junk mail problem (Проблема отказа от сообщений электронной почты)
- 680 — Message Transmission Protocol (Протокол передачи сообщений)
- 644 — On the problem of signature authentication for network mail (К проблеме аутентификации подписи в сетевых системах электронной почты)
- 577 — Mail priority (Приоритет электронной почты)
- 574 — Announcement of a mail facility at UCSB (Уведомление о возможностях системы электронной почты Калифорнийского университета в Санта-Барбара)
- 561 — Standardizing Network Mail Headers (Стандартизация заголовков сетевых систем электронной почты)
- 555 — Responses to critiques of the proposed mail protocol (Ответ на критику по поводу предложенного почтового протокола)
- 539, 524 — Thoughts on the mail protocol proposed in RFC 524 (Мысли по поводу почтового протокола, предложенного в RFC 524)
- 498 — On mail service to CCN (По поводу почтовой службы для компьютерной сети связи)
- 491 — What is “Free”? (Что значит “бесплатно”?)
- 475 — FTP and network mail system (Протокол FTP и сетевая почтовая система)
- 458 — Mail retrieval via FTP (Получение электронной почты по протоколу FTP)
- 333 — Proposed experiment with a Message Switching Protocol (Предложенный эксперимент с протоколом коммутации сообщений)
- 278, 224, 221, 196 — Revision of the Mail Box Protocol (Изменения в протоколе почтового ящика)

#### **6f. Факсимильная связь и растровые изображения**

- 2639 — Internet Printing Protocol/1.0: Implementer’s Guide (Протокол печати в Internet, версия 1.0: руководство разработчика)
- 2569 — Mapping between LPD and IPP Protocols (Соответствия между протоколами LPD и IPP)
- 2568 — Rationale for the Structure of the Model and Protocol for the Internet Printing Protocol (Логическое обоснование структуры модели и протокола для протокола печати в Internet)
- 2567 — Design Goals for an Internet Printing Protocol (Цели разработки протокола печати в Internet)
- 2566 — Internet Printing Protocol/1.0: Model and Semantics (Протокол печати в Internet, версия 1.0: модель и семантика)
- 2565 — Internet Printing Protocol/1.0: Encoding and Transport (Протокол печати в Internet, версия 1.0: кодировка и транспортировка)
- 2542 — Terminology and Goals for Internet Fax (Терминология и задачи службы передачи факсов в сети Internet)
- 2534 — Media Features for Display, Print, and Fax (Особенности носителя для отображения, печати и передачи по факсу)
- 2532 — Extended Facsimile Using Internet Mail (Расширенная службы передачи факсимильных сообщений с помощью электронной почты Internet)
- 2531 — Content Feature Schema for Internet Fax (Особенности структуры содержимого для службы передачи факсимильных сообщений в сети Internet)
- 2306 — Tag Image File Format (TIFF) — F Profile for Facsimile (Файловый формат тэга для изображений (TIFF) — F-профиль для факсимильной передачи)
- 2305 — A Simple Mode of Facsimile Using Internet Mail (Простой режим передачи факсимильных сообщений с помощью электронной почты Internet)

- 2304 — Minimal FAX address format in Internet Mail (Минимальный формат адреса факса при кодировании в адрес электронной почты Internet)
- 2303 — Minimal PSTN address format in Internet Mail (Минимальный формат адреса номера телефона при кодировании в адрес электронной почты Internet)
- 2301 — File Format for Internet Fax (Формат файла для службы передачи факсимильных сообщений по сети Internet)
- 2159 — A MIME Body Part for FAX (Часть тела MIME-сообщения, предназначенная для передачи факсимильных сообщений )
- 2083 — PNG (Portable Network Graphics) Specification Version 1.0 (Спецификация сетевого переносимого формата графики, версия 1.0)
- 1529, 1528, 1486 — Principles of Operation for the TPC.INT Subdomain: Remote Printing — Administrative Policies (Принципы функционирования поддомена TPC.INT: удаленная печать — административные правила)
- 1314 — A File Format for the Exchange of Images in the Internet (Форматы файлов для обмена изображениями в сети Internet)
- 809 — UCL facsimile system (Система передачи факсимильных сообщений факультета информатики университетского колледжа в Лондоне)
- 804 — CCITT draft recommendation T.4 (Предварительные рекомендации для Т.4)
- 803 — Dacom 450/500 facsimile data transcoding (Преобразование факсимильных данных Dacom 450/500)
- 798 — Decoding facsimile data from the Rapicom 450 (Декодирование факсимильных данных из Rapicom 450)
- 797 — Format for Bitmap files (Форматы файлов растровых изображений)
- 769 — Rapicom 450 facsimile file format (Формат файла факсимильного изображения Rapicom 450)

#### **6g. Графика и оконные системы**

- 1198 — FYI on the X window system (Общая информация по системе X window)
- 1013 — X Window System Protocol, version 11: Alpha update April 1987 (Протокол системы X window, версия 11: первоначальное обновление — апрель 1987 года)
- 965 — Format for a graphical communication protocol (Форматы данных для протокола передачи графической информации)
- 553 — Draft design for a text/graphics protocol (Предварительный проект протокола передачи текстовой и графической информации)
- 493 — Graphics Protocol (Протокол передачи графической информации)
- 401 — Conversion of NGP-0 Coordinates to Device Specific Coordinates (Преобразование координат из NGP-0 в зависимые от устройства)
- 398 — ICP Sockets (Сокеты ICP)
- 387 — Some experiences in implementing Network Graphics Protocol Level 0 (Некоторый опыт в реализации сетевого протокола передачи графической информации нулевого уровня)
- 351 — Graphics information form for the ARPANET graphics resources notebook (Графическая информация, полученная из блокнота графических ресурсов ARPANET)
- 336 — Level 0 Graphic Input Protocol (Протокол ввода графической информации нулевого уровня)
- 296 — DS-1 display system (Система отображения DS-1)
- 292 — Graphics Protocol: Level 0 only (Графический протокол: только нулевой уровень)
- 285 — Network graphics (Сетевая графическая информация)

- 268 — Graphics facilities information (Особенности сетевой графической информации)
- 199 — Suggestions for a network data-tablet graphics protocol (Предложения по сетевому протоколу ввода графической информации с планшета)
- 192 — Some factors which a Network Graphics Protocol must consider (Ряд факторов, которые должны быть учтены в сетевом протоколе передачи графической информации)
- 191 — Graphics implementation and conceptualization at Augmentation Research Center (Реализация графики на концептуальном уровне в научно-исследовательском центре ARC)
- 186 — Network graphics loader (Сетевой графический загрузчик)
- 184 — Proposed graphic display modes (Предложенные графические режимы работы дисплея)
- 181, 177 — Modifications to RFC 177 (Изменения в документе RFC 177)
- 178 — Network graphic attention handling (Обработка графических сигналов "Внимание")
- 125, 86 — Response to RFC 86: Proposal for Network Standard Format for a Graphics Data Stream (Ответ на документ RFC 86: предложенный сетевой стандарт формата для потоковой передачи графических данных)
- 94 — Some thoughts on Network Graphics (Несколько мыслей по поводу сетевой графики)

#### **6h. Управление данными**

- 304 — Data management system proposal for the ARPA network (Система управления данными для сети ARPA)
- 195 — Data computers-data descriptions and access language (Описание компьютерных данных и языка доступа)
- 194 — Data Reconfiguration Service: Compiler/Interpreter Implementation Notes (Служба реконфигурации данных: замечания по поводу реализации компилятора/интерпретатора)
- 166 — Data Reconfiguration Service: An implementation specification (Служба реконфигурации данных: техническое задание на реализацию)
- 144 — Data sharing on computer networks (Совместное использование данных в компьютерных сетях)
- 138 — Status report on proposed Data Reconfiguration Service (Отчет о положении дел в предложенной службе реконфигурации данных)
- 83 — Language-machine for data reconfiguration (Языковой механизм для реконфигурации данных)

#### **6i. Дистанционный ввод заданий (NETRJE, NETRJS)**

- 740, 599, 589, 325, 189, 88 — NETRJS Protocol (Протокол NETRJS)
- 725 — RJE protocol for a resource sharing network (Протокол дистанционного ввода заданий для сети коллективного использования ресурсов)
- 499 — Harvard's network RJE (Протокол RJE для сети Гарвардского университета)
- 490 — Surrogate RJS for UCLA-CCN (Суррогатный протокол RJS для сети Калифорнийского университета в Лос-Анджелесе)
- 477, 436 — Remote Job Service at UCSB (Служба дистанционного ввода заданий Калифорнийского университета в Санта-Барбре)
- 407 — Remote Job Entry Protocol (Протокол дистанционного ввода заданий)
- 368 — Comments on "Proposed Remote Job Entry Protocol" (Комментарии по поводу предложенного протокола дистанционного ввода заданий)

- 360 — Proposed Remote Job Entry Protocol (Предложенный протокол дистанционного ввода заданий)
- 338 — EBCDIC/ASCII Mapping for Network RJE (Преобразование кодировок EBCDIC/ASCII для сетевого дистанционного ввода заданий)
- 307 — Using network Remote Job Entry (Использование сетевого дистанционного ввода заданий)
- 283 — NETRJT: Remote Job Service Protocol for TIPS (NETRJT: протокол службы дистанционного ввода заданий для TIPS)
- 105 — Network Specifications for Remote Job Entry and Remote Job Output Retrieval at UCSB (Сетевые спецификации для дистанционного ввода заданий и дистанционного получения результатов выполнения задания в Калифорнийском университете в Санта-Барбаре)

#### **6j. Дистанционный вызов процедур (RPC)**

- 2695 — Authentication Mechanisms for ONC RPC (Механизм аутентификации для дистанционного вызова процедур при открытой сетевой обработке)
- 2203 — RPCSEC\_GSS Protocol Specification (Спецификация протокола RPCSEC\_GSS)
- 1833 — Binding Protocols for ONC RPC Version 2 (Протоколы привязки для дистанционного вызова процедур при открытой сетевой обработке, версия 2)
- 1831 — RPC: Remote Procedure Call Protocol Specification Version 2 (Спецификация протокола дистанционного вызова процедур, версия 2)
- 1057 — RPC: Remote Procedure Call Protocol specification: Version 2 (Спецификация протокола дистанционного вызова процедур, версия 2)
- 1050 — RPC: Remote Procedure Call Protocol specification (Спецификация протокола дистанционного вызова процедур)

#### **6k. Время и дата (NTP)**

- 2030, 1769, 1361 — Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI (Простой протокол синхронизации сетевого времени версии 4 для IPv4, IPv6 и OSI)
- 1708 — NTP PICS PROFORMA — For the Network Time Protocol Version 3 (NTP PICS PROFORMA — для протокола синхронизации сетевого времени версии 3)
- 1589 — A Kernel Model for Precision Timekeeping (Модель на основе ядра для точной синхронизации времени)
- 1305, 1119, 1059 — Network Time Protocol (Version 3) Specification, Implementation (Спецификация и реализация протокола синхронизации сетевого времени версии 3)
- 1165 — Network Time Protocol (NTP) over the OSI Remote Operations Service (Протокол синхронизации сетевого времени (NTP) на основе службы удаленной работы OSI)
- 1129 — Internet time synchronization: The Network Time Protocol (Синхронизация времени в сети Internet: протокол NTP)
- 1128 — Measured performance of the Network Time Protocol in the Internet system (Измерение технических характеристик протокола NTP в сети Internet)
- 958, 957, 956 — Network Time Protocol (NTP) (Протокол синхронизации сетевого времени (NTP))
- 868 — Time Protocol (Протокол синхронизации времени)
- 867 — Daytime Protocol (Протокол синхронизации времени и даты)
- 778 — DCNET Internet Clock Service (Служба времени Internet DCNET)
- 738 — Time server (Сервер службы времени)

- 685 — Response time in cross network debugging (Время ответа при перекрестной сетевой отладке)
- 34 — Some Brief Preliminary Notes on the Augmentation Research Center Clock (Несколько коротких предварительных замечаний по поводу службы времени научно-исследовательского центра ARC)
- 32 — Connecting M.I.T (Подключение к М.И.Т)
- 28 — Time Standards (Стандарты представления времени)

## **6I. Представление и отображение (XDR, кодировка символов, HTML, XML)**

- 2706 — ECML v1: Field Names for E-Commerce (Язык моделирования систем электронной коммерции, версия 1: имена полей, предназначенных для электронной коммерции)
- 2659 — Security Extensions For HTML (Расширения безопасности для языка HTML)
- 2482 — Language Tagging in Unicode Plain Text (Языковые дескрипторы в простом текстовом файле в формате Unicode)
- 2413 — Dublin Core Metadata for Resource Discovery (Дублинские базовые метаданные для обнаружения ресурсов)
- 2376 — XML Media Types (Типы носителя в языке XML)
- 2346 — Making Postscript and PDF International (Локализация файлов в формате Postscript и PDF)
- 2319 — Ukrainian Character Set KOI8-U (Украинский набор символов KOI8-U)
- 2279, 2044 — UTF-8, a transformation format of ISO 10646 (UTF-8 — измененный формат ISO 10646)
- 2237 — Japanese Character Encoding for Internet Messages (Кодировка японских символов в сообщениях Internet)
- 2183 — Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field (Смежная информация о представлении в сообщениях Internet: поле заголовка Content-Disposition)
- 2070 — Internationalization of the Hypertext Markup Language (Локализация языка гипертекстовой разметки)
- 1980 — A Proposed Extension to HTML : Client-Side Image Maps (Предложенное расширение языка HTML: клиентские карты изображений)
- 1952 — GZIP file format specification version 4.3 (Спецификация формата файла утилиты GZIP версии 4.3)
- 1951 — DEFLATE Compressed Data Format Specification version 1.3 (Спецификация формата сжатия данных по методу DEFLATE версии 1.3)
- 1950 — ZLIB Compressed Data Format Specification version 3.3 (Спецификация формата сжатия данных по методу ZLIB версии 3.3)
- 1947 — Greek Character Encoding for Electronic Mail Messages (Кодировка греческих символов в сообщениях электронной почты)
- 1942 — HTML Tables (Таблицы HTML)
- 1922 — Chinese Character Encoding for Internet Messages (Кодировка китайских символов в сообщениях электронной почты)
- 1874 — SGML Media Types (Типы носителя в языке SGML)
- 1867 — Form-based File Upload in HTML (Выгрузка файлов с помощью форм в языке HTML)
- 1866 — Hypertext Markup Language — 2.0 (Язык гипертекстовой разметки, версия 2.0)
- 1843 — HZ — A Data Format for Exchanging Files of Arbitrarily Mixed Chinese and ASCII characters (HZ — формат данных для обмена файлами, содержащих смесь произвольного количества китайский и ASCII-символов)

- 1842 — ASCII Printable Characters-Based Chinese Character Encoding for Internet Messages (Кодировка китайских символов на основе печатных ASCII-символов в сообщениях Internet)
- 1832 — XDR: External Data Representation Standard (XDR: стандарт представления внешних данных)
- 1815 — Character Sets ISO-10646 and ISO-10646-J-1 (Наборы символов ISO-10646 и ISO-10646-J-1)
- 1766 — Tags for the Identification of Languages (Дескрипторы для идентификации языка)
- 1557 — Korean Character Encoding for Internet Messages (Кодировка корейских символов в сообщениях электронной почты)
- 1555 — Hebrew Character Encoding for Internet Messages (Кодировка древнееврейских (иврит) символов в сообщениях электронной почты)
- 1554 — ISO-2022-JP-2: Multilingual Extension of ISO-2022-JP (ISO-2022-JP-2: многоязыковое расширение стандарта ISO-2022-JP)
- 1489 — Registration of a Cyrillic Character Set (Регистрация кириллического набора символов)
- 1468 — Japanese Character Encoding for Internet Messages (Кодировка японских символов в сообщениях электронной почты)
- 1456 — Conventions for Encoding the Vietnamese Language VISCII: Vietnamese Standard Code for Information Interchange VIQR: Vietnamese Quoted-Readable Specification (Соглашения по кодированию вьетнамского языка; VISCII: вьетнамский стандартный код обмена информацией; VIQR: спецификация представления вьетнамских символов с помощью 7-битового ASCII-кода)
- 1278 — A string encoding of Presentation Address (Строковое кодирование представительского адреса)
- 1197 — Using ODA for translating multimedia information (Использование ODA для преобразования мультимедийной информации)
- 1014 — XDR: External Data Representation standard (XDR: стандарт представления внешних данных)
- 1003 — Issues in defining an equations representation standard (Вопросы, возникающие при определении стандартов отображения математических формул)

#### **6м. Управление сетью (SNMP, CMOT, RMON)**

- 2593 — Script MIB Extensibility Protocol Version 1.0 (Расширяемость протокола Script MIB, версии 1.0)
- 2580, 1904, 1444 — Conformance Statements for SMIv2 (Операторы соответствия для SMIv2)
- 2579, 1903, 1443 — Textual Conventions for SMIv2 (Текстовые соглашения для SMIv2)
- 2578, 1902, 1442 — Structure of Management Information Version 2 (SMIv2) (Структура управляющей информации, версия 2)
- 2575, 2275, 2265 — View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP) (Модель управления доступом, основанная на представлениях, для простого протокола сетевого управления)
- 2574, 2274, 2264 — User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3) (Модель безопасности, использующая идентификацию пользователя как субъекта доступа, для простого протокола сетевого управления версии 3)
- 2573, 2273, 2263 — SNMP Applications (SNMP-приложения)

- 2572, 2272, 2262 — Message Processing and Dispatching for the Simple Network Management Protocol (SNMP) (Обработка сообщений и диспетчеризация для простого протокола сетевого управления)
- 2571, 2271, 2261 — An Architecture for Describing SNMP Management Frameworks (Общая структура описания инфраструктуры управления протокола SNMP)
- 2570 — Introduction to Version 3 of the Internet-standard Network Management Framework (Введение в стандартную инфраструктуру сетевого управления в Internet, версия 3)
- 2493 — Textual Conventions for MIB Modules Using Performance History Based on 15 Minute Intervals (Текстовые соглашения для модулей MIB, основанные на анализе производительности, измеренной с 15-минутным интервалом)
- 2438 — Advancement of MIB specifications on the IETF Standards Track (Продвижение спецификации MIB при регистрации стандартов IETF)
- 2257 — Agent Extensibility (AgentX) Protocol Version 1 (Расширяемость протокола агента, версии 1)
- 2107 — Ascend Tunnel Management Protocol — ATMP (Протокол управления туннелированием фирмы Ascend)
- 2089 — V2ToV1 Mapping SNMPv2 onto SNMPv1 within a bilingual SNMP agent (Преобразование данных протокола SNMPv2 в SNMPv1 с помощью двухязыкового SNMP-агента)
- 2039 — Applicability of Standards Track MIBs to Management of World Wide Web Servers (Применение системы регистрации стандартов в MIB для управления серверами World Wide Web)
- 1910 — User-based Security Model for SNMPv2 (Модель безопасности, использующая идентификацию пользователя как субъекта доступа, для протокола SNMPv2)
- 1909 — An Administrative Infrastructure for SNMPv2 (Административная инфраструктура для протокола SNMPv2)
- 1908, 1452 — Coexistence between Version 1 and Version 2 of the Internet-standard Network Management Framework (Совместное использование стандартной инфраструктуры сетевого управления в Internet, версий 1 и 2)
- 1906, 1449 — Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2) (Привязки к транспортным протоколам для простого протокола сетевого управления версии 2)
- 1905, 1448 — Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2) (Функции простого протокола сетевого управления версии 2)
- 1901 — Introduction to Community-based SNMPv2 (Общие сведения о административной инфраструктуре, основанной на сообществе протокола SNMPv2)
- 1856 — The Opstat Client-Server Model for Statistics Retrieval (Модель клиент-серверной операционной статистики для статистического поиска)
- 1592, 1228 — Simple Network Management Protocol Distributed Protocol Interface Version 2.0 (Протокол распределенного интерфейса протокола SNMP)
- 1503 — Algorithms for Automating Administration in SNMPv2 Managers (Алгоритмы автоматизации административных задач, используемые в приложениях SNMPv2)
- 1446 — Security Protocols for version 2 of the Simple Network Management Protocol (SNMPv2) (Протоколы безопасности для протокола SNMPv2)
- 1445 — Administrative Model for version 2 of the Simple Network Management Protocol (SNMPv2) (Административная модель для протокола SNMPv2)

- 1441 — Introduction to version 2 of the Internet-standard Network Management Framework (Общие сведения о стандартной инфраструктуре сетевого управления в Internet, версия 2)
- 1420, 1298 — SNMP over IPX (Протокол SNMP в сети на основе протокола IPX)
- 1419 — SNMP over AppleTalk (Протокол SNMP в сети на основе протокола AppleTalk)
- 1418, 1283, 1161 — SNMP over OSI (Протокол SNMP в сети на основе протокола OSI)
- 1369 — Implementation Notes and Experience for the Internet Ethernet MIB (Замечания по поводу реализации и опыта использования базы управляющей информации для сети Ethernet)
- 1352 — SNMP Security Protocols (Протоколы безопасности для протокола SNMP)
- 1351 — SNMP Administrative Model (Административная модель для протокола SNMP)
- 1346 — Resource Allocation, Control, and Accounting for the Use of Network Resources (Распределение, управление и контроль использования сетевых ресурсов)
- 1303 — A Convention for Describing SNMP-based Agents (Соглашения по описанию агентов протокола SNMP)
- 1270 — SNMP Communications Services (Службы взаимодействия протокола SNMP)
- 1239 — Reassignment of experimental MIBs to standard MIBs (Переназначение экспериментальной MIB в стандартную MIB)
- 1224 — Techniques for managing asynchronously generated alerts (Методики обработки асинхронно генерируемых сигналов “Внимание”)
- 1215 — Convention for defining traps for use with the SNMP (Соглашения по определению прерывания для использования в протоколе SNMP)
- 1212 — Concise MIB definitions (Краткие определения базы управляющей информации)
- 1189, 1095 — Common Management Information Services and Protocols for the Internet (CMOT and CMIP) (Общие службы управляющей информации и протоколы для Internet)
- 1187 — Bulk Table Retrieval with the SNMP (Массовая выборка информации из таблиц с помощью протокола SNMP)
- 1157, 1098, 1067 — Simple Network Management Protocol (SNMP) (Простой протокол сетевого управления)
- 1155, 1065 — Structure and identification of management information for TCP/IP-based internets (Структура и опознавание управляющей информации для объединенных сетей на основе протокола TCP/IP)
- 1109 — Report of the second Ad Hoc Network Management Review Group (Отчет, сделанный второй специальной экспертной группой сетевого управления)
- 1089 — SNMP over Ethernet (Протокол SNMP в сети Ethernet)
- 1076 — HEMS monitoring and control language (Язык мониторинга и управления HEMS)
- 1028 — Simple Gateway Monitoring Protocol (Простой протокол мониторинга шлюза)
- 1024 — HEMS variable definitions (Определение переменных языка HEMS)
- 1023 — HEMS monitoring and control language (Язык мониторинга и управления HEMS)
- 1022 — High-level Entity Management Protocol (HEMP) (Протокол управления объектами высокого уровня)
- 1021 — High-level Entity Management System (HEMS) (Система управления объектами высокого уровня)

## **6п. Определения базы управляющей информации (MIB)**

- 2720, 2064 — Traffic Flow Measurement: Meter MIB (Измерение величины потока трафика: счетчик MIB)
- 2677 — Definitions of Managed Objects for the NBMA Next Hop Resolution Protocol (NHRP) (Определения контролируемых объектов для протокола определения ближайшего адресата в нешироковещательных сетях с множественным доступом)
- 2674 — Definitions of Managed Objects for Bridges with Traffic Classes, Multicast Filtering and Virtual LAN Extensions (Определения контролируемых объектов для мостов с классами трафика, многоадресатной фильтрацией и расширениями виртуальных локальных сетей)
- 2670 — Radio Frequency (RF) Interface Management Information Base for MCNS/DOCSIS compliant RF interfaces (Радиочастотный интерфейс базы управляющей информации для совместимых с MCNS/DOCSIS интерфейсов)
- 2669 — DOCSIS Cable Device MIB Cable Device Management Information Base for DOCSIS compliant Cable Modems and Cable Modem Termination Systems (База управляющей информации для кабельных модемов, совместимых с DOCSIS, и оконечной системе кабельных модемов)
- 2668, 2239 — Definitions of Managed Objects for IEEE 802.3 Medium Attachment Units (MAUs) (Определения контролируемых объектов для устройств подключения к среде IEEE 802.3)
- 2667 — IP Tunnel MIB (База управляющей информации для IP-туннеля)
- 2666 — Definitions of Object Identifiers for Identifying Ethernet Chip Sets (Определения идентификаторов объектов для наборов микросхем Ethernet)
- 2665, 2358, 1650 — Definitions of Managed Objects for the Ethernet-like Interface Types (Определения контролируемых объектов для типов интерфейсов, подобных Ethernet)
- 2662 — Definitions of Managed Objects for the ADSL Lines (Определения контролируемых объектов для каналов связи ADSL)
- 2621 — RADIUS Accounting Server MIB (Сервер учета MIB RADIUS)
- 2620 — RADIUS Accounting Client MIB (Клиент учета MIB RADIUS)
- 2619 — RADIUS Authentication Server MIB (Сервер аутентификации MIB RADIUS)
- 2618 — RADIUS Authentication Client MIB (Клиент аутентификации MIB RADIUS)
- 2613 — Remote Network Monitoring MIB Extensions for Switched Networks Version 1.0 (Расширения MIB для дистанционного мониторинга сети версии 1.0)
- 2605, 1567 — Directory Server Monitoring MIB (MIB для мониторинга сервера каталогов)
- 2594 — Definitions of Managed Objects for WWW Services (Определения контролируемых объектов для служб WWW)
- 2592 — Definitions of Managed Objects for the Delegation of Management Script (Определения контролируемых объектов для делегирования сценариев управления)
- 2591 — Definitions of Managed Objects for Scheduling Management Operations (Определения контролируемых объектов для управления расписанием)
- 2584 — Definitions of Managed Objects for APPN/HPR in IP Networks (Определения контролируемых объектов для высокопроизводительных устройств маршрутизации в IP-сетях)
- 2564 — Application Management MIB (MIB для управления приложениями)
- 2562 — Definitions of Protocol and Managed Objects for TN3270E Response Time Collection Using SMIv2 (TN3270E-RT-MIB) (Определения протокола и

- контролируемых объектов для сбора информации о времени ответа TN3270E с использованием SMIv2)
- 2561 — Base Definitions of Managed Objects for TN3270E Using SMIv2 (Основные определения контролируемых объектов для TN3270E с использованием SMIv2)
- 2558, 1595 — Definitions of Managed Objects for the SONET/SDH Interface Type (Определения контролируемых объектов для типов интерфейсов SONET/SDH)
- 2515, 1695 — Definitions of Managed Objects for ATM Management (Определения контролируемых объектов для управления сетью ATM)
- 2514 — Definitions of Textual Conventions and OBJECT-IDENTITIES for ATM Management (Определения текстовых соглашений и ОБЪЕКТ-IDENTITIES для управления сетью ATM)
- 2513 — Managed Objects for Controlling the Collection and Storage of Accounting Information for Connection-Oriented Networks (Управляемые объекты для управления процессом сбора и хранения учетной информации для сетей, ориентированных на установку соединения)
- 2512 — Accounting Information for ATM Networks (Учетная информация для сетей ATM)
- 2496, 1407, 1233 — Definitions of Managed Object for the DS3/E3 Interface Type (Определения контролируемых объектов для типов интерфейсов DS3/E3)
- 2495, 1406, 1232 — Definitions of Managed Objects for the DS1, E1, DS2 and E2 Interface Types (Определения контролируемых объектов для типов интерфейсов DS1, E1, DS2 и E2)
- 2494 — Definitions of Managed Objects for the DS0 and DS0 Bundle Interface Type (Определения контролируемых объектов для типов интерфейсов DS0 и связанных DS0)
- 2457 — Definitions of Managed Objects for Extended Border Node (Определения контролируемых объектов для расширенных граничных узлов)
- 2456 — Definitions of Managed Objects for APPN TRAPS (Определения контролируемых объектов для APPN TRAPS)
- 2455, 2155 — Definitions of Managed Objects for APPN (Определения контролируемых объектов для усовершенствованной одноранговой сети)
- 2417, 2366 — Definitions of Managed Objects for Multicast over UNI 3.0/3.1 based ATM Networks (Определения контролируемых объектов для широковещательных сетей ATM на основе UNI 3.0/3.1)
- 2320 — Definitions of Managed Objects for Classical IP and ARP Over ATM Using SMIv2 (IPOA-MIB) (Определения контролируемых объектов для классического протокола IP и ARP в сети ATM при использовании SMIv2 (IPOA-MIB))
- 2287 — Definitions of System-Level Managed Objects for Applications (Определения контролируемых объектов системного уровня для приложений)
- 2266 — Definitions of Managed Objects for IEEE 802.12 Repeater Devices (Определения контролируемых объектов для повторителей стандарта IEEE 802.12)
- 2249, 1566 — Mail Monitoring MIB (База управляющей информации для мониторинга электронной почты)
- 2248, 1565 — Network Services Monitoring MIB (База управляющей информации для мониторинга сетевых служб)

- 2238 — Definitions of Managed Objects for HPR using SMIv2 (Определения контролируемых объектов для высокопроизводительных устройств маршрутизации при использовании SMIv2)
- 2233, 1573, 1229 — The Interfaces Group MIB using SMIv2 (База управляющей информации для группы интерфейсов при использовании SMIv2)
- 2232 — Definitions of Managed Objects for DLUR using SMIv2 (Определения контролируемых объектов для DLUR при использовании SMIv2)
- 2214 — Integrated Services Management Information Base. Guaranteed Service Extensions using SMIv2 (База управляющей информации интегрированных служб. Расширения гарантированных служб при использовании SMIv2)
- 2213 — Integrated Services Management Information Base using SMIv2 (База управляющей информации интегрированных служб при использовании SMIv2)
- 2128 — Dial Control Management Information Base using SMIv2 (База управляющей информации коммутируемых каналов при использовании SMIv2)
- 2127 — ISDN Management Information Base using SMIv2 (База управляющей информации ISDN при использовании SMIv2)
- 2115, 1315 — Management Information Base for Frame Relay DTEs Using SMIv2 (База управляющей информации для терминального оборудования сети Frame Relay при использовании SMIv2)
- 2108, 1516, 1368 — Definitions of Managed Objects for IEEE 802.3 Repeater Devices using SMIv2 (Определения контролируемых объектов для повторителей стандарта IEEE 802.3 при использовании SMIv2)
- 2096, 1354 — IP Forwarding Table MIB (База управляющей информации для таблицы пересылки протокола IP)
- 2074 — Remote Network Monitoring MIB Protocol Identifiers (MIB идентификаторов протокола для дистанционного мониторинга сети)
- 2063 — Traffic Flow Measurement: Architecture (Структура системы измерения потока трафика)
- 2051 — Definitions of Managed Objects for APPC using SMIv2 (Определения контролируемых объектов для усовершенствованного интерфейса связи между программами при использовании SMIv2)
- 2037 — Entity MIB using SMIv2 (Объекты MIB при использовании SMIv2)
- 2024 — Definitions of Managed Objects for Data Link Switching using SMIv2 (Определения контролируемых объектов для коммутируемых каналов связи при использовании SMIv2)
- 2021 — Remote Network Monitoring Management Information Base Version 2 using SMIv2 (MIB версии 2 для дистанционного мониторинга сети при использовании SMIv2)
- 2020 — IEEE 802.12 Interface MIB (MIB для интерфейса IEEE 802.12)
- 2013 — SNMPv2 Management Information Base for the User Datagram Protocol using SMIv2 (MIB протокола SNMPv2 для протокола UDP при использовании SMIv2)
- 2012 — SNMPv2 Management Information Base for the Transmission Control Protocol using SMIv2 (MIB протокола SNMPv2 для протокола TCP при использовании SMIv2)
- 2011 — SNMPv2 Management Information Base for the Internet Protocol using SMIv2 (MIB протокола SNMPv2 для протокола IP при использовании SMIv2)
- 2006 — The Definitions of Managed Objects for IP Mobility Support using SMIv2 (Определения контролируемых объектов для поддержки мобильного протокола IP при использовании SMIv2)

- 1907, 1450 — Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2) (База управляющей информации для протокола SNMPv2)
- 1850, 1253, 1252, 1248 — OSPF Version 2 Management Information Base (База управляющей информации для протокола OSPF версии 2)
- 1792 — TCP/IPX Connection MIB Specification (Спецификация МИВ для соединений TCP/IPX)
- 1759 — Printer MIB (База управляющей информации для принтера)
- 1757, 1271 — Remote Network Monitoring Management Information Base (База управляющей информации для дистанционного мониторинга сети)
- 1749 — IEEE 802.5 Station Source Routing MIB using SMIv2 (База управляющей информации для маршрутизации от источника узлов сети IEEE 802.5)
- 1748, 1743, 1231 — IEEE 802.5 MIB using SMIv2 (База управляющей информации для сети IEEE 802.5 при использовании SMIv2)
- 1747 — Definitions of Managed Objects for SNA Data Link Control (SDLC) using SMIv2 (Определения контролируемых объектов для протокола управления передачей данных SNA при использовании SMIv2)
- 1742, 1243 — AppleTalk Management Information Base II (База управляющей информации II протокола AppleTalk)
- 1724, 1389 — RIP Version 2 MIB Extension (Расширения базы управляющей информации для протокола RIP версии 2)
- 1697 — Relational Database Management System (RDBMS) Management Information Base (MIB) using SMIv2 (База управляющей информации для систем управления реляционными базами данных при использовании SMIv2)
- 1696 — Modem Management Information Base (MIB) using SMIv2 (База управляющей информации для модема при использовании SMIv2)
- 1694, 1304 — Definitions of Managed Objects for SMDS Interfaces using SMIv2 (Определения контролируемых объектов для интерфейсов высокоскоростной коммутируемой службы передачи данных)
- 1666, 1665 — Definitions of Managed Objects for SNA NAUs using SMIv2 (Определения контролируемых объектов для адресуемых сетевых элементов системной сетевой архитектуры при использовании SMIv2)
- 1660, 1318 — Definitions of Managed Objects for Parallel-printer-like Hardware Devices using SMIv2 (Определения контролируемых объектов для устройств, наподобие принтеров с параллельным интерфейсом при использовании SMIv2)
- 1659, 1317 — Definitions of Managed Objects for RS-232-like Hardware Devices using SMIv2 (Определения контролируемых объектов для устройств, оснащенных интерфейсом RS-232 при использовании SMIv2)
- 1658, 1316 — Definitions of Managed Objects for Character Stream Devices using SMIv2 (Определения контролируемых объектов для устройств потоковой передачи символов при использовании SMIv2)
- 1657 — Definitions of Managed Objects for the Fourth Version of the Border Gateway Protocol (BGP-4) using SMIv2 (Определения контролируемых объектов для четвертой версии протокола граничного шлюза (BGP-4) при использовании SMIv2)
- 1643, 1623, 1398, 1284 — Definitions of Managed Objects for the Ethernet-like Interface Types (Определения контролируемых объектов для интерфейсов, наподобие Ethernet)
- 1628 — UPS Management Information Base (База управляющей информации для источника бесперебойного питания)
- 1612 — DNS Resolver MIB Extensions (Расширения базы управляющей информации для распознавателей DNS)

- 1611 — DNS Server MIB Extensions (Расширения базы управляющей информации для DNS-серверов)
- 1604, 1596 — Definitions of Managed Objects for Frame Relay Service (Определения контролируемых объектов для служб сети Frame Relay)
- 1593 — SNA APPN Node MIB (База управляющей информации для узлов усовершенствованной одноранговой сети SNA)
- 1559, 1289 — DECnet Phase IV MIB Extensions (Расширения базы управляющей информации для 4-й фазы протокола DECnet)
- 1525, 1493, 1286 — Definitions of Managed Objects for Source Routing Bridges (Определения контролируемых объектов для мостов, поддерживающих маршрутизацию от источника)
- 1515 — Definitions of Managed Objects for IEEE 802.3 Medium Attachment Units (MAUs) (Определения контролируемых объектов для устройств подключения к среде IEEE 802.3)
- 1514 — Host Resources MIB (База управляющей информации для ресурсов узлов)
- 1513 — Token Ring Extensions to the Remote Network Monitoring MIB (Расширения сети Token Ring базы управляющей информации для дистанционного мониторинга сети)
- 1512, 1285 — FDDI Management Information Base (База управляющей информации для FDDI)
- 1474 — The Definitions of Managed Objects for the Bridge Network Control Protocol of the Point-to-Point Protocol (Определения контролируемых объектов для протокола управления мостовой схемой протокола PPP)
- 1473 — The Definitions of Managed Objects for the IP Network Control Protocol of the Point-to-Point Protocol (Определения контролируемых объектов для протокола управления IP-сетью протокола PPP)
- 1472 — The Definitions of Managed Objects for the Security Protocols of the Point-to-Point Protocol (Определения контролируемых объектов для протоколов безопасности протокола PPP)
- 1471 — The Definitions of Managed Objects for the Link Control Protocol of the Point-to-Point Protocol (Определения контролируемых объектов для протокола управления каналом связи протокола PPP)
- 1461 — SNMP MIB extension for MultiProtocol Interconnect over X.25 (Расширения базы управляющей информации протокола SNMP для многопротокольной связи с помощью стандарта X.25)
- 1451 — Manager-to-Manager Management Information Base (База управляющей информации для протокола обмена сообщениями между агентами)
- 1447 — Party MIB for version 2 of the Simple Network Management Protocol (SNMPv2) (Часть базы управляющей информации для простого протокола сетевого управления версии 2 (SNMPv2))
- 1414 — Identification MIB (Идентификация базы управляющей информации)
- 1382 — SNMP MIB Extension for the X.25 Packet Layer (Расширения базы управляющей информации протокола SNMP для уровня пакетов протокола X.25)
- 1381 — SNMP MIB Extension for X.25 LAPB (Расширения базы управляющей информации протокола SNMP для сбалансированного протокола доступа к каналу связи)
- 1353 — Definitions of Managed Objects for Administration of SNMP Parties (Определения контролируемых объектов для управления группами протокола SNMP)
- 1269 — Definitions of Managed Objects for the Border Gateway Protocol: Version 3 (Определения контролируемых объектов для третьей версии протокола граничного шлюза (BGP-8))

- 1230 — IEEE 802.4 Token Bus MIB (База управляющей информации для маркерной шины IEEE 802.4)
- 1227 — SNMP MUX protocol and MIB (Протокол мультиплексирования SNMP и база управляющей информации)
- 1214 — OSI internet management: Management Information Base (База управляющей информации для протоколов сетевого управления OSI)
- 1213, 1158, 1156, 1066 — Management Information Base for Network Management of TCP/IP-based internets: MIB-II (База управляющей информации для протоколов сетевого управления на основе TCP/IP: MIB-II)

#### **6o. Службы каталогов (X.500, LDAP, белые страницы)**

- 2714 — Schema for Representing CORBA Object References in an LDAP Directory (Проект представления ссылок на объекты CORBA в каталоге LDAP)
- 2713 — Schema for Representing Java(tm) Objects in an LDAP Directory (Проект представления ссылок на объекты Java в каталоге LDAP)
- 2696 — LDAP Control Extension for Simple Paged Results Manipulation (Расширение механизма управления LDAP для простого постраничного вывода результатов)
- 2657 — LDAPv2 Client vs the Index Mesh (Клиент LDAPv2 и сервер Index Mesh)
- 2649 — An LDAP Control and Schema for Holding Operation Signatures (Механизм управления LDAP и проект сохранения действующих подписей)
- 2596 — Use of Language Codes in LDAP (Использование кодов языков в LDAP)
- 2589 — Lightweight Directory Access Protocol (v3): Extensions for Dynamic Directory Services (Облегченный протокол службы доступа к каталогам версии 3: расширения для динамических служб каталогов)
- 2587 — Internet X.509 Public Key Infrastructure LDAPv2 Schema (Проект инфраструктуры открытых ключей протокола LDAPv2 для Internet на основе сертификатов X.509)
- 2585 — Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP (Инфраструктура открытых ключей для Internet на основе сертификатов X.509: операционные протоколы FTP и HTTP)
- 2560 — X.509 Internet Public Key Infrastructure Online Certificate Status Protocol — OCSP (Инфраструктура открытых ключей для Internet на основе сертификатов X.509: протокол оперативной проверки состояния сертификатов OCSP)
- 2559 — Internet X.509 Public Key Infrastructure Operational Protocols — LDAPv2 (Инфраструктура открытых ключей для Internet на основе сертификатов X.509: операционные протоколы — LDAPv2)
- 2528 — Internet X.509 Public Key Infrastructure. Representation of Key Exchange Algorithm (KEA) Keys in Internet X.509 Public Key Infrastructure Certificates (Инфраструктура открытых ключей для Internet на основе сертификатов X.509: представление алгоритма обмена ключами (KEA) в инфраструктуре открытых ключей для Internet на основе сертификатов X.509)
- 2527 — Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework (Инфраструктура открытых ключей для Internet на основе сертификатов X.509: политика сертификатов и основные правила сертификации)
- 2511 — Internet X.509 Certificate Request Message Format (Формат запроса сертификатов X.509 в Internet)
- 2510 — Internet X.509 Public Key Infrastructure Certificate Management Protocols (Инфраструктура открытых ключей для Internet на основе сертификатов X.509: протоколы управления сертификатами)

- 2459 — Internet X.509 Public Key Infrastructure Certificate and CRL Profile (Инфраструктура открытых ключей для Internet на основе сертификатов X.509: сертификаты и профиль списка аннулирования сертификатов)
- 2377 — Naming Plan for Internet Directory-Enabled Applications (Проект именования для приложений доступа к службе каталогов Internet)
- 2307 — An Approach for Using LDAP as a Network Information Service (Практика использования LDAP в качестве сетевой информационной службы)
- 2294, 1836 — Representing the O/R Address hierarchy in the X.500 Directory Information Tree (Представление иерархии адресов O/R в информационном дереве каталогов X.500)
- 2293, 1837 — Representing Tables and Subtrees in the X.500 Directory (Представление таблиц и поддеревьев в каталоге X.500)
- 2256 — A Summary of the X.500(96) User Schema for use with LDAPv3 (Краткая информация о пользовательских схемах X.500(96) для использования в протоколе LDAPv3)
- 2255 — The LDAP URL Format (Формат URL LDAP)
- 2254, 1960, 1558 — The String Representation of LDAP Search Filters (Строковые представления поисковых фильтров LDAP)
- 2253 — Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names (Облегченный протокол службы доступа к каталогам версии 3: представление различительных имен в виде строки формата UTF-8)
- 2252 — Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions (Облегченный протокол службы доступа к каталогам версии 3: определение синтаксиса атрибутов)
- 2251 — Lightweight Directory Access Protocol (v3) (Облегченный протокол службы доступа к каталогам версии 3)
- 2247 — Using Domains in LDAP/X.500 Distinguished Names (Использование доменов в различительных именах LDAP/X.500)
- 2218 — A Common Schema for the Internet White Pages Service (Общая схема использования, применяемая к службам белых страниц Internet)
- 2148 — Deployment of the Internet White Pages Service (Разворачивание службы белых страниц Internet)
- 2120 — Managing the X.500 Root Naming Context (Управление контекстом корневых имен X.500)
- 2116, 1632, 1292 — X.500 Implementations Catalog-96 (Реализация X.500 в каталоге-96)
- 2079 — Definition of an X.500 Attribute Type and an Object Class to Hold Uniform Resource Identifiers (URIs) (Определение типов атрибутов X.500 и классов объектов для хранения унифицированных идентификаторов ресурсов)
- 1959 — An LDAP URL Format (Формат URL LDAP)
- 1943 — Building an X.500 Directory Service in the US (Создание службы каталогов X.500 в США)
- 1823 — The LDAP Application Program Interface (Программный интерфейс приложений LDAP)
- 1804 — Schema Publishing in X.500 Directory (Схема публикации в каталоге X.500)
- 1803 — Recommendations for an X.500 Production Directory Service (Рекомендации по созданию службы каталогов X.500)
- 1802 — Introducing Project Long Bud: Internet Pilot Project for the Deployment of X.500 Directory. Information in Support of X.400 Routing (Опытный проект развертывания каталога X.500 в Internet. Информация в поддержку маршрутизации X.400)

- 1801 — MHS use of the X.500 Directory to support MHS Routing (Использование системы управления сообщениями каталога X.500 для поддержки MHS-маршрутизации)
- 1798 — Connection-less Lightweight X.500 Directory Access Protocol (Облегченный протокол службы доступа к каталогам X.500 не требующий установки соединения)
- 1781, 1484 — Using the OSI Directory to Achieve User Friendly Naming (Использование каталогов OSI для достижения удобной для пользователей системы имен)
- 1779, 1485 — A String Representation of Distinguished Names (Строковое представление различительных имен)
- 1778, 1488 — The String Representation of Standard Attribute Syntaxes (Строковое представление стандартного синтаксиса атрибутов)
- 1777, 1487 — Lightweight Directory Access Protocol (Облегченный протокол службы доступа к каталогам)
- 1684 — Introduction to White Pages Services based on X.500 (Общие сведения о службах белых страниц на основе X.500)
- 1617, 1384 — Naming and Structuring Guidelines for X.500 Directory Pilots (Инструкция по именованию и структурированию для администраторов каталога X.500)
- 1609 — Charting Networks in the X.500 Directory (Составление схем сетей в каталоге X.500)
- 1608 — Representing IP Information in the X.500 Directory (Представление информации протокола IP в каталоге X.500)
- 1564 — DSA Metrics (OSI-DS 34 (v3)) (Метрики DSA)
- 1562 — Naming Guidelines for the AARNet X.500 Directory Service (Руководство по именованию для службы каталогов X.500 сети AARNet)
- 1491 — A Survey of Advanced Usages of X.500 (Обзор прогрессивных способов использования X.500)
- 1431 — DUA Metrics (OSI-DS 33 (v2)) (Метрики DUA)
- 1430 — A Strategic Plan for Deploying an Internet X.500 Directory Service (Стратегический план развертывания службы каталогов X.500 в Internet)
- 1373 — Portable DUAs (Переносимые агенты пользователей каталогов)
- 1309 — Technical Overview of Directory Services Using the X.500 Protocol (Технический обзор служб каталогов, использующих протокол X.500)
- 1308 — Executive Introduction to Directory Services Using the X.500 Protocol (Вводный курс для супервизора по службам каталогов на основе протокола X.500)
- 1279 — X.500 and Domains (Протокол X.500 и домены)
- 1277 — Encoding Network Addresses to Support Operation over Non-OSI Lower Layers (Кодирование сетевого адреса для поддержки функционирования системы на основе нижних уровней, не относящихся к модели OSI)
- 1276 — Replication and Distributed Operations extensions to provide an Internet Directory using X.500 (Расширения для репликации и распределенных операций для обеспечения работы каталога Internet на основе протокола X.500)
- 1275 — Replication Requirements to provide an Internet Directory using X.500 (Требования к репликации для обеспечения работы каталога Internet на основе протокола X.500)
- 1274 — The COSINE and Internet X.500 Schema (Объединение сетей в Европе и схема каталога Internet на основе протокола X.500)
- 1255, 1218 — A Naming Scheme for c=US (Схема именования для c=US)
- 1249 — DIXIE Protocol Specification (Спецификация протокола DIXIE)

1202 — Directory Assistance service (Служба поддержки каталогов)  
1107 — Plan for Internet directory services (Проект служб каталогов Internet)

#### **6р. Информационные службы (HTTP, Gopher, WAIS)**

- 2718 — Guidelines for new URL Schemes (Основные принципы новых схем URL)  
2660 — The Secure HyperText Transfer Protocol (Защищенный протокол передачи гипертекста)  
2656 — Registration Procedures for SOIF Template Types (Процедуры регистрации для типов шаблонов обобщенного формата для обмена объектами)  
2655 — CIP Index Object Format for SOIF Objects (Формат индексных объектов CIP для объектов SOIF)  
2654 — A Tagged Index Object for use in the Common Indexing Protocol (Тэговый индексный объект, который используется в обобщенном протоколе индексирования)  
2653 — CIP Transport Protocols (Транспортные протоколы CIP)  
2652 — MIME Object Definitions for the Common Indexing Protocol (CIP) (Определение объектов MIME для обобщенного протокола индексирования)  
2651 — The Architecture of the Common Indexing Protocol (CIP) (Структура обобщенного протокола индексирования)  
2617, 2069 — HTTP Authentication: Basic and Digest Access Authentication (Аутентификация протокола HTTP: основная аутентификация доступа и аутентификация на основе дайджеста)  
2616, 2068 — Hypertext Transfer Protocol — HTTP/1.1 (Протокол передачи гипертекста, HTTP/1.1)  
2611 — URN Namespace Definition Mechanisms (Механизмы определения пространства имен унифицированных ресурсов)  
2518 — HTTP Extensions for Distributed Authoring — WEBDAV (Расширения протокола HTTP для распределенного управления документами — протокол WEBDAV)  
2483 — URI Resolution Services Necessary for URN Resolution (Обязательные службы преобразования унифицированных идентификаторов ресурсов для преобразования унифицированных имен ресурсов)  
2397 — The “data” URL scheme (Схема URL “data”)  
2396 — Uniform Resource Identifiers (URI): Generic Syntax (Унифицированные идентификаторы ресурсов (URI): общий синтаксис)  
2392, 2111 — Content-ID and Message-ID Uniform Resource Locators (Поля Content-ID и Message-ID унифицированных указателей ресурсов)  
2388 — Returning Values from Forms: multipart/form-data (Ввод данных с помощью форм: заголовок MIME multipart/form-data)  
2378 — The CCSO Nameserver (Ph) Architecture (Структура серверов имен CCSO)  
2369 — The Use of URLs as Meta-Syntax for Core Mail List Commands and their Transport through Message Header Fields (Использование URL в качестве метасинтаксиса для основных команд управления списками рассылок и их транспортировки с помощью значений полей заголовков сообщений)  
2368 — The mailto URL scheme (Схема URL mailto)  
2345 — Domain Names and Company Name Retrieval (Система доменных имен и определение имен для предприятий)  
2310 — The Safe Response Header Field (Поле заголовка, предназначенное для безопасного ответа)  
2296 — HTTP Remote Variant Selection Algorithm — RVSA/1.0 (Алгоритм дистанционного выбора вариантов протокола HTTP)  
2295 — Transparent Content Negotiation in HTTP (Согласование “прозрачного” содержимого в протоколе HTTP)

- 2291 — Requirements for a Distributed Authoring and Versioning Protocol for the World Wide Web (Требования к протоколу распределенного управления версиями документов для World Wide Web)
- 2288 — Using Existing Bibliographic Identifiers as Uniform Resource Names (Использование существующих библиографических идентификаторов в качестве унифицированных имен ресурсов)
- 2276 — Architectural Principles of Uniform Resource Name Resolution (Основные структурные принципы преобразования унифицированных имен ресурсов)
- 2259, 2258 — Simple Nomenclator Query Protocol (SNQP) (Простой протокол построения номенклатурных запросов)
- 2227 — Simple Hit-Metering and Usage-Limiting for HTTP (Простой механизм отслеживания статистики использования Web-страниц и ее ограничения для протокола HTTP)
- 2187, 2186 — Application of Internet Cache Protocol (ICP), version 2 (Применение протокола кэширования в Internet версии 2)
- 2169 — A Trivial Convention for using HTTP in URN Resolution (Простое соглашение по использованию протокола HTTP при преобразовании URN)
- 2168 — Resolution of Uniform Resource Identifiers using the Domain Name System (Преобразование унифицированных идентификаторов ресурсов с помощью доменной системы имен)
- 2167, 1714 — Referral Whois (RWhois) Protocol V1.5 (Направленный протокол Whois версии 1.5)
- 2145 — Use and Interpretation of HTTP Version Numbers (Использование и интерпретация номеров версий протокола HTTP)
- 2141 — URN Syntax (Синтаксис унифицированных имен ресурсов)
- 2122 — VEMMI URL Specification (Спецификация URL VEMMI)
- 2109 — HTTP State Management Mechanism (Механизм отслеживания состояний протокола HTTP)
- 2084 — Considerations for Web Transaction Security (Анализ безопасности Web-транзакций)
- 2056 — Uniform Resource Locators for Z39.50 (Унифицированные указатели ресурсов для Z39.50)
- 1945 — Hypertext Transfer Protocol — HTTP/1.0 (Протокол передачи гипертекста HTTP/1.0)
- 1914 — How to Interact with a Whois++ Mesh (Способы взаимодействия с сеткой службы Whois++)
- 1913 — Architecture of the Whois++ Index Service (Структура службы индексирования Whois++)
- 1835 — Architecture of the WHOIS++ service (Структуры службы Whois++)
- 1834 — Whois and Network Information Lookup Service, Whois++ (Whois и служба сетевого поиска информации, Whois++)
- 1808 — Relative Uniform Resource Locators (Относительные унифицированные указатели информационного ресурса)
- 1738 — Uniform Resource Locators (URL) (Унифицированный указатель информационного ресурса)
- 1737 — Functional Requirements for Uniform Resource Names (Функциональные требования для унифицированного указателя информационного ресурса)
- 1736 — Functional Recommendations for Internet Resource Locators (Функциональные рекомендации для указателей информационных ресурсов Internet)
- 1729 — Using the Z39.50 Information Retrieval Protocol (Использование протокола поиска информации Z39.50)
- 1728 — Resource Transponders (Преобразователи ресурсов)

- 1727 — A Vision of an Integrated Internet Information Service (Взгляд на интегрированную службу информации в Internet)
- 1630 — Universal Resource Identifiers in WWW: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World Wide Web (Универсальные идентификаторы ресурсов в WWW: унифицированный синтаксис для выражения имен и адресов объектов сети, использующихся в World Wide Web)
- 1625 — WAIS over Z39.50-1988 (Служба WAIS на основе протокола Z39.50-1988)
- 1614 — Network Access to Multimedia Information (Сетевой доступ к мультимедийной информации)
- 1436 — The Internet Gopher Protocol (a distributed document search and retrieval protocol) (Протокол Gopher в Internet (протокол поиска и выборки распределенных документов))
- 954, 812 — NICNAME/WHOIS (Сервер NICNAME/WHOIS)

#### **6q. Протоколы начальной загрузки и конфигурирования (BOOTP, DHCP)**

- 2563 — DHCP Option to Disable Stateless Auto-Configuration in IPv4 Clients (Параметры протокола DHCP, предназначенные для запрещения автоматической конфигурации клиентов протокола IPv4 без учета состояния)
- 2485 — DHCP Option for The Open Group's User Authentication Protocol (Параметры протокола DHCP, предназначенные для протокола аутентификации пользователей, разработанного Open Group)
- 2242 — NetWare/ IP Domain Name and Information (Доменные имена и информация в протоколе IP для сетей NetWare)
- 2241 — DHCP Options for Novell Directory Services (Параметры протокола DHCP, предназначенные для служб каталогов фирмы Novell)
- 2132, 1533, 1497, 1395, 1084, 1048 — DHCP Options and BOOTP Vendor Extensions (Параметры протокола DHCP и расширения производителей для протокола BOOTP)
- 2131, 1541, 1531 — Dynamic Host Configuration Protocol (Протокол динамической конфигурации узла сети)
- 1542, 1532 — Clarifications and Extensions for the Bootstrap Protocol (Разъяснения по поводу некоторых расширений протокола BOOTP)
- 1534 — Interoperation Between DHCP and BOOTP (Взаимодействие протоколов DHCP и BOOTP)
- 951 — Bootstrap Protocol (Протокол начальной загрузки)

#### **6г. Передачи мультимедийных данных в реальном масштабе времени и качество обслуживания (RSVP, RTP)**

- 2719 — Framework Architecture for Signaling Transport (Инфраструктура для передачи сигналов)
- 2705 — Media Gateway Control Protocol (MGCP) Version 1.0 (Протокол управления шлюзом среды передачи данных)
- 2689, 2688 — Integrated Services Mappings for Low Speed Networks (Карты соответствия интегрированных служб для низкоскоростных сетей)
- 2658 — RTP Payload Format for PureVoice(tm) Audio (Формат тела пакета протокола RTP для передачи аудиоинформации PureVoice)
- 2543 — SIP: Session Initiation Protocol (Протокол инициирования сеанса связи)
- 2490 — A Simulation Model for IP Multicast with RSVP (Модель эмуляции работы многоадресатной передачи с помощью протокола резервирования ресурсов)

- 2458 — Toward the PSTN/Internet Inter-Networking — Pre-PINT Implementations (Еще раз о реализации объединенной сети на основе коммутируемой телефонной сети общего доступа)
- 2448 — AT&T's Error Resilient Video Transmission Technique (Методики устойчивой к ошибкам передачи видеоданных)
- 2435, 2035 — RTP Payload Format for JPEG-compressed Video (Формат тела пакета протокола RTP для передачи видеоизображений, сжатых по методу JPEG)
- 2431 — RTP Payload Format for BT.656 Video Encoding (Формат тела пакета протокола RTP для передачи видеоданных, закодированных в стандарте BT.656)
- 2430 — A Provider Architecture for Differentiated Services and Traffic Engineering (PASTE) (Структура провайдера для схем дифференцированного обслуживания и передачи трафика)
- 2429 — RTP Payload Format for the 1998 Version of ITU-T Rec. H.263 Video (H.263+) (Формат тела пакета протокола RTP для передачи видеоданных стандарта H.263+)
- 2423, 2422, 2421, 1911 — VPIM Voice Message MIME Sub-type Registration (Регистрация подтипа MIME для голосовых сообщений)
- 2386 — A Framework for QoS-based Routing in the Internet (Инфраструктура маршрутизации в Internet на основе заданного качества обслуживания)
- 2382 — A Framework for Integrated Services and RSVP over ATM (Инфраструктура интегрированных служб и протокола RSVP в сети ATM)
- 2381 — Interoperation of Controlled-Load Service and Guaranteed Service with ATM (Взаимодействие служб управления нагрузкой и гарантированного качества обслуживания в сети ATM)
- 2380 — RSVP over ATM Implementation Requirements (Требования по реализации протокола RSVP в сети ATM)
- 2379 — RSVP over ATM Implementation Guidelines (Основные принципы реализации протокола RSVP в сети ATM)
- 2361 — WAVE and AVI Codec Registries (Регистрация кодеков WAVE и AVI)
- 2354 — Options for Repair of Streaming Media (Параметры, предназначенные для восстановления потоковых сред передачи данных)
- 2343 — RTP Payload Format for Bundled MPEG (Формат тела пакета протокола RTP для передачи видеоизображения в формате MPEG)
- 2327 — SDP: Session Description Protocol (Протокол описания сеанса связи SDP)
- 2326 — Real Time Streaming Protocol (RTSP) (Протокол потоковой передачи данных в реальном масштабе времени)
- 2250, 2038 — RTP Payload Format for MPEG1/MPEG2 Video (Формат тела пакета протокола RTP для передачи видеоизображения в формате MPEG1/MPEG2)
- 2216 — Network Element Service Specification Template (Шаблон спецификации элемента сетевой службы)
- 2215 — General Characterization Parameters for Integrated Service Network Elements (Параметры для общей характеристики элементов интегрированных сетевых служб)
- 2212 — Specification of Guaranteed Quality of Service (Спецификация службы гарантированного качества обслуживания)
- 2211 — Specification of the Controlled-Load Network Element Service (Спецификация элемента службы управления нагрузкой)
- 2210 — The Use of RSVP with IETF Integrated Services (Использование протокола RSVP в интегрированных службах IETF)

- 2209 — Resource ReSerVation Protocol (RSVP) — Version 1 Message Processing Rules (Протокол резервирования ресурсов версии 1: правила обработки сообщений)
- 2208 — Resource ReSerVation Protocol (RSVP) — Version 1 Applicability Statement Some Guidelines on Deployment (Протокол резервирования ресурсов. Заявление по поводу применимости версии 1. Некоторые принципы внедрения протокола)
- 2207 — RSVP Extensions for IPSEC Data Flows (Расширения протокола RSVP для передачи потоков данных протокола IPsec)
- 2206 — RSVP Management Information Base using SMIv2 (База управляющей информации для протокола RSVP при использовании SMIv2)
- 2205 — Resource ReSerVation Protocol (RSVP) — Version 1 Functional Specification (Протокол резервирования ресурсов — функциональные требования к версии 1)
- 2198 — RTP Payload for Redundant Audio Data (Формат тела пакета протокола RTP, позволяющий уменьшить количество передаваемых аудиоданных)
- 2190 — RTP Payload Format for H.263 Video Streams (Формат тела пакета протокола RTP для передачи потоков видеоданных в формате H.263)
- 2032 — RTP Payload Format for H.261 Video Streams (Формат тела пакета протокола RTP для передачи потоков видеоданных в формате H.261)
- 2029 — RTP Payload Format of Sun's CellB Video Encoding (Формат тела пакета протокола RTP для передачи потоков видеоданных, закодированных в формате CellB фирмы Sun)
- 1890 — RTP Profile for Audio and Video Conferences with Minimal Control (RTP-профиль для аудио- и видеоконференций с минимальным количеством средств управления)
- 1889 — RTP: A Transport Protocol for Real-Time Applications (RTP: транспортный протокол для приложений, работающих в реальном масштабе времени)
- 1821 — Integration of Real-time Services in an IP-ATM Network Architecture (Интеграция служб реального времени в структуру сети IP-ATM)
- 1789 — INETPhone: Telephone Services and Servers on Internet (INETPhone: телефонная служба и серверы в Internet)
- 1257 — Isochronous applications do not require jitter-controlled networks (Изохронным приложениям не требуются сети с управляемой величиной дрожания)
- 1193 — Client requirements for real-time communication services (Клиентские требования для выполнения взаимодействия с серверами в реальном масштабе времени)
- 741 — Specifications for the Network Voice Protocol (NVP) (Спецификация сетевого протокола передачи голосовых данных)

#### **6s. Другие**

- 2703 — Protocol-independent Content Negotiation Framework (Независимая от протокола инфраструктура согласования содержимого)
- 2614 — An API for Service Location (API для обнаружения необходимых служб)
- 2610 — DHCP Options for Service Location Protocol (Параметры протокола DHCP для протокола обнаружения служб)
- 2609 — Service Templates and Service: Schemes (Шаблоны служб и сами службы: проекты)
- 2608, 2165 — Service Location Protocol, Version 2 (Протокол обнаружения служб, версия 2)

- 2552 — Architecture for the Information Brokerage in the ACTS Project GAIA  
(Структура для информационного посредничества в проекте ACTS GAIA)
- 2533 — A Syntax for Describing Media Feature Sets (Синтаксис для описания возможностей среды передачи данных)
- 2447, 2446, 2445 — iCalendar Message-Based Interoperability Protocol (iMIP)  
(Протокол взаимодействия на основе сообщений iCalendar)
- 2244 — ACAP — Application Configuration Access Protocol (Протокол конфигурирования доступа со стороны приложений)
- 2229 — A Dictionary Server Protocol (Протокол сервера словаря)
- 2188 — AT&T/Neda's Efficient Short Remote Operations (ESRO) Protocol Specification Version 1.2 (Спецификация протокола AT&T/Neda эффективных коротких дистанционных операций версии 1.2)
- 2016 — Uniform Resource Agents (URAs) (Унифицированные агенты ресурсов)
- 1861, 1645, 1568 — Simple Network Paging Protocol — Version 3 — Two-Way Enhanced (Простой сетевой протокол разбивки на страницы, версия 3 — двухступенчатое расширение)
- 1756 — Remote Write Protocol — Version 1.0 (Протокол дистанционной записи, версия 1)
- 1703, 1569 — Principles of Operation for the TPC.INT Subdomain: Radio Paging — Technical Procedures (Принципы работы поддомена TPC.INT: радиопейджеры — формальные процедуры)
- 1692 — Transport Multiplexing Protocol (TMux) (Транспортный мультиплексирующий протокол)
- 1530 — Principles of Operation for the TPC.INT Subdomain: General Principles and Policy (Принципы работы поддомена TPC.INT: общие принципы и правила)
- 1492 — An Access Control Protocol, Sometimes Called TACACS (Протокол управления доступом, иногда называемый TACACS)
- 1459 — Internet Relay Chat Protocol (Протокол ретрансляции диалогов в Internet)
- 1429 — Listserv Distribute Protocol (Протокол рассылки Listserv)
- 1413, 931, 912 — Identification Protocol (Протокол идентификации)
- 1307 — Dynamically Switched Link Control Protocol (Протокол управления динамически коммутируемым каналом связи)
- 1288, 1196, 1194, 742 — The Finger User Information Protocol (Протокол получения информации о пользователе Finger)
- 1179 — Line printer daemon protocol (Протокол демона управления построчно печатающим устройством)
- 978 — Voice File Interchange Protocol (VFIP) (Протокол обмена файлами, содержащими голосовые данные)
- 909 — Loader-Debugger Protocol (Протокол загрузки и отладки)
- 891 — DCN local-network protocols (Протоколы локальной распределенной компьютерной сети)
- 887 — Resource Location Protocol (Протокол поиска ресурсов)
- 866 — Active users (Активные пользователи)
- 865 — Quote of the Day Protocol (Протокол управления цитатой дня)
- 864 — Character Generator Protocol (Протокол управления генератором символов)
- 863, 348 — Discard Protocol (Протокол аннулирования)
- 862, 347 — Echo Protocol (Протокол эха)
- 767 — Structured format for transmission of multi-media documents (Структурированный формат для передачи мультимедийных документов)
- 759 — Internet Message Protocol (Протокол сообщений Internet)
- 734 — SUPDUP Protocol (Высокоэффективный протокол telnet для отображения данных SUPDUP)

- 666 — Specification of the Unified User-Level Protocol (Спецификация унифицированного протокола пользовательского уровня)
- 621 — NIC user directories at SRI ARC (Каталоги пользователей NIC в SRI ARC)
- 569 — NETED: A Common Editor for the ARPA Network (NETED: универсальный редактор для сети ARPA)
- 470 — Change in socket for TIP news facility (Изменение в сокете, поддерживающем новости для пользователей TIP)
- 451 — Tentative proposal for a Unified User Level Protocol (Экспериментальное предложение для унифицированного протокола пользовательского уровня)
- 109 — Level III Server Protocol for the Lincoln Laboratory NIC 360/67 Host (Серверный протокол уровня III для узла NIC 360/67 лаборатории Линкольна)
- 98, 79 — Logger Protocol Proposal (Предложенный протокол регистрации)
- 29 — Response to RFC 28 (Ответ на документ RFC 28)

## **7. Программная документация**

- 1761 — Snoop Version 2 Packet Capture File Format (Формат файла перехвата пакетов программы Snoop версии 2)
- 496 — TNLS quick reference card is available (Выпущена справочная карточка пользователя TNLS)
- 494 — Availability of MIX and MIXAL in the Network (Доступность MIX и MIXAL в сети)
- 488 — NLS classes at network sites (Классы NLS в сетевых центрах)
- 485 — MIX and MIXAL at UCSB (MIX и MIXAL в Калифорнийском университете в Санта-Барбара)
- 431 — Update on SMFS Login and Logout (Изменение команд Login и Logout системы SMFS Калифорнийского университета в Санта-Барбара)
- 411 — New MULTICS Network Software Features (Новые возможности сетевого программного обеспечения системы MULTICS)
- 409 — Tenex interface to UCSB's Simple-Minded File System (Интерфейс Tenex к простой файловой системе Калифорнийского университета в Санта-Барбара)
- 399 — SMFS Login and Logout (Команды Login и Logout системы SMFS)
- 390 — TSO Scenario (Сценарии TSO)
- 382 — Mathematical Software on the ARPA Network (Математическое программное обеспечение в сети ARPA)
- 379 — Using TSO at CCN (Использование системы разделения времени в CCN)
- 373 — Arbitrary Character Sets (Произвольные наборы символов)
- 350 — User Accounts for UCSB On-Line System (Учетные записи пользователей для интерактивной системы Калифорнийского университета в Санта-Барбара)
- 345 — Interest in Mixed Integer Programming (MPSX on NIC 360/91 at CCN) (Заинтересованность в программировании с использованием смешанной целочисленной арифметики (система MPSX на узле NIC 360/91 в CCN))
- 321 — CBI Networking Activity at MITRE (Сетевой процесс, основанный на командах компьютеру в MITRE)
- 311 — New Console Attachments to the UCSB Host (Новые консольные дополнения к узлу UCSB)
- 251 — Weather data (Данные о погоде)
- 217 — Specifications changes for OLS, RJE/RJOR, and SMFS (Изменения в спецификации для OLS, RJE/RJOR и SMFS)

- 174 — UCLA — Computer Science Graphics Overview (Обзор графических возможностей компьютеров факультета информатики Калифорнийского университета в Лос-Анджелесе)
- 122 — Network specifications for UCSB's Simple-Minded File System (Сетевые спецификации для простой файловой системы Калифорнийского университета в Санта-Барбаре)
- 121 — Network on-line operators (Сетевые интерактивные операторы)
- 120 — Network PL1 subprograms (Сетевые подпрограммы на языке PL1)
- 119 — Network Fortran subprograms (Сетевые подпрограммы на языке Fortran)
- 74 — Specifications for network use of the UCSB On-Line System (Спецификации по использованию сети в интерактивной системе Калифорнийского университета в Санта-Барбаре)

## **8. Специфические сети (см. также раздел 3)**

### **8а. ARPANET**

- 1005, 878, 851, 802 — ARPANET AHIP-E Host Access Protocol (enhanced AHIP) (Протокол AHIP-Е для доступа к узлу сети ARPANET (расширенный протокол AHIP))
- 852 — ARPANET short blocking feature (Возможность сокращения блокировки узла сети ARPANET)
- 789 — Vulnerabilities of network control protocols: An example (Пример уязвимости сетевых управляющих протоколов)
- 745 — JANUS interface specifications (Спецификация интерфейса JANUS)
- 716 — Interim Revision to Appendix F of BBN 1822 (Промежуточная редакция приложения F документа BBN 1822)
- 704 — IMP/Host and Host/IMP Protocol change (Изменения протоколов IMP/Host и Host/IMP)
- 696 — Comments on the IMP/Host and Host/IMP Protocol changes (Комментарии по поводу изменений в протоколах IMP/Host и Host/IMP)
- 695 — Official change in Host-Host Protocol (Официальные изменения в протоколе обмена сообщениями между двумя узлами сети)
- 692 — Comments on IMP/Host Protocol changes (RFCs 687 and 690) (Комментарии по поводу изменений в протоколе IMP/Host (Документы RFC 687 и 690))
- 690 — Comments on the proposed Host/IMP Protocol changes (Комментарии по поводу изменений в протоколе Host/IMP)
- 687 — IMP/Host and Host/IMP Protocol changes (Изменения в протоколах IMP/Host и Host/IMP)
- 667 — BBN host ports (Порты узла BBN)
- 660 — Some changes to the IMP and the IMP/Host interface (Некоторые изменения в интерфейсах IMP и IMP/Host)
- 642 — Ready line philosophy and implementation (Стратегия линии готовности и реализации)
- 638, 633 — IMP/TIP preventive maintenance schedule (Расписание проведения профилактических мероприятий в системах IMP/TIP)
- 632 — Throughput degradations for single packet messages (Падение пропускной способности для сообщений, состоящих из одного пакета)
- 627 — ASCII text file of hostnames (По поводу текстового файла в формате ASCII, содержащего список узлов)
- 626 — On a possible lockup condition in IMP subnet due to message sequencing (К вопросу об условиях возникновения блокировки в подсети IMP, вызванной упорядочением сообщений)

- 625 — On-line hostnames service (Оперативная служба имен узлов)  
 623 — Comments on on-line host name service (Комментарии по поводу оперативной службы имен узлов)  
 622 — Scheduling IMP/TIP down time (Расписание отключения систем IMP/TIP)  
 620 — Request for monitor host table updates (Запрос на отслеживание обновлений таблицы узлов)  
 619 — Mean round-trip times in the ARPANET (Среднее время полного прохождения пакетов в сети ARPANET)  
 613 — Network connectivity: A response to RFC 603 (Связность узлов сети: ответ на RFC 603)  
 611 — Two changes to the IMP/Host Protocol to improve user/network communications (Два изменения в протоколе IMP/Host, направленные на улучшение взаимодействия пользователь/сеть)  
 606 — Host names on-line (Оперативный доступ к именам узлов сети)  
 594 — Speedup of Host-IMP interface (Ускорение интерфейса Host-IMP)  
 591 — Addition to the Very Distant Host specifications (Дополнения к спецификации очень удаленных узлов сети)  
 568, 567 — Response to RFC 567 — cross country network bandwidth (Ответ на RFC 567 — производительность международной сети)  
 548 — Hosts using the IMP Going Down message (Узлы, использующие сообщение IMP Going Down)  
 547 — Change to the Very Distant Host specification (Изменения спецификации очень удаленных узлов сети)  
 533 — Message-ID numbers (Номера идентификаторов сообщений)  
 528 — Software checksumming in the IMP and network reliability (Программы для вычисления контрольных сумм в IMP и надежность сети)  
 521 — Restricted use of IMP DDT (Ограничено использование IMP DDT)  
 508 — Real-time data transmission on the ARPANET (Передача данных в реальном масштабе времени в сети ARPANET)  
 476, 434 — IMP/TIP memory retrofit schedule (rev 2) (Расписание изменения распределения памяти в IMP/TIP, издание 2)  
 449, 442 — Current flow-control scheme for IMPSYS (Действующая схема управления потоком для IMPSYS)  
 447, 445 — IMP/TIP memory retrofit schedule (Расписание изменения распределения памяти в IMP/TIP)  
 417 — Link usage violation (Нарушение использования канала)  
 410 — Removal of the 30-Second Delay When Hosts Come Up (Устранение 30-секундной задержки при подключении узла сети)  
 406 — Scheduled IMP Software Releases (Запланированные выпуски программного обеспечения IMP)  
 395 — Switch Settings on IMPs and TIPs (Настройки коммутатора в IMP и TIP)  
 394 — Two Proposed Changes to the IMP-Host Protocol (Два предлагаемых изменения в протоколе IMP-Host)  
 369 — Evaluation of ARPANET services January-March, 1972 (Оценка служб ARPANET за январь–март 1972 года)  
 335 — New Interface — IMP/360 (Новый интерфейс — IMP/360)  
 312 — Proposed Change in IMP-to-Host Protocol (Предлагаемые изменения в протоколе IMP-to-Host)  
 297 — TIP Message Buffers (Размер буферов сообщений TIP)  
 280 — A Draft of Host Names (Рабочий вариант списка имен узлов)  
 274 — Establishing a local guide for network usage (Утверждение локальных правил использования сети)  
 273, 237 — More on standard host names (Подробнее о стандартизации имен узлов)

- 271 — IMP System change notifications (Уведомление об изменении системы IMP)  
 270 — Correction to BBN Report No 1822 (Изменения в отчете BBN № 1822)  
 263 — “Very Distant” Host interface (Интерфейс “очень удаленного” узла)  
 254 — Scenarios for using ARPANET computers (Сценарии использования компьютеров в сети ARPANET)  
 247 — Proffered set of standard host names (Предпочитаемый набор стандартных имен узлов)  
 241 — Connecting computers to MLC ports (Подключение компьютеров к портам многоканального контроллера)  
 239 — Host mnemonics proposed in RFC 226 (NIC 7625) (Мнемоники узлов, предложенные в RFC 226 (NIC 7625))  
 236 — Standard host names (Стандартные имена узлов)  
 233 — Standardization of host call letters (Стандартизация позывных узлов)  
 230 — Toward reliable operation of minicomputer-based terminals on a TIP (Еще раз о надежности выполняемых операций на терминале миникомпьютера в системе TIP)  
 229 — Standard host names (Стандартные имена узлов)  
 228 — Clarification (Пояснение)  
 226 — Standardization of host mnemonics (Стандартизация мнемонических имен узлов)  
 218 — Changing the IMP status reporting facility (Изменение возможности получения отчетов по состоянию системы IMP)  
 213 — IMP System change notification (Уведомление об изменении системы IMP)  
 209 — Host/IMP interface documentation (Документация по интерфейсу Host/IMP)  
 208 — Address tables (Таблицы адресов)  
 73, 67 — Response to NWG/RFC 67 (Ответ на документ NWG/RFC 67)  
 71 — Reallocation in Case of Input Error (Перераспределение в случае возникновения ошибки ввода)  
 70 — Note on Padding (Замечание по поводу заполнения)  
 64 — Getting rid of marking (Избавление от маркировки)  
 41 — IMP-IMP Teletype Communication (Взаимодействие по телетайпу между системами IMP)  
 25 — No High Link Numbers (Не должно быть больших номеров каналов)  
 19 — Two protocol suggestions to reduce congestion at swap bound nodes (Два предлагаемых протокола для уменьшения перегрузки узлов, выгружающих данные на диск)  
 17 — Some questions re: Host-IMP Protocol (Несколько вопросов по поводу протокола Host-IMP Protocol)  
 12 — IMP-Host interface flow diagrams (Структурные схемы интерфейса IMP-Host)  
 7 — Host-IMP interface (Интерфейс “узел-IMP”)  
 6 — Conversation with Bob Kahn (Интервью с Бобом Канном)

#### **8b. Протоколы внешнего интерфейса узла сети**

- 929, 928, 705, 647 — Proposed Host-Front End Protocol (Предложенный протокол внешнего интерфейса узла сети)

#### **8c. Протокол управления сетью ARPANET (устаревший предшественник протокола TCP/IP)**

- 801 — NCP/TCP transition plan (План перехода с протокола NCP на протокол TCP)  
 773 — Comments on NCP/TCP mail service transition strategy (Комментарии по поводу стратегии перехода почтовых систем с протокола NCP на протокол TCP)

- 714 — Host-Host Protocol for an ARPANET-Type Network (Протокол обмена сообщениями между узлами сети типа ARPANET)  
 689 — Tenex NCP finite state machine for connections (Конечный автомат протокола управления сетью Tenex для соединений)  
 663 — Lost message detection and recovery protocol (Обнаружение утерянного сообщения и протокол восстановления)  
 636 — TIP/Tenex reliability improvements (Улучшение надежности системы TIP/Tenex)  
 635 — Assessment of ARPANET protocols (Оценка протоколов ARPANET)  
 534, 516, 512 — Lost message detection (Обнаружение утерянного сообщения)  
 492, 467 — Response to RFC 467 (Ответ на RFC 467)  
 489 — Comment on resynchronization of connection status proposal (Комментарии по поводу предложенного механизма повторной синхронизации состояния соединения)  
 425 — “But my NCP costs \$500 a day” (“Однако мой NCP обходится \$500 в день”)  
 210 — Improvement of Flow Control (Усовершенствование управления потоками)  
 176 — Comments on “Byte size for connections” (Комментарии по поводу размера байта при соединении)  
 165 — Proffered official Initial Connection Protocol (Официально предложенный протокол начальной установки соединения)  
 147 — Definition of a socket (Определение сокета)  
 142 — Time-Out Mechanism in the Host-Host Protocol (Механизм тайм-аута в протоколе обмена сообщениями между узлами сети)  
 132, 124, 107, 102 — Typographical Error in RFC 107 (Опечатки в RFC 107)  
 129 — Request for comments on socket name structure (Запрос на комментарии по поводу структуры, содержащей имя сокета)  
 128 — Bytes (Байты)  
 117 — Some comments on the official protocol (Несколько комментариев по поводу официального протокола)  
 72 — Proposed Moratorium on Changes to Network Protocol (Предлагается мораторий на изменения, вносимые в сетевой протокол)  
 68 — Comments on Memory Allocation Control Commands: CEASE, ALL, GVB, RET, and RFNM (Пояснения к командам управления распределением памяти: CEASE, ALL, GVB, RET и RFNM)  
 65 — Comments on Host/Host Protocol document #1 (Пояснения к протоколу обмена сообщениями между узлами сети, документ №1)  
 60 — Simplified NCP Protocol (Упрощенный протокол управления сетью)  
 59 — Flow Control — Fixed Versus Demand Allocation (Управление потоком — сравнение фиксированного распределения и распределения по запросу)  
 58 — Logical Message Synchronization (Синхронизация логических сообщений)  
 57, 54 — Thoughts and Reflections on NWG/RFC 54 (Размышления по поводу документа NWG/RFC 54)  
 56 — Third Level Protocol: Logger Protocol (Протокол третьего уровня: протокол регистрации)  
 55 — Prototypical implementation of the NCP (Реализация протокола NCP по прототипу)  
 50, 49, 47, 45, 44, 40, 39, 38, 36, 33 — Comments on the Meyer Proposal (Комментарии по поводу предложения Мейера)  
 42 — Message Data Types (Типы данных сообщения)  
 23 — Transmission of Multiple Control Messages (Передача нескольких управляющих сообщений)  
 22 — Host-host control message formats (Форматы управляющих сообщений протокола “узел-узел”)

- 18 — IMP-IMP and HOST-HOST Control Links (Каналы управления протоколов IMP-IMP и “узел-узел”)
- 15 — Network subsystem for time sharing hosts (Сетевая подсистема для узлов работающих в режиме разделения времени)
- 11 — Implementation of the Host-Host software procedures in GORDO (Реализация программных процедур взаимодействия между двумя узлами сети в GORDO)
- 9, 1 — Host software (Программное обеспечение узла сети)
- 8 — Functional specifications for the ARPA Network (Функциональные требования для сети ARPA)
- 5 — Decode Encode Language (DEL) (Язык декодирования/кодирования)
- 2 — Host software (Программное обеспечение узла сети)

#### **8d. Протокол начальной установки соединения в сети ARPANET**

- 202 — Possible Deadlock in ICP (Возможные ситуации взаимоблокировки в протоколе ICP)
- 197 — Initial Connection Protocol — Reviewed (Пересмотренный протокол начальной установки соединения)
- 161 — Solution to the race condition in the ICP (Решение проблемы конкурентной борьбы в протоколе ICP)
- 151, 148, 143, 127, 123 — Comments on a proffered official ICP: RFCs 123, 127 (Комментарии по поводу официально предложенного протокола ICP: документы RFC 123, 127)
- 150 — Use of IPC Facilities: A Working Paper (Использование средств протокола IPC: рабочий доклад)
- 145 — Initial Connection Protocol Control Commands (Команды управления протоколом начальной установки соединения)
- 93 — Initial Connection Protocol (Протокол начальной установки соединения)
- 80 — Protocols and Data Formats (Протоколы и форматы данных)
- 66 — Third level ideas and other noise (Идеи протокола третьего уровня и другие слухи)

#### **8e. USENET**

- 1036, 850 — Standard for interchange of USENET messages (Стандарт для обмена сообщениями USENET)

#### **8f. Другие**

- 1553 — Compressing IPX Headers Over WAN Media (CIPX) (Сжатие IPX-заголовков при передаче в среде распределенных сетей)
- 1132 — Standard for the transmission of 802.2 packets over IPX networks (Стандарт передачи пакетов 802.2 по сетям IPX)
- 935 — Reliable link layer protocols (Надежные протоколы канального уровня)
- 916 — Reliable Asynchronous Transfer Protocol (RATP) (Надежный протокол асинхронной передачи данных)
- 914 — Thinwire protocol for connecting personal computers to the Internet (Протокол для подключения персональных компьютеров к Internet по тонким проводам)
- 824 — CRONUS Virtual Local Network (Виртуальная локальная сеть CRONUS)

## **9. Система измерений**

### **9a. Общие вопросы**

- 2724 — RTFM: New Attributes for Traffic Flow Measurement (RTFM: Новые атрибуты для измерения величины потока трафика)
- 2723 — SRL: A Language for Describing Traffic Flows and Specifying Actions for Flow Groups (Простой язык правил: язык для описания потоков трафика и указания действий для групп потоков)
- 2722 — Traffic Flow Measurement: Architecture (Измерение потока трафика: структура)
- 2721 — RTFM: Applicability Statement (RTFM: заявление по поводу применимости)
- 2681 — A Round-trip Delay Metric for IPPM (Метрика оценки полного времени прохождения пакета для IPPM)
- 2680 — A One-way Packet Loss Metric for IPPM (Односторонняя метрика оценки потерь пакетов для IPPM)
- 2679 — A One-way Delay Metric for IPPM (Односторонняя метрика оценки задержки прохождения пакета для IPPM)
- 2678, 2498 — IPPM Metrics for Measuring Connectivity (Метрики IPPM для оценки возможности взаимодействия компьютеров)
- 2544, 1944 — Benchmarking Methodology for Network Interconnect Devices (Методика оценки производительности для устройств, взаимодействующих по сети)
- 2432 — Terminology for IP Multicast Benchmarking (Терминология для оценки производительности многоадресатной передачи по протоколу IP)
- 2330 — Framework for IP Performance Metrics (Инфраструктура для метрик производительности протокола IP)
- 2285 — Benchmarking Terminology for LAN Switching Devices (Терминология для оценки производительности устройств коммутации локальной сети)
- 1857, 1404 — A Model for Common Operational Statistics (Модель для создания общей статистики по работе устройства)
- 1273 — Measurement Study of Changes in Service-Level Reachability in the Global TCP/IP Internet: Goals, Experimental Design, Implementation, and Policy Considerations (Система оценки изменений достижимости на уровне служб в глобальной сети Internet на основе протокола TCP/IP: цели, экспериментальный проект, реализация и анализ правил)
- 1262 — Guidelines for Internet Measurement Activities (Принципы выполнения измерений в Internet)
- 557 — Revelations in network host measurements (Подробности измерения параметров узлов)
- 546 — Tenex load averages for July 1973 (Средняя загрузка системы Tenex за июль 1973 года)
- 462 — Responding to user needs (Реакция на требования пользователей)
- 415 — Tenex bandwidth (Пропускная способность системы Tenex)
- 392 — Measurement of host costs for transmitting network data (Измерение издержек в работе узла сети, вызванных передачей данных по сети)
- 352 — TIP Site Information Form (Информационный бланк пользователя системы TIP)
- 308 — ARPANET host availability data (Данные по доступности узлов ARPANET)
- 286 — Network Library Information System (Информационная система сетевой библиотеки)
- 214, 193 — Network checkpoint (Сетевые контрольные точки)
- 198 — Site Certification — Lincoln Labs 360/67 (Сертификация сетевого центра лаборатории Линкольна 360/67)

- 182 — Compilation of list of relevant site reports (Составление списка отчетов значимых сетевых центров)
- 180 — File system questionnaire (Опросный лист по файловой системе)
- 156 — Status of the Illinois site: Response to RFC 116 (Статус сетевого центра штата Иллинойс: ответ на документ RFC 116)
- 153 — SRI ARC-NIC status (Статус SRI ARC-NIC)
- 152 — SRI Artificial Intelligence status report (Данные о состоянии группы исследования искусственного интеллекта Станфордского научно-исследовательского института)
- 126 — Graphics Facilities at Ames Research Center (Возможности по обработке графической информации исследовательского центра Амеса)
- 112 — User/Server Site Protocol: Network host questionnaire responses (Протокол пользователь/сервер: ответы на вопросы, заданные в RFC 106)
- 106 — User/Server Site Protocol Network Host Questionnaire (Протокол пользователя/сервер: опросный лист узла сети)
- 104 — Link 191 (Канал 191)

### **9b. Обзоры**

- 971 — Survey of data representation standards (Обзор стандартов представления данных)
- 876 — Survey of SMTP implementations (Обзор реализаций протокола SMTP)
- 848 — Who provides the “little” TCP services? (Кто должен обеспечивать поддержку “облегченных” служб протокола TCP)
- 847 — Summary of Smallberg surveys (Краткая информация об обзорах Дэвида Смаллберга)
- 846, 845, 843, 842, 839, 838, 837, 836, 835, 834, 833, 832 — Who talks TCP? — survey of 22 February 1983 (Кто пользуется протоколом TCP? Обзор за 22 февраля 1983 года)
- 844 — Who talks ICMP, too? — Survey of 18 February 1983 (Кто пользуется еще и протоколом ICMP? Обзор за 18 февраля 1983 года)
- 787 — Connectionless data transmission survey/tutorial (Передача данных без установки соединения (обзор/справочник))
- 565 — Storing network survey data at the datacomputer (Данные обследования сетей хранения данных в дейтакомпьютерах)
- 545 — Of what quality be the UCSB resources evaluators? (Какого качества должны быть эксперты по оценке ресурсов в Калифорнийском университете в Санта-Барбаре)
- 530 — Report on the Survey project (Отчет по поводу проекта SURVEY)
- 523 — SURVEY is in operation again (Система SURVEY снова работает)
- 519 — Resource evaluation (Исследование ресурсов)
- 514 — Network make-work (Создание сетевых рабочих мест)
- 464 — Resource notebook framework (Инфраструктура блокнота ресурсов)
- 460 — NCP survey (Обзор протокола управления сетью)
- 459 — Network questionnaires (Сетевой опросный лист)
- 450 — MULTICS sampling timeout change (Опытное изменение значения таймаута в системе MULTICS)
- 446 — Proposal to consider a network program resource notebook (Предложение к обсуждению программы сетевого блокнота ресурсов)
- 96 — An Interactive Network Experiment to Study Modes of Access the Network Information Center (Интерактивный сетевой эксперимент по изучению режимов доступа к сетевому информационному центру)
- 90 — CCN as a Network Service Center (Использование CCN в качестве сетевого сервисного центра)

- 81 — Request for Reference Information (Запрос на справочную информацию)  
78 — NCP Status Report: UCSB/Rand (Данные о состоянии NCP: UCSB/Rand)

### **9c. Статистические данные**

- 1030 — On testing the NETBLT Protocol over divers networks (К вопросу использования протокола NETBLT в разнородных сетях)  
996 — Statistics server (Сервер сбора статистических данных)  
618 — Few observations on NCP statistics (Несколько замечаний по поводу статистических данных NCP)  
612, 601, 586, 579, 566, 556, 538, 522, 509, 497, 482, 455, 443, 422, 413, 400, 391, 378 — Traffic statistics (December 1973) (Статистические данные по трафику за декабрь 1973 года)  
603, 597, 376, 370, 367, 366, 362, 353, 344, 342, 332, 330, 326, 319, 315, 306, 298, 293, 288, 287, 267, 266 - Response to RFC 597: Host status (Ответ на RFC 597: состояние узла сети)  
550 — NIC NCP experiment (Эксперимент NIC NCP)  
388 — NCP statistics (Статистика NCP)  
255, 252, 240, 235 — Status of network hosts (Состояние узлов сети)

## **10. Конфиденциальность, безопасность и аутентификация**

### **10a. Общие сведения**

- 2716 — PPP EAP TLS Authentication Protocol (Протокол аутентификации EAP TLS протокола PPP)  
2712 — Addition of Kerberos Cipher Suites to Transport Layer Security (TLS) (Добавления к комплекту шифрования Kerberos для реализации защищенного транспортного протокола)  
2704 — The KeyNote Trust-Management System Version 2 (Система управления на основе доверительных отношений KeyNote, версия 2)  
2693 — SPKI Certificate Theory (Теория сертификации SPKI)  
2692 — SPKI Requirements (Требования к простой инфраструктуре открытых ключей)  
2630 — Cryptographic Message Syntax (Синтаксис зашифрованного сообщения)  
2628 — Simple Cryptographic Program Interface (Crypto API) (Простой криптографический программный интерфейс Crypto API)  
2627 — Key Management for Multicast: Issues and Architectures (Управление ключами при многоадресатной рассылке: вопросы и структура)  
2538 — Storing Certificates in the Domain Name System (DNS) (Хранение сертификатов в системе доменных имен)  
2537 — RSA/MD5 KEYs and SIGs in the Domain Name System (DNS) (Хранение ключей RSA/MD5 и сигнатур в системе доменных имен)  
2536 — DSA KEYs and SIGs in the Domain Name System (DNS) (Хранение ключей DSA/MD5 и сигнатур в системе доменных имен)  
2523, 2522 — Photuris: Extended Schemes and Attributes (Протокол Photuris: расширенная структура и атрибуты)  
2504 — Users' Security Handbook (Руководство по безопасности для пользователей)  
2479 — Independent Data Unit Protection. Generic Security Service Application Program Interface (IDUP-GSS-API) (Независимая защита блоков данных. Защищенный программный интерфейс общих служб IDUP-GSS-API)  
2478, 2078, 1508 — The Simple and Protected GSS-API Negotiation Mechanism (Простой и защищенный механизм GSS-API)  
2444 — The One-Time-Password SASL Mechanism (Механизм присвоения одноразовых паролей SASL)

- 2440 — OpenPGP Message Format (Формат сообщения OpenPGP)  
2437, 2313 — PKCS #1: RSA Cryptography Specifications Version 2.0 (PKCS #1: спецификация системы шифрования RSA, версия 2.0)  
2367 — PF\_KEY Key Management API, Version 2 (Интерфейс прикладных программ системы управления ключами PF\_KEY, версия 2)  
2315 — PKCS 7: Cryptographic Message Syntax Version 1.5 (PKCS 7: Синтаксис зашифрованного сообщения, версия 1.5)  
2314 — PKCS 10: Certification Request Syntax Version 1.5 (PKCS 10: синтаксис запроса на сертификацию, версия 1.5)  
2289, 2243, 1938 — A One-Time Password System (Система присвоения одноразовых паролей)  
2267 — Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing (Фильтрация входа в сеть: ликвидация атаки методом отказа от обслуживания поступившего запроса, в котором используется подмена IP-адреса отправителя)  
2246 — The TLS Protocol Version 1.0 (Защищенный транспортный протокол, версия 1.0)  
2245 — Anonymous SASL Mechanism (Анонимный механизм SASL)  
2222 — Simple Authentication and Security Layer (SASL) (Уровень простой аутентификации и безопасности)  
2196, 1244 — Site Security Handbook (Руководство по безопасности для сетевого центра)  
2179 — Network Security For Trade Shows (Система безопасности сети при работе в демонстрационном режиме)  
2094 — Group Key Management Protocol (GKMP) Architecture (Структура протокола управления группами ключей)  
2093 — Group Key Management Protocol (GKMP) Specification (Спецификация протокола управления группами ключей)  
2025 — The Simple Public-Key GSS-API Mechanism (SPKM) (Простой механизм открытых ключей и GSS-API)  
1991 — PGP Message Exchange Formats (Форматы сообщений для обмена данными PGP)  
1964 — The Kerberos Version 5 GSS-API Mechanism (Механизм GSS-API протокола Kerberos версии 5)  
1949 — Scalable Multicast Key Distribution (Масштабируемая многоадресатная рассылка ключей)  
1948 — Defending Against Sequence Number Attacks (Защита от атак методом подмены порядкового номера)  
1898 — CyberCash Credit Card Protocol Version 0.8 (Протокол работы с кредитной картой CyberCash версии 0.8)  
1824 — The Exponential Security System TESS: An Identity-Based Cryptographic Protocol for Authenticated Key-Exchange (E.I.S.S.-Report 1995/4) (Экспоненциальная система безопасности TESS: криптографический протокол, основанный на проверке подлинности для аутентичного обмена ключами)  
1805 — Location-Independent Data/Software Integrity Protocol (Независимый от местонахождения протокол проверки целостности данных и программ)  
1760 — The S/KEY One-Time Password System (Система присвоения одноразовых паролей S/KEY)  
1751 — A Convention for Human-Readable 128-bit Keys (Соглашение по записи 128-битовых ключей в удобной для восприятия человеком форме)  
1750 — Randomness Recommendations for Security (Рекомендации по улучшению системы безопасности на основе случайности)

- 1704 — On Internet Authentication (Аутентификация в Internet)  
 1511 — Common Authentication Technology Overview (Обзор распространенных технологий аутентификации)  
 1510 — The Kerberos Network Authentication Service (V5) (Служба сетевой аутентификации Kerberos (V5))  
 1509 — Generic Security Service API : C-bindings (Интерфейс прикладных программ обобщенной службы безопасности)  
 1507 — DASS — Distributed Authentication Security Service (DASS — распределенная служба аутентификации и безопасности)  
 1457 — Security Label Framework for the Internet (Инфраструктура безопасных меток для Internet)  
 1455 — Physical Link Security Type of Service (Тип обслуживания защищенного физического канала связи)  
 1424 — Privacy Enhancement for Internet Electronic Mail: Part IV: Key Certification and Related Services (Усиление секретности для электронной почты Internet, часть IV: сертификация ключей и связанные службы)  
 1423, 1115 — Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifiers (Усиление секретности для электронной почты Internet, часть III: алгоритмы, режимы и идентификаторы)  
 1422, 1114 — Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management (Усиление секретности для электронной почты Internet, часть II: управление ключами на основе сертификатов)  
 1421, 1113, 989 — Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures (Усиление секретности для электронной почты Internet, часть II: шифрование сообщений и процедуры аутентификации)  
 1355 — Privacy and Accuracy Issues in Network Information Center Databases (Вопросы секретности и точности информации в базах данных сетевого информационного центра)  
 1281 — Guidelines for the Secure Operation of the Internet (Принципы выполнения защищенных операций в Internet)  
 1170 — Public key standards and licenses (Стандарты открытых ключей и лицензирование)  
 1135 — Helminthiasis of the Internet (Заражение червями в Internet)  
 1108 — US Department of Defense Security Options for the Internet Protocol (Параметры безопасности протоколов Internet министерства обороны США)  
 1040 — Privacy enhancement for Internet electronic mail: Part I: Message encipherment and authentication procedures (Усиление секретности для электронной почты Internet, часть II: шифрование сообщений и процедуры аутентификации)  
 1038 — Draft revised IP security option (Проект пересмотра параметров безопасности протокола IP)  
 1004 — Distributed-protocol authentication scheme (Схема аутентификации в распределенном протоколе)  
 972 — Password Generator Protocol (Протокол генерации паролей)

#### **10б. Алгоритмы шифрования, аутентификации и обмена ключами**

- 2631 — Diffie-Hellman Key Agreement Method (Метод согласования ключей Диффи-Хелмана)  
 2612 — The CAST-256 Encryption Algorithm (Алгоритм шифрования CAST-256)  
 2286 — Test Cases for HMAC-RIPemd160 and HMAC-RIPemd128 (Контрольный пример для тестирования функций HMAC-RIPemd160 и HMAC-RIPemd128)

- 2268 — A Description of the RC2(r) Encryption Algorithm (Описание алгоритма шифрования RC2(r))
- 2202 — Test Cases for HMAC-MD5 and HMAC-SHA-1 (Контрольный пример для тестирования функций HMAC-MD5 и HMAC-SHA-1)
- 2144 — The CAST-128 Encryption Algorithm (Алгоритм шифрования CAST-128)
- 2040 — The RC5, RC5-CBC, RC5-CBC-Pad, and RC5-CTS Algorithms (Алгоритмы RC5, RC5-CBC, RC5-CBC-Pad и RC5-CTS)
- 1810 — Report on MD5 Performance (Отчет по производительности алгоритма MD5)
- 1321 — The MD5 Message-Digest Algorithm (Алгоритм составления дайджеста сообщения MD5)
- 1320, 1186 — The MD4 Message-Digest Algorithm (Алгоритм составления дайджеста сообщения MD4)
- 1319 — The MD2 Message-Digest Algorithm (Алгоритм составления дайджеста сообщения MD2)

#### **10c. Защищенный протокол IP (IPSec)**

- 2709 — Security Model with Tunnel-mode IPsec for NAT Domains (Модель безопасности для доменов NAT при использовании туннельного режима протокола IPsec)
- 2451 — The ESP CBC-Mode Cipher Algorithms (Алгоритмы шифрования для ESP в режиме CBC)
- 2412 — The OAKLEY Key Determination Protocol (Протокол определения ключей OAKLEY)
- 2411 — IP Security Document Roadmap (Схема документации по защищенному протоколу IP)
- 2410 — The NULL Encryption Algorithm and Its Use With IPsec (Алгоритм нулевого шифрования и его использование в протоколе IPsec)
- 2409 — The Internet Key Exchange (IKE) (Протокол обмена ключами в Internet IKE)
- 2408 — Internet Security Association and Key Management Protocol (ISAKMP) (Сообщество безопасности в Internet и протокол управления ключами)
- 2407 — The Internet IP Security Domain of Interpretation for ISAKMP (Адаптация домена защищенного протокола IP для ISAKMP)
- 2406, 1827 — IP Encapsulating Security Payload (ESP) (Инкапсуляции зашифрованных данных в протоколе IP)
- 2405 — The ESP DES-CBC Cipher Algorithm With Explicit IV (Алгоритм шифрования DES в режиме цепочки связанных блоков (CBC) с явной инициализацией вектора)
- 2404 — The Use of HMAC-SHA-1-96 within ESP and AH (Использование алгоритма HMAC-SHA-1-96 в ESP и заголовке аутентификации)
- 2403 — The Use of HMAC-MD5-96 within ESP and AH (Использование алгоритма HMAC-MD5-96 в ESP и заголовке аутентификации)
- 2402, 1826 — IP Authentication Header (Заголовок аутентификации протокола IP)
- 2401, 1825 — Security Architecture for the Internet Protocol (Структура системы безопасности для протокола IP)
- 2104 — HMAC: Keyed-Hashing for Message Authentication (HMAC: хеширование ключей для аутентификации сообщений)
- 2085 — HMAC-MD5 IP Authentication with Replay Prevention (Алгоритм аутентификации HMAC-MD5 протокола IP с невозможностью воспроизведения последовательности действий)
- 1852 — IP Authentication using Keyed SHA (Аутентификация протокола IP с использованием безопасного алгоритма хеширования ключей)

- 1851 — The ESP Triple DES Transform (Адаптация алгоритма тройного DES для инкапсуляции зашифрованных данных)
- 1829 — The ESP DES-CBC Transform (Адаптация алгоритма DES для инкапсуляции зашифрованных данных)
- 1828 — IP Authentication using Keyed MD5 (Аутентификация протокола IP на основе алгоритма MD5 с ключами)

## **11. Опыт использования сети и демонстрация ее работы**

- 2123 — Traffic Flow Measurement: Experiences with NeTraMet (Измерение потока трафика: опыт использования NeTraMet)
- 1435 — IESG Advice from Experience with Path MTU Discovery (Мнение руководящей инженерной группы Internet по поводу опыта определения MTU по пути следования пакетов)
- 1306 — Experiences Supporting By-Request Circuit-Switched T3 Networks (Опыт поддержки коммутации по запросу в сетях Т3)
- 967 — All victims together (Все жертвы вместе)
- 573 — Data and file transfer: Some measurement results (Передача данных и файлов: некоторые результаты измерений)
- 525 — MIT-MATHLAB meets UCSB-OLS - an example of resource sharing (Пример совместного использования ресурсов в системах MATHLAB Массачусетского технологического института и OLS Калифорнийского университета в Санта-Барбара)
- 439 — PARRY encounters the DOCTOR (Перри встретил доктора)
- 420 — CCA ICCC weather demo (Демонстрация прогноза погоды ССА ICCC)
- 372 — Notes on a Conversation with Bob Kahn on the ICCC (Замечания по поводу интервью с Бобом Канном в ICCC)
- 364 — Serving remote users on the ARPANET (Обслуживание удаленных пользователей в ARPANET)
- 302 — Exercising The ARPANET (Применение ARPANET)
- 231 — Service center standards for remote usage: A user's view (Стандарты для дистанционного использования сервисного центра: точка зрения пользователя)
- 227 — Data transfer rates (Rand/UCLA) (Скорости передачи данных (Rand/UCLA))
- 113 — Network activity report: UCSB/Rand (Отчет по активности сети (UCSB/Rand))
- 89 — Some historic moments in networking (Несколько исторических моментов организации сетей)
- 4 — Network timetable (Сетевое расписание)

## **12. Документация сетевого центра**

- 30, 27, 24, 10, 3 — Documentation Conventions (Соглашения по написанию документации)

## **13. Стандарты протоколов, принятые другими группами, заинтересованными в использовании Internet**

### **13a. ANSI**

- 183 — EBCDIC codes and their mapping to ASCII (Кодировка EBCDIC и ее преобразование в кодировку ASCII)
- 20 — ASCII format for network interchange (Формат ASCII для сетевого обмена)

### **13b. NRC**

- 942 — Transport protocols for Department of Defense data networks (Транспортные протоколы для сетей передачи данных министерства обороны США)  
939 — Executive summary of the NRC report on transport protocols for Department of Defense data networks (Резюме руководства по поводу отчета NRC о транспортных протоколах для сетей передачи данных министерства обороны США)

### **13c. ISO**

- 1698 — Octet Sequences for Upper-Layer OSI to Support Basic Communications Applications (Последовательности октетов для протоколов OSI верхнего уровня, предназначенных для поддержки основных приложений передачи данных)  
1629, 1237 — Guidelines for OSI NSAP Allocation in the Internet (Правила выделения ресурса OSI NSAP)  
1575, 1139 — An Echo Function for CLNP (ISO 8473) (Эхо-функция для CLNP (ISO 8473))  
1574 — Essential Tools for the OSI Internet (Основные средства реализации объединенной сети на основе протоколов OSI)  
1561 — Use of ISO CLNP in TUBA Environments (Использование протокола CLNP ISO в окружении TUBA)  
1330 — Recommendations for the Phase I Deployment of OSI Directory Services (X.500) and OSI Message Handling Services (X.400) within the ESNET Community (Рекомендации по первой фазе развертывания службы каталогов OSI (X.500) и службы обработки сообщений OSI (X.400) в сообществе ESNET)  
1238, 1162 — CLNS MIB for use with Connectionless Network Protocol (ISO 8473) and End System to Intermediate System (ISO 9542) (База управляющей информации CLNS для использования с протоколом сетевого обслуживания, не требующим установки соединения (ISO 8473) и протоколом передачи информации от конечной до промежуточной системы (ISO 9542))  
1223 — OSI CLNS and LLC1 protocols on Network Systems HYPERchannel (Протоколы OSI CLNS и LLC1 в сетевой системе HYPERchannel)  
1008 — Implementation guide for the ISO Transport Protocol (Руководство по реализации транспортного протокола ISO)  
1007 — Military supplement to the ISO Transport Protocol (Дополнения к транспортному протоколу ISO, предназначенные для применения в военных целях)  
995 — End System to Intermediate System Routing Exchange Protocol for use in conjunction with ISO 8473 (Использование протокола обмена маршрутной информацией между конечной и промежуточной системами совместно с протоколом ISO 8473)  
994 — Final text of DIS 8473, Protocol for Providing the Connectionless-mode Network Service (Окончательный вариант протокола для обеспечения работы сетевых служб, не требующих установки соединения (DIS 8473))  
982 — Guidelines for the specification of the structure of the Domain Specific Part (DSP) of the ISO standard NSAP address (Основные особенности спецификации структуры доменной части адреса NSAP стандарта ISO)

- 941 — Addendum to the network service definition covering network layer addressing (Дополнения к определению сетевой службы, относящееся к системе адресации сетевого уровня)
- 926 — Protocol for providing the connectionless mode network services (Протокол, обеспечивающий работу сетевых служб, не требующих установки соединения)
- 905 — ISO Transport Protocol specification ISO DP 8073 (Спецификация транспортного протокола ISO DP 8073)
- 892 — ISO Transport Protocol specification (Спецификация транспортного протокола ISO)
- 873 — Illusion of vendor support (Иллюзия поддержки производителем)

## **14. Взаимодействие с другими приложениями и протоколами**

### **14a. Преобразование протоколов и мосты**

- 1086 — ISO-TP0 bridge between TCP and X.25 (Мост ISO-TP0 между протоколами TCP и X.25)
- 1029 — More fault tolerant approach to address resolution for a Multi-LAN system of Ethernets (Устойчивый к отказам оборудования подход к проблеме преобразования адреса для локальных систем, содержащих несколько сетей Ethernet)

### **14b. Туннелирование и разбиение на уровни**

- 2661 — Layer Two Tunneling Protocol “L2TP” (Протокол туннелирования второго уровня L2TP)
- 2556 — OSI connectionless transport services on top of UDP Applicability Statement for Historic Status (Заявление для восстановления исторической справедливости по поводу применимости протокола UDP в транспортных службах OSI, не требующих установки соединения)
- 2353 — APPN/HPR in IP Networks APPN Implementers' Workshop Closed Pages Document (Протоколы APPN/HPR в IP-сетях. Окончательный вариант документа семинара разработчиков протокола APPN)
- 2166 — APPN Implementer's Workshop Closed Pages Document DLSw v2.0 Enhancements (Расширения протокола DLSw версии 2.0. Окончательный вариант документа семинара разработчиков протокола APPN)
- 2126, 1859, 1006 — ISO Transport Service on top of TCP (ITOT) (Использование протокола TCP транспортными службами ISO (ITOT))
- 2114, 2106 — Data Link Switching: Client Access Protocol (Коммутация каналов: протокол доступа клиента)
- 1795, 1434 — Data Link Switching: Switch-to-Switch Protocol AIW DLSw RIG: DLSw Closed Pages, DLSw Standard Version 1 (Коммутация каналов: протокол взаимодействия между двумя коммутаторами AIW DLSw RIG: окончательный вариант стандарта DLSw версии 1)
- 1791 — TCP And UDP Over IPX Networks With Fixed Path MTU (Использование протоколов TCP и UDP в сетях IPX с фиксированным значением MTU)
- 1634, 1551, 1362 — Novell IPX Over Various WAN Media (IPXWAN) (Использование протокола Novell IPX в различных распределенных средах передачи данных (IPXWAN))
- 1613 — Cisco Systems X.25 over TCP (XOT) (Использование протокола X.25 компании Cisco Systems в сети на основе протокола TCP (ХОТ))
- 1538 — Advanced SNA/IP: A Simple SNA Transport Protocol (Расширенный протокол SNA/IP: простой транспортный протокол SNA)

- 1356 — MultiProtocol Interconnect on X.25 and ISDN in the Packet Mode (Многопротокольное взаимодействие между сетями X.25 и ISDN в пакетном режиме)
- 1240 — OSI connectionless transport services on top of UDP: Version 1 (Использование протокола UDP в транспортных службах OSI, не требующих установки соединения)
- 1234 — Tunneling IPX traffic through IP networks (Туннелирование IPX-трафика в IP-сетях)
- 1085 — ISO presentation services on top of TCP/IP based internets (Реализация служб представления данных ISO в объединенной сети на основе протокола TCP/IP)
- 1070 — Use of the Internet as a subnet for experimentation with the OSI network layer (Использование Internet в качестве подсети при экспериментировании с сетевым уровнем модели OSI)
- 983 — ISO transport arrives on top of the TCP (Использование протокола TCP транспортными службами ISO)

#### **14c. Преобразование имен, адресов и идентификаторов**

- 1439 — The Uniqueness of Unique Identifiers (Вопросы уникальности уникальных идентификаторов)
- 1236 — IP to X.121 address mapping for DDN (Преобразование адресов протокола IP и X.121 для сети цифровой передачи данных)
- 1069 — Guidelines for the use of Internet-IP addresses in the ISO Connectionless-Mode Network Protocol (Основные принципы использования IP-адресов в сетевом протоколе ISO, не требующем установки соединения)

### **15. Разнообразная информация**

#### **15a. Общие вопросы**

- 2664, 1594, 1325, 1206, 1177 — FYI on Questions and Answers — Answers to Commonly Asked “New Internet User” Questions (Ответы на часто задаваемые вопросы новыми пользователями Internet)
- 2636, 2604 — Wireless Device Configuration (OTASP/OTAPA) via ACAP (Конфигурирование беспроводных устройств (OTASP/OTAPA) с помощью протокола ACAP)
- 2635 — DON'T SPEW A Set of Guidelines for Mass Unsolicited Mailings and Postings (spam\*) (Не надо мусорить. Основные правила, которые надо соблюдать при массовой рассылке электронных сообщений и публикации информации (спама))
- 2626 — The Internet and the Millennium Problem (Year 2000) (Internet и проблема 2000 года)
- 2555 — 30 Years of RFCs (К 30-летию документов RFC)
- 2468 — I REMEMBER IANA (Вспоминая IANA)
- 2441 — Working with Jon, Tribute delivered at UCLA, October 30, 1998 (Работая вместе с Джоном. Письмо — дань памяти Джону Постелу, пришедшее в Калифорнийский университет в Лос-Анджелесе 30 октября 1998 года)
- 2351 — Mapping of Airline Reservation, Ticketing, and Messaging Traffic over IP (Схема резервирования и продажи билетов на самолеты и передача трафика сообщений по IP-сети)
- 2350 — Expectations for Computer Security Incident Response (Надежды на отклик по поводу инцидента, связанного с компьютерной безопасностью)

- 2309 — Recommendations on Queue Management and Congestion Avoidance in the Internet (Рекомендации по управлению очередями и избежанию перегрузок в Internet)
- 2235 — Hobbes' Internet Timeline (Временная диаграмма Хоббса)
- 2234 — Augmented BNF for Syntax Specifications: ABNF (Расширенная форма Бэкуса-Наура для спецификации синтаксиса (ABNF))
- 2151, 1739 — A Primer On Internet and TCP/IP Tools and Utilities (Руководство по широко распространенным средствам и утилитам Internet и протокола TCP/IP)
- 2150 — Humanities and Arts: Sharing Center Stage on the Internet (Люди и искусство: распределение центральных ролей в Internet)
- 2057 — Source Directed Access Control on the Internet (Документ, посвященный проблеме управления доступом в Internet)
- 1983, 1392 — Internet Users' Glossary (Словарь терминов для пользователей Internet)
- 1958 — Architectural Principles of the Internet (Принципы структурирования в Internet)
- 1941, 1578 — Frequently Asked Questions for Schools (Список часто задаваемых вопросов для учащихся)
- 1935 — What is the Internet, Anyway? (Что такое Internet вообще?)
- 1865 — EDI Meets the Internet Frequently Asked Questions about Electronic Data Interchange (EDI) on the Internet (Электронный обмен данными приходит в Internet. Список часто задаваемых вопросов по поводу электронного обмена данными в Internet)
- 1855 — Netiquette Guidelines (Основные принципы сетевого этикета)
- 1775 — To Be "On" the Internet (Как подключиться к Internet)
- 1758, 1417, 1295 — NADF Standing Documents: A Brief Overview (Краткий обзор документов Северо-Американского каталогного форума)
- 1746 — Ways to Define User Expectations (Пути определения ожиданий пользователей)
- 1709 — K-12 Internetworking Guidelines (Основные принципы построения объединенных сетей сообщества K-12)
- 1691 — The Document Architecture for the Cornell Digital Library (Структура документов для цифровой библиотеки Корнельского университета)
- 1633 — Integrated Services in the Internet Architecture: an Overview (Структура интегрированных служб в Internet: обзор)
- 1580 — Guide to Network Resource Tools (Руководство по использованию средств доступа к сетевым ресурсам)
- 1501 — OS/2 User Group (Группа пользователей OS/2)
- 1498 — On the Naming and Binding of Network Destinations (К вопросу об именах и привязках сетевых получателей)
- 1470, 1147 — FYI on a Network Management Tool Catalog: Tools for Monitoring and Debugging TCP/IP Internets and Interconnected Devices (Информационный документ по поводу средства управления сетевым каталогом: средства наблюдения и отладки объединенных сетей на основе протокола TCP/IP и взаимодействующих устройств)
- 1462 — FYI on "What is the Internet?" (Информационный документ посвященный проблеме "Что такое Internet?")
- 1453 — A Comment on Packet Video Remote Conferencing and the Transport/Network Layers (Комментарии по поводу удаленной пакетной видеоконференции и транспортному/сетевому уровнях)
- 1432 — Recent Internet Books (Новые книги об Internet)

- 1402, 1290 — There's Gold in them thar Networks! or Searching for Treasure in all the Wrong Places (Руководство по поиску информации в Internet)
- 1400 — Transition and Modernization of the Internet Registration Service (Переход и модернизация службы регистрации Internet)
- 1359 — Connecting to the Internet — What Connecting Institutions Should Anticipate (Подключение к Internet: какой тип подключения нужен для организации)
- 1345 — Character Mnemonics and Character Sets (Символьные мнемоники и наборы символов)
- 1336, 1251 — Who's Who in the Internet: Biographies of IAB, IESG and IRSG Members (Кто есть кто в Internet: биографии сотрудников IAB, IESG и IRSG)
- 1324 — A Discussion on Computer Network Conferencing (Дискуссии на конференции, посвященной компьютерным сетям)
- 1302 — Building a Network Information Services Infrastructure (Создание инфраструктуры сетевых информационных служб)
- 1300 — Remembrances of Things Past (Вспоминая прошлое)
- 1296 — Internet Growth (1981–1991) (Рост Internet с 1981 по 1991 год)
- 1291 — Mid-Level Networks Potential Technical Services (Потенциал сетевых технических служб среднего уровня)
- 1259 — Building the open road: The NREN as test-bed for the national public network (Создание открытой магистрали: государственная научно-исследовательская и образовательная сеть как испытательная модель открытой общенациональной компьютерной сети)
- 1242 — Benchmarking terminology for network interconnection devices (Терминология по производительности взаимодействующих сетевых устройств)
- 1208 — Glossary of networking terms (Словарь сетевых терминов)
- 1207 — FYI on Questions and Answers: Answers to commonly asked “experienced Internet user” questions (Ответы на часто задаваемые вопросы опытными пользователями Internet)
- 1199, 1099 — Request for Comments Summary Notes: 1100–1199 (Сводка документов RFC с номерами 1100–1199)
- 1192 — Commercialization of the Internet summary report (Отчетный доклад по поводу коммерциализации Internet)
- 1181 — RIPE Terms of Reference (Компетенция Европейской континентальной сети TCP/IP)
- 1180 — TCP/IP tutorial (Учебник по протоколу TCP/IP)
- 1178 — Choosing a name for your computer (Как выбрать имя для своего компьютера)
- 1173 — Responsibilities of host and network managers: A summary of the “oral tradition” of the Internet (Ответственность администраторов узлов и сетей: краткая сводка неписанных правил Internet)
- 1169 — Explaining the role of GOSIP (Объяснение роли поддерживаемых государством спецификаций для протоколов OSI)
- 1167 — Thoughts on the National Research and Education Network (Размышления по поводу государственной научно-исследовательской и образовательной сети)
- 1118 — Hitchhikers guide to the Internet (Руководство по Internet для подключившихся пользователей)
- 1015 — Implementation plan for interagency research Internet (План реализации для межведомственного исследования Internet)

- 992 — On communication support for fault tolerant process groups (К вопросу о поддержке устойчивого к отказам взаимодействия между рабочими группами)
- 874 — Critique of X.25 (Критические замечания по поводу протокола X.25)
- 531 — Feast or famine? A response to two recent RFC's about network information (Все или ничего? Ответ на два последний документа RFC по поводу сетевой информации)
- 473 — MIX and MIXAL? (MIX или MIXAL?)
- 472 — Illinois' reply to Maxwell's request for graphics information (NIC 14925) (Ответ университета штата Иллинойс на запрос Максвелла графической информации (NIC 14925))
- 429 — Character Generator Process (Процесс генерации символов)
- 408 — NETBANK
- 361 — Daemon Processes on Host 106 (Фоновые процессы на узле 106)
- 313 — Computer based instruction (Машинные команды)
- 256 — IMPSYS change notification (Уведомление об изменении IMPSYS)
- 225 — Rand/UCSB network graphics experiment (Эксперимент по обмену графическими данными по сети Rand/UCSB)
- 219 — User's view of the datacomputer (Мнение пользователя по поводу дейтакомпьютеров)
- 187 — Network/440 Protocol Concept (Концепция протокола Network/440)
- 169 — Computer networks (Компьютерные сети)
- 146 — Views on issues relevant to data sharing on computer networks (Мнение по поводу совместного использования данных в компьютерных сетях)
- 13 — Zero Text Length EOF Message (Сообщение о конце файла, содержащее текст нулевой длины)

### **15b. Библиография**

- 2007 — Catalogue of Network Training Materials (Каталог сетевых учебных материалов)
- 1463 — FYI on Introducing the Internet — A Short Bibliography of Introductory Internetworking Readings (Информационный документ, посвященный введению в Internet: краткий список литературы, предназначенный для начального ознакомления с Internet)
- 1175 — FYI on where to start: A bibliography of internetworking information (Информационный документ, посвященный проблеме начала обучения: список литературы, содержащей общую информацию об объединенной сети)
- 1012 — Bibliography of Request For Comments 1 through 999 (Библиография документов RFC с 1 по 999 номер)
- 829 — Packet satellite technology reference sources (Справочные материалы по спутниковой технологии коммутации пакетов)
- 290 — Computer networks and data sharing: A bibliography (Компьютерные сети и совместное использование данных: библиографический список)
- 243 — Network and data sharing bibliography (Библиографический список, посвященный сетям и совместному использованию данных)

### **15c. Юмористические документы RFC**

- 2551 — The Roman Standards Process — Revision III (Римский процесс стандартизации, выпуск III)
- 2550 — Y10K and Beyond (Y10K и выше)
- 2549 — IP over Avian Carriers with Quality of Service (Использование протокола IP в птичьем полете с качественным обслуживанием)

- 2325 — Definitions of Managed Objects for Drip-Type Heated Beverage Hardware Devices using SMIv2 (Определение контролируемых объектов для устройств самогоноварения с использованием SMIv2)
- 2324 — Hyper Text Coffee Pot Control Protocol (HTCPCP/1.0) (Гипертекстовый протокол управления кофейником)
- 2323 — IETF Identification and Security Guidelines (Основные принципы безопасности и идентификации IETF)
- 2322 — Management of IP numbers by peg-dhcp (Управление IP-номерами с помощью протокола peg-dhcp)
- 2321 — RITA — The Reliable Internetwork Troubleshooting Agent (RITA — надежный агент, отыскивающий неисправности в сети)
- 2100 — The Naming of Hosts (Присваивание имен узлам сети)
- 1927 — Suggested Additional MIME Types for Associating Documents (Предлагаемые дополнительные типы MIME для вложенных документов)
- 1926 — An Experimental Encapsulation of IP Datagrams on Top of ATM (Экспериментальная инкапсуляция IP-дейтаграмм в ячейки ATM)
- 1925 — The Twelve Networking Truths (Двенадцать сетевых истин)
- 1924 — A Compact Representation of IPv6 Addresses (Компактное представление адреса протокола IPv6)
- 1882 — The 12-Days of Technology Before Christmas (Двенадцать технологических дней перед Рождеством)
- 1776 — The Address is the Message (Адрес — это сообщение)
- 1607 — A VIEW FROM THE 21ST CENTURY (Взгляд из XXI века)
- 1606 — A Historical Perspective On The Usage Of IP Version 9 (Исторические перспективы использования протокола IP версии 9)
- 1605 — SONET to Sonnet Translation (Трансляция SONET в сонеты)
- 1438 — Internet Engineering Task Force Statements Of Boredom (SOBs) (Заявление IETF по поводу скуки)
- 1437 — The Extension of MIME Content-Types to a New Medium (Расширение заголовка MIME Content-Types для поддержки новых сред)
- 1313 — Today's Programming for KRFC AM 1313 Internet Talk Radio (Современные программы для речевого радио Internet KRFC AM 1313 )
- 1217 — Memo from the Consortium for Slow Commotion Research (CSCR) (Меморандум консорциума по поводу медленных суматошных исследований)
- 1216 — Gigabit network economics and paradigm shifts (Сдвиги в экономике гигабитовых сетей и парадигм)
- 1149 — Standard for the transmission of IP datagrams on avian carriers (Стандарт для передачи IP-дейтаграмм в птичьем полете)
- 1121 — Act one — the poems (Первый акт — стихотворение)
- 1097 — Telnet subliminal-message option (Параметр протокола Telnet для подсознательных сообщений)
- 968 — Twas the night before start-up (Бессонная ночь накануне пуска)
- 748 — Telnet randomly-lose option (Параметр протокола TELNET, позволяющий удалять символы случайным образом)
- 527 — ARPAWOCKY

## **16. Список неиспользуемых номеров документов RFC**

2727, 2726, 2725, 2708, 2707, 2700, 2699, 2600, 2599, 2576, 1849, 1840, 1839, 1260, 1182, 1061, 853, 723, 715, 711, 710, 709, 693, 682, 676, 673, 670, 668, 665, 664, 650, 649, 648, 646, 641, 639, 605, 583, 575, 572, 564, 558, 554, 541, 540, 536, 517, 507, 502, 484, 481, 465, 444, 428, 427, 424, 397, 383, 380, 375, 358, 341, 337, 284, 279, 277, 275, 272, 262, 261, 260, 259, 258, 257, 248, 244, 220, 201, 159, 92, 26, 14

# Приложение 2

## Словарь терминов и аббревиатур по сетевым технологиям

### Терминология TCP/IP

Подобно любой области науки и технологии, в области сетевых технологий существует своя терминология и жаргонный язык. Этот язык весьма сложен для понимания и изучения, поскольку он представляет собой причудливую смесь жаргонных словечек, названий протоколов и аббревиатур. Для непосвященных диалог между двумя профессионалами напоминает бессмысленную белиберду, в основном состоящую из акронимов, которые употребляются по каждому удобному случаю. И даже после изучения этой книги некоторые специфические термины могут остаться для читателя достаточно туманными. Проблема усугубляется как огромным количеством терминов, так и тем, что некоторые из них плохо определены.

Приведенный в этом приложении словарь терминов поможет решить описанную проблему. Для каждого термина, который имеет какое-то отношение к Internet, приведено короткое пояснение. Этот словарь терминов не предназначен для использования в качестве учебного пособия новичками. Он задумывался как краткий справочник для специалистов по сетевым технологиям, к которому они должны обращаться за разъяснением того или иного термина или акронима. Скорее всего для читателей книги словарь терминов может оказаться более полезным в качестве справочника, к которому они должны обращаться после изучения отдельных тем.

### Словарь сетевых терминов и аббревиатур, собранных в алфавитном порядке

#### 1000Base-T

Название стандарта передачи данных со скоростью 1000 Мбит/с (1 Гбит/с) по сети Ethernet с использованием неэкранированной (скрученной) витой пары.

#### 100Base-T

Название стандарта передачи данных со скоростью 100 Мбит/с по сети Ethernet с использованием неэкранированной (скрученной) витой пары. Используется также более специфичный термин *100Base-TX*.

#### 10Base2

Название стандарта передачи данных по сети Ethernet с использованием тонкого коаксиального кабеля.

**10Base5**

Название оригинального стандарта передачи данных по сети Ethernet с использованием толстого коаксиального кабеля.

**10Base-T**

Название стандарта передачи данных со скоростью 10 Мбит/с по сети Ethernet с использованием неэкранированной (скрученной) витой пары.

**127.0.0.1**

Адрес протокола IP, используемый для тестирования работоспособности узла сети (так называемый *адрес петли обратной связи*). Пакеты, посылаемые по этому адресу, обрабатываются локальным модулем протокола и никогда не покидают пределов локальной машины (т.е. не посылаются в сеть).

**576**

Минимальный размер дейтаграммы, которую должны обрабатывать все узлы сети и маршрутизаторы.

**802.3**

Стандарт IEEE для сети Ethernet.

**822**

Стандартный формат семейства протоколов TCP/IP для сообщений, посылаемых по электронной почте. Специалисты часто называют его “стандартом 822”, или “форматом 822”. Название возникло из-за того, что окончательный вариант стандарта сообщений электронной почты был опубликован в документе [RFC 822]. Формат “822” заменил ранее использовавшийся формат “733”.

**9180**

Стандартный размер блока MTU, использующийся для передачи IP-дейтаграмм по сети ATM.

**AAL5**

*ATM Adaptation Layer 5*, или *протокол адаптации ATM уровня 5*. Один из протоколов семейства ATM. Существует несколько уровней адаптации. Пятый уровень используется для передачи данных.

**ABR**

1) *Available Bit Rate*, или *доступная скорость передачи*. Название, используемое в протоколе ATM, для обозначения негарантированной скорости передачи данных. 2) *Area Border Router*, или *пограничный маршрутизатор зоны*. Маршрутизатор, использующийся для взаимодействия с маршрутизаторами других зон по протоколу OSPF.

**ACK**

Сокращение от слова *acknowledgement*, или *подтверждение приема*.

**ADSL**

*Asymmetric Digital Subscriber Line*, или *асимметричная цифровая абонентская линия*. Популярный вариант цифровой абонентской линии DSL. Технология для создания несимметричного доступа к сети, позволяющая абоненту принимать сигналы со скоростью до 8 Мбит/с, а передавать — со скоростью до 1 Мбит/с на расстояние до 7 км.

## **Advanced Networks and Services**

Компания, которая до 1995 года была владельцем магистрального канала Internet и обеспечивала его работоспособность.

### **AH**

*Authentication Header*, или *заголовок аутентификации*. Заголовок, который используется в протоколе IPsec для проверки подлинности отправителя дейтаграммы.

### **ANS**

Аббревиатура компании *Advanced Networks and Services*.

### **ANSI**

*American National Standards Institute*, или *Национальный институт стандартизации США*. Группа, которая определяет стандарты США в области информационных технологий. Специалисты ANSI принимали участие в создании стандартов сетевых протоколов.

### **ANSNET**

Глобальная сеть компании ANS, которая до 1995 года являлась магистралью Internet.

### **API**

*Application Program Interface*, или *интерфейс прикладных программ*. Спецификация функций и процедур, которые может вызвать прикладная программа для выполнения определенных действий, например для взаимодействия по сети с другой программой. Для обмена данными через объединенную сеть чаще всего используется API сокетов.

### **API сокетов (socket API)**

Набор библиотечных процедур, которые вызывают прикладные программы для выполнения обмена данными по сети с помощью семейства протоколов TCP/IP. Данный термин был введен в качестве абстрактного понятия в системе Unix.

### **ARP**

*Address Resolution Protocol*, или *протокол преобразования адреса*. Протокол семейства TCP/IP, используемый для динамической привязки высокоуровневых IP-адресов к низкоуровневым физическим адресам сетевого оборудования. Действие протокола ARP ограничено рамками одной физической сети, а использоваться он может только в тех сетях, которые поддерживают на аппаратном уровне широковещательную передачу пакетов.

### **ARPA**

*Advanced Research Projects Agency*, или *управление перспективного планирования научно-исследовательских работ*. Государственная организация, которая финансировала работы, связанные с сетью ARPANET, а после ликвидации последней — работы, связанные с глобальной сетью Internet. Внутри ARPA была группа, ответственная за работу сети ARPANET. Вначале ее называли IPTO (*Information Processing Techniques Office*, или *отдел методов обработки информации*), а затем — ISTO (*Information Systems Technology Office*, или *технологический отдел информационных систем*). Через много лет ARPA была переименована в *DARPA*.

## **ARPANET**

Первая сеть, связавшая компьютеры, находящиеся друг от друга на большом расстоянии, финансируемая ARPA (впоследствии DARPA) и запущенная в эксплуатацию компанией BBN. С 1969 по 1990 года она использовалась как испытательный полигон для проведения научных исследований в области сетей, а также как основная магистральная сеть во время разработки Internet. Основу сети ARPANET составляли самостоятельные коммутаторы пакетов, связанные выделенными линиями.

## **ARQ**

*Automatic Repeat Request*, или *автоматический запрос на повторение*. Способ повышения надежности передачи данных с помощью протокола, поддерживающего положительные или отрицательные сигналы подтверждения приема в сочетании с механизмом повторной передачи. Отправитель автоматически повторяет запрос, если на него не получен ответ.

## **AS**

*Autonomous System*, или *автономная система (AC)*. Набор сетей и маршрутизаторов, управляемых одной административной единицей. В пределах автономной системы маршрутизаторы тесно взаимодействуют между собой, рассылая информацию о достижимости сетей, а также маршрутную информацию с помощью одного из протоколов внутреннего шлюза. Маршрутизаторы внутри автономной системы полностью доверяют информации, полученной друг от друга. Однако для взаимодействия двух автономных систем, необходимо назначить по одному специальному маршрутизатору в каждой АС, которые будут обмениваться друг с другом информацией о достижимости сетей.

## **ASN.1**

*Abstract Syntax Notation.1*, или *абстрактная синтаксическая запись версии 1*. Стандартный протокол представления данных ISO, используемый в протоколе SNMP для представления сообщений.

## **ATM**

*Asynchronous Transfer Mode*, или *асинхронный режим передачи*. Сетевая технология, ориентированная на установку соединения с получателем, в которой на самом нижнем уровне данные передаются в виде ячеек небольшого фиксированного размера (как правило, 53 октета). Использование технологии ATM имеет неоспоримое преимущество при передаче голосовых и видеоданных, а также данных, относящихся к одной базовой технологии.

## **ATMARP**

Протокол, используемый узлом сети для преобразования адреса при отправке IP-пакета по сети ATM.

## **AUI**

*Attachment Unit Interface*, или *интерфейс подключаемых устройств*. Соединитель, используемый для подключения толстого провода сети Ethernet.

## **base64**

Тип кодировки, используемой в стандарте MIME для пересылки нетекстовых данных (например, двоичных файлов) по электронной почте.

**BCP**

*Best Current Practice*, или *лучшие методики*. Название, присвоенное под множеству документов RFC, в которых описаны рекомендации IETF по поводу использования, конфигурирования или развертывания технологий объединенных сетей.

**BGP**

*Border Gateway Protocol*, или *протокол граничного шлюза*. Один из основных протоколов, используемых в Internet для обеспечения маршрутизации между автономными системами. Относится к классу протоколов внешнего шлюза. Существует четыре варианта протокола BGP. В настоящее время используется его последняя (четвертая) версия, или BGP-4.

**BISYNC**

*Binary Synchronous Communication*, или *двоичный синхронный протокол связи*. Один из ранних низкоуровневых протоколов связи, разработанный фирмой IBM и используемый для передачи данных по синхронным каналам связи. В отличие от большинства современных протоколов канального уровня, протокол BISYNC является байт-ориентированным. Это означает, что в нем используются специальные символы для обозначения начала и конца каждого фрейма. В современных коммерческих приложениях протокол BISYNC часто называют BSC.

**BNC**

Тип соединителя, который используется для подключения сетевой платы к тонкому проводу Ethernet.

**BOOTP**

*Bootstrap Protocol*, или *протокол начальной загрузки*. Протокол, используемый узлом сети при начальной загрузке для получения от сервера своих параметров, таких как IP-адрес и др.

**bps**

*bits per second*, или *бит в секунду*. Характеристика скорости передачи данных в канале связи.

**Bps**

*bytes per second*, или *байт в секунду*. Характеристика скорости передачи данных в канале связи.

**brouter**

*Bridging Router*, или *мост-маршрутизатор*. Устройство, которое работает как мост для одних протоколов и как маршрутизатор для других. Например, это устройство может служить мостом для протоколов DECNET и маршрутизатором для протокола IP.

**BSC**

*Binary Synchronous Communication*, или *двоичный синхронный протокол связи*. См. BISYNC.

**BSD UNIX**

*Berkeley Software Distribution UNIX*. Версия операционной системы UNIX, выпущенная в Калифорнийском университете в Беркли, а также одна из коммерческих систем, построенная на ее основе. В BSD UNIX была впервые включена поддержка протоколов семейства TCP/IP.

**CBT**

*Core Based Trees*, или *наращиваемые от центра деревья*. Протокол многоадресатной маршрутизации, управляемый по запросу, в котором выполняется построение совместно используемого дерева пересылки дейтаграмм.

**CCIRN**

*Coordinating Committee for Intercontinental Research Networking*, или *Координационный комитет по межконтинентальным научно-исследовательским сессиям*. Международная группа экспертов, которая координирует исследовательскую работу в области объединенных сетей.

**CCITT**

*Consultative Committee on International Telephony and Telegraphy*, или *Международный консультативный комитет по телеграфии и телефонии*. Старое название *Международного телекоммуникационного союза* (International Telecommunications Union, или ITU).

**CDDI**

*Copper Distributed Data Interface*, или *распределенный интерфейс передачи данных по медному кабелю*. Технология FDDI, приспособленная к использованию в сети на основе медного кабеля.

**CGI**

*Common Gateway Interface*, или *интерфейс общего шлюза*. Технология, которая используется Web-сервером для динамического создания Web-страниц в ответ на получение запроса от пользователя.

**CIDR**

*Classless Inter-Domain Routing*, или *бесклассовая междоменная маршрутизация*. Стандарт, в котором подробно описана бесклассовая система адресации и используемые при этом схемы маршрутизации.

**CL**

См. *служба, не требующая установки соединения*.

**CO**

См. *служба, требующая установки соединения*.

**COPS**

*Common Open Policy Service*, или *общая открытая служба правил*. Протокол, используемый совместно с протоколом резервирования ресурсов (RSVP) для проверки того, удовлетворяет ли полученный запрос принятым правилам и ограничениям.

**CRC**

*Cyclic Redundancy Code*, или *код циклического избыточного контроля*. Целое число, полученное в результате вычислений по определенному алгоритму, выполненных над последовательностью октетов сообщения, которые рассматриваются как целые числа. Полученное значение используется на приемной стороне для обнаружения ошибок, возникающих в результате передачи данных по каналу связи от одной машины до другой. Обычно в сети с коммутацией пакетов значение CRC вычисляется сетевым оборудованием и добавляется к пакету непосредственно перед его передачей по сети. После получения пакета сетевое оборудование

дование проверяет его целостность путем повторного вычисления значения CRC и сравнения его со значением CRC, находящимся в пакете. Поскольку алгоритм вычисления CRC сложнее обычного сложения, которое используется при вычислении контрольной суммы, с помощью CRC удается обнаружить большее количество ошибок, чем с помощью контрольной суммы. Ср. с *контрольной суммой*.

#### **CR-LF**

*Carriage Return — Line Feed*, или *возврат каретки — перевод строки*. Двухсимвольная управляющая последовательность, которая используется в качестве признака конца текстовой строки в протоколах, относящихся к уровню приложений, таких как *TELNET* и *SMTP*.

#### **CSMA/CD**

*Carrier Sense Multiple Access with Collision Detection*, или *множественный доступ с контролем несущей и обнаружением коллизий*. Характеристика сетевого оборудования, применение которого позволяет подключить к общей передающей среде несколько сетевых станций. Доступ к передающей среде со стороны сетевой станции может осуществляться только в те моменты, когда никакая другая станция не передает данные в сеть. Сетевое оборудование оснащается также специальным механизмом, позволяющим предотвратить одновременный доступ к среде со стороны нескольких станций. Технология CSMA/CD используется в сетях *Ethernet*.

#### **CSU/DSU**

*Channel Service Unit/Data Service Unit*, или *модуль обслуживания канала и данных*. Электронное устройство, с помощью которого компьютер или маршрутизатор подключаются к цифровой линии передачи данных, арендованной у телефонной компании. Несмотря на то что устройство выполняет две функции, обычно оно реализовано в виде единого электронного блока.

#### **DARPA**

*Defense Advanced Research Projects Agency*, или *управление перспективных исследовательских программ* при Министерстве обороны США. Ее прежнее название — ARPA.

#### **DCE**

*Data Communications Equipment*, или *аппаратура передачи данных*. Термин, используемый в описании стандартов протоколов ITU и относящийся к коммутационному оборудованию, на основе которого формируется сеть с коммутацией пакетов. Он введен для того, чтобы отличать устройства, формирующие сеть, от устройств, которые подключаются к сети (компьютеры или терминалы). См. также *DTE*.

#### **DDCMP**

*Digital Data Communication Message Protocol*, или *протокол передачи сообщений посредством цифровых данных*. Протокол канального уровня, используемый в первоначальной сетевой магистрали NSFNET.

#### **DDN**

*Defense Data Network*, или *оборонная сеть передачи данных*. Часть глобальной сети Internet, к которой подключены организации Министерства обороны США.

#### **DHCP**

*Dynamic Host Configuration Protocol*, или *протокол динамической конфигурации узла сети*. Протокол, используемый узлом сети при начальной загрузке

для получения от сервера всех необходимых параметров, таких как IP-адрес и др. Этот протокол особенно популярен у провайдеров Internet, поскольку позволяет назначить узлу сети временный IP-адрес.

#### **DiffServe**

*Differentiated Services*, или *дифференцированное обслуживание*. Новая система обслуживания протокола IP, принятая взамен устаревшего типа обслуживания. Система DiffServe обеспечивает до 64 разных типов обслуживания, которые имеют разный приоритет. В заголовке каждой дейтаграммы предусмотрено специальное поле, в котором отправитель указывает желаемый тип обслуживания.

#### **DNS**

*Domain Name System*, или *доменная система имен*. Система управления распределенной базой данных, работающая в оперативном режиме, которая используется для преобразования имен машин, представленных в удобном для восприятия человеком виде, в IP-адреса. Серверы DNS, соединенные посредством Internet, образуют иерархическое адресное пространство, которое позволяет передать права по назначению имен узлов сети и их адресов администрации сетевого центра. Кроме того, с помощью DNS выполняется преобразование имен серверов получателей электронной почты в IP-адреса.

#### **DS3**

Характеристика скорости передачи данных в выделенном канале связи, используемом в телефонии и приблизительно равном 45 Мбит/с.

#### **DSL**

*Digital Subscriber Line*, или *цифровая абонентская линия*. Ряд технологических решений, используемых для предоставления услуг по высокоскоростной передаче цифровых данных по медному кабелю, проложенному от местной телефонной станции до рабочего места локального абонента или офиса.

#### **DTE**

*Data Terminal Equipment*, или *терминальное оборудование*. Термин, используемый в описании стандартов протоколов ITU и относящийся к компьютерам и/или терминалам (т.е. к оконечным устройствам сети). Он введен для того, чтобы отличать оконечные устройства от устройств, формирующих сеть с коммутацией пакетов. См. также *DCE*.

#### **DVMRP**

*Distance Vector Multicast Routing Protocol*, или *дистанционно-векторный протокол многоадресатной маршрутизации*. Специальный протокол, который используется для распространения многоадресатных маршрутов.

#### **E.164**

Формат адреса, принятый ITU и используемый в сети ATM.

#### **EACK**

*Extended Acknowledgement*, или *расширенный сигнал подтверждения приема*. Синоним термина *SACK*.

#### **EGP**

*Exterior Gateway Protocol*, или *протокол внешнего шлюза*. Термин, применяемый к протоколу, который используется маршрутизатором, принадлежащим к одной автономной системе, для рассылки информации о достижимости внут-

ренних сетей маршрутизатору, принадлежащему к другой автономной системе. В настоящее время в качестве протокола внешнего шлюза чаще всего используется протокол *BGP-4*.

#### **EIA**

*Electronics Industry Association*, или *ассоциация электронной промышленности*. Организация, принимающая стандарты для электронной промышленности. В частности, ею принятые два популярных стандарта — RS232C и RS422, в которых описаны электрические параметры линии последовательной передачи данных, соединяющей терминал с компьютером или два компьютера между собой.

#### **ESP**

*Encapsulating Security Payload*, или *инкапсуляция зашифрованных данных*. Особый формат пакета, используемый в протоколе IPsec для передачи зашифрованной информации.

#### **Ethernet**

Популярная технология построения локальных сетей, изобретенная в исследовательском центре Пало Альто корпорации Xerox. В технологии Ethernet используется пассивный коаксиальный кабель, к которому подключаются все взаимодействующие между собой компоненты. В данной технологии используется идеология множественного доступа с контролем несущей и обнаружением коллизий (CSMA/CD), которая не гарантирует доставку пакетов. Совместными усилиями специалистов из Xerox Corporation, Digital Equipment Corporation и Intel Corporation был разработан и принят стандарт передачи данных по сети Ethernet со скоростью 10 Мбит/с. В первоначальном варианте сети Ethernet использовался толстый коаксиальный кабель. В последующих модернизациях использовался сначала тонкий коаксиальный кабель (*thinnet*), а затем кабель на основе витой пары (*10Base-T*).

#### **EUI-64**

Стандарт IEEE 64-битовой адресации второго уровня.

#### **FCCSET**

*Federal Coordinating Council for Science, Engineering, and Technology*, или *Федеральный координационный совет по науке, технике и технологиям*. Правительственная группа, известная своими отчетами, посвященными высокоскоростным вычислениям и исследованиям в области скоростной передачи данных по сетям.

#### **FDDI**

*Fiber Distribution Data Interface*, или *распределенный интерфейс передачи данных по волоконно-оптическим каналам*. Кольцевая сетевая технология с маркерным (эстафетным) доступом, используемая в волоконно-оптических каналах связи. В стандарте FDDI предусмотрена передача данных со скоростью 100 Мбит/с при использовании световой волны длиной 1300 нм. Длина сегмента сети может составлять около 200 км при условии использования повторителей через каждые 2 км.

#### **FDM**

*Frequency Division Multiplexing*, или *мультиплексная передача с частотным разделением каналов*. Метод передачи нескольких независимых потоков данных в одной среде (например, кабеле или эфире), при котором каждому потоку данных назначается своя несущая частота. Устройство, объединяющее потоки дан-

ных, называется мультиплексором, а устройство, разделяющее потоки данных, — демультиплексором. См. также *демультиплексирование* и *TDM*.

#### **FIN**

Специальный сегмент протокола TCP, используемый для закрытия соединения. Поскольку TCP-соединение двухстороннее, то для его закрытия каждая из сторон должна отправить друг другу сегмент FIN.

#### **FTP**

*File Transfer Protocol*, или *протокол передачи файлов*. Один из стандартов семейства протоколов TCP/IP, высокоуровневый протокол, который используется для передачи файлов от одной машины до другой. В качестве транспортного в протоколе FTP используется протокол TCP.

#### **FYI**

*For Your Information*, или *к вашему сведению*. Подмножество документов RFC, в которых содержится учебная информация либо общие сведения о семействе протоколов TCP/IP или подключении к Internet.

#### **gated**

*Gateway Daemon*, или *демон шлюза*. Программа, запускаемая на маршрутизаторе и выполняющая сбор маршрутной информации внутри одной автономной системы и анонсирование ее другой автономной системе. Для сбора маршрутной информации внутри автономной системы используется один из протоколов внутреннего шлюза, а для анонсирования этой информации другой автономной системе — один из протоколов внешнего шлюза.

#### **GGP**

*Gateway to Gateway Protocol*, или *межшлюзовой протокол*. Протокол, который ранее использовался шлюзами ядра сети для обмена маршрутной информацией. В настоящее время протокол GGP считается устаревшим и не рекомендуется к использованию.

#### **gopher**

Одна из первых текстовых информационных служб Internet, в которой использовалась система меню.

#### **GOSIP**

*Government Open Systems Interconnection Profile*, или *поддерживаемые государством спецификации для протоколов OSI*. Нормативный документ правительства США, содержащий названия организаций, которые могут использовать протоколы OSI в сетях, создаваемых после августа 1991 года. Несмотря на то что документ GOSIP создавался с целью ограничения использования семейства протоколов TCP/IP в объединенных сетях государственных и правительственные организаций, позднее к нему было выпущено пояснение, в котором указывалось, что эти организации могут продолжать использование протоколов TCP/IP.

#### **GRE**

*Generic Routing Encapsulation*, или *обобщенная маршрутная инкапсуляция*. Схема инкапсуляции данных в протоколе IP, которая включает в себя инкапсуляцию типа IP-в-IP как одну из возможных.

#### **H.323**

Рекомендация ITU для семейства протоколов, используемых в IP-телефонии.

## **HELLO**

Протокол, использовавшийся в первой магистральной сети NSFNET. Несмотря на то что протокол HELLO давно устарел, он представляет определенный интерес, поскольку в нем в качестве метрики маршрутизации используется величина задержки передачи пакета, а путь к получателю выбирается на основании минимальной задержки.

## **HELO**

Команда, которая используется для инициализации сеанса связи по протоколу *SMTP*.

## **HTML**

*Hypertext Markup Language*, или *язык гипертекстовой разметки*. Стандартный формат документа, используемый для создания Web-страниц.

## **HTTP**

*Hypertext Transfer Protocol*, или *протокол передачи гипертекста*. Протокол, который используется для передачи HTML-документов от Web-сервера к Web-браузеру.

## **IAB**

*Internet Architecture Board*, или *Архитектурный совет Internet*. Небольшая группа лиц, разрабатывающих стратегию развития глобальной сети Internet и руководящих исследованиями семейства протоколов TCP/IP. Ранее эта организация называлась *координационным советом Internet (Internet Activities Board)*. См. также *IETF*.

## **IAC**

*Interpret As Command*, или *интерпретировать как команду*. Управляющий символ, используемый в протоколе TELNET для выделения команд в потоке передаваемых данных.

## **IANA**

*Internet Assigned Number Authority*, или *Агентство по выделению имен и уникальных параметров протоколов Internet*. Ответственным за выделение IP-адресов и назначение констант, используемых в семействе протоколов TCP/IP, первоначально был один человек — Джон Постел. В 1999 году эта организация была преобразована в *ICANN*.

## **ICANN**

*Internet Corporation For Assigned Names and Numbers*, или *некоммерческая организация по назначению имен и адресов в Internet*. Новая организация, которой после смерти Джона Постела в 1999 году были переданы все права IANA.

## **ICCB**

*Internet Control and Configuration Board*, или *Совет по управлению и структуре Internet*. Организация, которая была предшественницей IAB.

## **ICMP**

*Internet Control Message Protocol*, или *протокол межсетевых управляющих сообщений*. Неотъемлемая часть протокола IP, которая выполняет обработку ошибок и рассылку управляющих сообщений. В частности, узлы сети и маршрутизаторы используют протокол ICMP для уведомления отправителя дейтаграммы.

о проблемах, связанных с ее доставкой до конечного получателя. В протокол ICMP также включена возможность эхо-запроса, которая используется для проверки достижимости получателя и его работоспособности.

#### **ICMPv6**

*Internet Control Message Protocol version 6.* Версия протокола ICMP, использующаяся в протоколе IPv6.

#### **IEN**

*Internet Engineering Notes*, или *экспериментальные рабочие документы Internet*. Набор документов, которые разрабатывались параллельно с RFC. Несмотря на то что эти документы давно устарели, в части из них можно найти обсуждения семейства протоколов TCP/IP и Internet, сделанные на раннем этапе, которые не включены в документы RFC.

#### **IESG**

*Internet Engineering Steering Group*, или *руководящая инженерная группа Internet*. Комитет, в который входят председатель инженерной группы IETF и руководители подразделений. IESG координирует действия рабочих групп IETF.

#### **IETF**

*Internet Engineering Task Force*, *инженерная группа по решению конкретной задачи Internet*. Группа лиц, подчиняющихся IAB, которые проектируют глобальную сеть Internet и внедряют новые протоколы семейства TCP/IP. IETF разделена на подразделения, каждое из которых возглавляет независимый руководитель. Подразделения, в свою очередь, состоят из рабочих групп.

#### **IGMP**

*Internet Group Management Protocol*, или *межсетевой протокол управления группами*. Протокол, который используется узлами сети для извещения локальных маршрутизаторов о подключении к указанной многоадресатной группе. Как только от данной группы отключиться последний абонент, маршрутизатор прекращает перенаправление дейтаграмм в локальную сеть, посланных по адресу группы.

#### **IGP**

*Interior Gateway Protocol*, или *протокол внутреннего шлюза*. Общий термин, используемый для обозначения протокола, с помощью которого в пределах автономной системы распространяется информация о достижимости сетей, а также маршрутная информация. Единого стандарта протокола IGP не существует, однако чаще всего для этой цели используется протокол *RIP*.

#### **IMP**

*Interface Message Processor*, или *интерфейсный процессор сообщений*. Первичное название устройства коммутации пакетов в сети ARPANET. В настоящее время этот термин иногда некорректно употребляется для обозначения коммутатора пакетов.

#### **InATMARP**

*Inverse ATM ARP*, или *инверсный протокол ATMARP*. Одна из частей протокола преобразования адреса, которая используется в сетях с множественным доступом, не поддерживающих режим широковещания, таких как ATM.

## **INOC**

*Internet Network Operations Center*, или *сетевой операционный центр Internet*. Первоначально это была группа лиц компании BBN, занимавшихся мониторингом сети и управлявших системой базовых маршрутизаторов. В настоящее время этот термин применяется к группе или организации, занимающейся мониторингом объединенной сети.

## **Internet**

Крупнейшая международная глобальная сеть, охватывающая более двухсот стран, в которой используются протоколы семейства TCP/IP. Это позволяет создать для пользователя впечатление, что он работает в одной большой виртуальной сети.

## **Internet-червь (Internet worm)**

Программа, которая автоматически распространяет сама себя по Internet и воспроизводит до бесконечности свои копии на узлах сети. Internet-червь придумал один из студентов, который имел в своем распоряжении локальную сеть с несколькими бездействующими компьютерами и выход в Internet.

## **IP**

*Internet Protocol*, или *протокол Internet*. Один из основных протоколов семейства TCP/IP, в котором определяется понятие IP-дейтаграммы как единицы передачи информации по объединенной сети и который обеспечивает негарантированную доставку пакетов, не требующую установки соединения с получателем. Неотъемлемой составной частью протокола IP является протокол ICMP, служащий для рассылки управляющих сообщений и сообщений об ошибке. Под аббревиатурой TCP/IP понимается название всего семейства протоколов, поскольку протоколы TCP и IP играют в нем основную роль.

## **IPng**

*Internet Protocol — the Next Generation*, или *протокол IP следующего поколения*. Термин, который применяется в отношении всех действий по разработке спецификации и стандартизации следующей версии протокола IP. См. также *IPv6*.

## **IPsec**

*IP Security*, или *защищенный протокол IP*. Стандарт обеспечения защиты системы, который позволяет отправителю выбрать метод аутентификации и шифрования содержимого дейтаграммы. Протокол IPsec можно использовать совместно как с протоколом IPv4, так и с IPv6.

## **IPv4**

*Internet Protocol version 4*, или *протокол Internet, версия 4*. Официальное название текущей версии протокола IP.

## **IPv6**

*Internet Protocol version 6*, или *протокол Internet, версия 6*. Название следующей версии протокола IP. См. также *IPng*.

## **IP-адрес**

32-битовое двоичное число, назначаемое каждому узлу при подключении к объединенной сети на основе семейства протоколов TCP/IP. IP-адреса являются абстрактным понятием физических адресов, точно так же, как объединенная сеть является абстракцией физических сетей. Чтобы повысить эффективность

процесса маршрутизации, каждый IP-адрес разделяется на две части, определяющие сеть и узел в этой сети.

#### **IP-в-IP**

Инкапсуляция одной IP-дейтаграммы в другую IP-дейтаграмму для последующей передачи по туннелю. Инкапсуляция IP-в-IP часто используется для передачи многоадресатных дейтаграмм через Internet.

#### **IP-дейтаграмма**

Основная единица передачи информации в объединенной сети на основе протокола TCP/IP. IP-дейтаграмма в объединенной сети является полным аналогом фрейма, передающегося в физической сети. Каждая дейтаграмма, помимо прочего, содержит адреса отправителя и конечного получателя, а также данные.

#### **IP-коммутация (IP switching)**

Первоначально, высокоскоростная технология перенаправления IP-дейтаграмм, разработанная фирмой Ipsilon Corporation. В настоящее время этот термин обычно используется для обозначения любой аналогичной технологии.

#### **IP-маршрутизатор (IP router)**

Одно из устройств, позволяющее соединить между собой две или более (возможно разнотипных) сетевых системы и выполняющее передачу пакетов между ними. Как следует из названия, маршрутизатор выполняет поиск в своих таблицах элемента, соответствующего адресу конечного получателя, извлекает из него адрес ближайшей точки перехода и отправляет по нему дейтаграмму.

#### **IP-телефония (IP telephony)**

Телефонная система, в которой для передачи оцифрованных голосовых данных используется протокол IP.

#### **IP-шлюз (IP gateway)**

Синоним *IP-маршрутизатора*.

#### **IRSG**

*Internet Research Steering Group*, или *руководящая исследовательская группа Internet*. Группа лиц, руководящих IRTF.

#### **IRTF**

*Internet Research Task Force*, или *исследовательская группа по решению конкретной задачи Internet*. Группа лиц, занимающихся исследованиями, связанными с семейством протоколов TCP/IP и подключением к Internet. Активность этой группы не столь велика, как IETF.

#### **ISDN**

*Integrated Services Digital Network*, или *цифровая сеть с интеграцией служб*. Название услуги по передаче цифровой данных, предоставляемой телефонными компаниями.

#### **ISO**

*International Organization for Standardization*, или *международная организация по стандартизации*. Международная организация, которая разрабатывает и принимает стандарты для сетевых протоколов. Организацией ISO была принята знаменитая семиуровневая эталонная модель, с помощью которой описывается организационная структура сетевых протоколов. Несмотря на то что организаци-

ей ISO было также предложено семейство протоколов для взаимодействия открытых систем (Open System Interconnection, или OSI), оно так и не было принято для широкого коммерческого использования.

#### **ISOC**

См. *сообщество Internet*.

#### **ISODE**

*ISO Development Environment*, или *среда разработки, отвечающая требованиям ISO*. Программное обеспечение, в котором реализован интерфейс протокола транспортного уровня ISO на основе протокола TCP/IP. Эта среда была создана для того, чтобы разработчики могли проводить эксперименты с высокоуровневыми протоколами модели OSI не создавая объединенную сеть, в которой бы использовались низкоуровневые протоколы семейства OSI.

#### **ISP**

*Internet Service Provider*, или *провайдер услуг Internet*. Организация, которая продает доступ к сети Internet, обеспечивая своим клиентам либо постоянное подключение к сети, либо доступ по коммутируемым линиям связи с помощью модема.

#### **ITU**

См. *международный телекоммуникационный союз*.

#### **Kbps**

*Kilo Bits Per Second*, или *килобит в секунду*. Единица измерения скорости передачи данных в канале связи, равная  $2^{10}$  бит/с. См. также *Гбит/с*, *Мбит/с* и *бод*.

#### **LAN**

*Local Area Network*, или *локальная сеть*. Технология построения физических сетей, охватывающих небольшие расстояния (до нескольких тысяч метров). Обычно скорость передачи данных в локальных сетях колеблется от 10 Мбит/с до нескольких Гбит/с. В качестве примеров локальных сетей можно привести сети типа Ethernet и FDDI. См. также *MAN* и *WAN*.

#### **LIS**

*Logical IP Subnet*, или *логическая IP-подсеть*. Группа компьютеров, соединенных между собой посредством сети ATM, которая используется в качестве одной (зачастую локальной) физической сети. Компьютер, подключенный к одной логической подсети не может напрямую посыпать дейтаграммы компьютеру, подключенному к другой логической подсети.

#### **LLC**

*Logical Link Control*, или *управление логическим соединением*. Одно из полей в заголовке NSAP.

#### **LSR**

*Loose Source Route*, или *нестрогая маршрутизация от источника*. Один из параметров протокола IP, в котором перечислен список адресов маршрутизаторов, через которые в указанном порядке должна обязательно пройти дейтаграмма. В отличие от строгой маршрутизации от источника, при нестрогой маршрутизации дейтаграмма может проходить через дополнительные маршрутизаторы, адреса которых не указаны в списке. См. также *SSR*.

## **MABR**

*Multicast Area Border Router*, или *многоадресатный пограничный маршрутизатор зоны*. Термин, используемый в многоадресатных расширениях протокола OSPF (MOSPF), для обозначения многоадресатного маршрутизатора, который обменивается маршрутной информацией с многоадресатным маршрутизатором другой зоны.

## **MAC**

*Media Access Control*, или *управление доступом к среде передачи данных*. Обобщенное название набора низкоуровневых аппаратных протоколов, используемых для доступа к определенной сети. Часто термин МАС-адрес используют как синоним термина *физический адрес*.

## **MAN**

*Metropolitan Area Network*, или *региональная вычислительная сеть*. Физическая высокоскоростная сеть (обычно от 100 Мбит/с до нескольких Гбит/с), протянувшаяся на значительные расстояния с целью охвата всего региона. См. *LAN* и *WAN*.

## **MBONE**

*Multicast BackBONE*, или *магистраль многоадресатной передачи данных*. Партнерское соглашение между несколькими сетевыми центрами о пересылке многоадресатных дейтаграмм между ними по IP-туннелю, проложенному через Internet.

## **Mbps**

*Millions of Bits Per Second*, или *миллион битов в секунду*. Единица измерения скорости передачи данных в канале связи, равная  $2^{20}$  бит/с. См. также *Гбит/с*, *Кбит/с* и *бод*.

## **MIB**

*Management Information Base*, или *база управляющей информации*. Набор переменных, собранных в базе данных, которые должны храниться в управляемом по протоколу SNMP устройстве, чтобы к ним мог обратиться агент протокола SNMP. С помощью выборки и присваивания значений переменным администраторы могут управлять устройством.

## **MILNET**

*Military Network*, или *военная сеть*. Первоначально она входила в состав ARPANET, а с 1984 года была выделена в самостоятельную сеть.

## **MIME**

*Multipurpose Internet Mail Extensions*, или *многоцелевые расширения электронной почты в сети Internet*. Стандарт, используемый для кодирования бинарных данных, таких как изображения, в текстовый формат для передачи в виде сообщения электронной почты.

## **Mosaic**

Название одного из первых графических браузеров Web.

## **MOSPF**

*Multicast Open Shortest Path First*. Многоадресатные расширения протокола маршрутизации OSPF.

## **MPLS**

*Multi-Protocol Label Switching*, или *многопротокольная коммутация меток*. Технология, которая используется в высокоскоростных коммутирующих устройствах для пересылки IP-дейтаграмм. В основу работы протокола MPLS положены технологии IP-коммутации и коммутации меток.

## **mroute**

*Multicast ROUTE Daemon*, или *демон многоадресатной маршрутизации*. Программа, используемая совместно с семейством протоколов, поддерживающих многоадресатный режим передачи протокола IP, для маршрутизации многоадресатных дейтаграмм.

## **MSL**

*Maximum Segment Lifetime*, или *максимальное время жизни сегмента*. Максимальное значение интервала времени, в течение которого дейтаграмма может находиться в пределах объединенной сети. Значение MSL используется в семействе протоколов TCP/IP для обозначения интервала времени, в течение которого могут появляться дубликаты пакетов.

## **MSS**

*Maximum Segment Size*, или *максимальный размер сегмента*. Термин, используемый в протоколе TCP. Он означает размер максимального блока данных, который может быть передан в одном сегменте. Во время установки соединения, отправитель и получатель согласовывают между собой максимальный размер сегмента.

## **MTU**

*Maximum Transfer Unit*, *Maximum Transmission Unit*, или *максимальная единица передачи данных в сети*. Размер максимального блока данных, который может быть передан по физической сети. Величина MTU зависит от используемого сетевого оборудования.

## **NACK**

*Negative Acknowledgement*, или *отрицательное уведомление*. Сообщение, посылаемое получателем данных их отправителю в случае возникновения каких-либо проблем, например при потере дейтаграммы или ее повреждении. Как правило, в ответ на получение сигнала NACK, отправитель выполняет повторную передачу данных.

## **NAK**

Синоним термина *NACK*.

## **NAP**

*Network Access Point*, или *точка доступа к сети*. Граница раздела между несколькими автономными системами, физическое место в сети, к которому крупные провайдеры подключают свои сети. В точке доступа располагается сервер маршрутизации, на котором хранится копия арбитражной базы данных и поддерживается протокол BGP для обмена информацией о достижимости сетей между провайдерами. Кроме использования точек доступа, многие провайдеры заключают между собой дополнительные партнерские соглашения по обмену трафиком.

## **NAT**

*Network Address Translation*, или *преобразование сетевых адресов*. Технология, позволяющая узлам сети, которым назначен локальный IP-адрес, обмени-

ваться информацией с машинами, находящимися во внешних сетях, например в глобальной сети Internet.

#### **NBMA**

*Non-Broadcast Multi-Access*, или *сети с множественным доступом, не поддерживающие режим широковещания*. Одна из характеристик сети передачи данных, к которой можно подключить большое количество компьютеров, но в которой на аппаратном уровне не поддерживается режим широковещания. Одним из примеров сетей NBMA является ATM.

#### **NetBIOS**

*Network Basic Input Output System*, или *сетевая базовая система ввода/вывода*. Стандарт сетевого программного интерфейса, используемый в сетях на основе компьютеров IBM PC и совместимых с ними персональных компьютеров. В стандарт семейства протоколов TCP/IP включена инструкция, в которой оговорен процесс преобразования команд протокола NetBIOS в эквивалентные команды протокола TCP/IP.

#### **NFS**

*Network File System*, или *сетевая файловая система*. Протокол, созданный фирмой SUN Microsystems, Inc., на основе протокола IP, который позволяет группе взаимодействующих между собой компьютеров получать доступ к файлам друг друга так, как если бы они находились на локальном диске.

#### **NIC**

*Network Interface Card*, или *плата сетевого интерфейса*. Электронное устройство, посредством которого компьютер подключается к сети. Обычно плата сетевого интерфейса вставляется в один из свободных слотов расширения компьютера.

#### **NIST**

*National Institute of Standards and Technology*, или *Национальный институт стандартов и технологий*. Бывшее Национальное бюро стандартов (National Bureau of Standards). NIST является одной из организаций в США, которая занимается разработкой стандартов сетевых протоколов.

#### **NLA**

*Next Level Aggregation*, или *группировка следующего уровня*. Третье справа поле в избирательном адресе протокола IPv6. См. также *TLA*.

#### **NOC**

*Network Operations Center*, или *сетевой операционный центр*. См. *INOC*.

#### **NSAP**

*Network Service Access Point*, или *точка доступа к сетевой службе*. Формат закодированного физического адреса, длиной 20 октетов. Форум ATM рекомендует использовать адреса NSAP.

#### **NSF**

*National Science Foundation*, или *Национальный научный фонд США*. Государственная организация в США, которая финансировала ряд проектов по разработке и внедрению технологий Internet.

## **NSFNET**

*National Science Foundation NETwork*, или сеть национального научного фонда США. Так называется магистральная сеть Internet в США, которая была создана на средства NSF.

## **NVT**

*Network Virtual Terminal*, сетевой виртуальный терминал. Символично ориентированный протокол, используемый в протоколе TELNET.

## **OSI**

*Open Systems Interconnection*, или взаимодействие открытых систем. Название разработанного ISO семейства протоколов, которое в свое время конкурировало с семейством протоколов TCP/IP. Однако оно так и не прижилось, и поэтому сейчас практически не поддерживается в коммерческих системах.

## **OSPF**

*Open Shortest Path First*, или открытый протокол поиска кратчайших маршрутов. Протокол маршрутизации, отслеживающий состояние соединения, разработанный IETF.

## **OUI**

*Organizationally Unique Identifier*, или уникальный идентификатор организации. Число, являющееся частью физического адреса, назначаемого организации, выпускающей сетевое оборудование. Производитель назначает каждому выпущенному им устройству уникальный физический адрес, который состоит из префикса OUI и суффикса, содержащего номер устройства.

## **PCM**

*Pulse Code Modulation*, или импульсно-кодовая модуляция. Стандарт кодирования голосовых данных, используемый в цифровой телефонии, согласно которому аналоговый сигнал оцифровывается с частотой 8000 раз в секунду при ширине выборки 8 бит.

## **PDN**

*Public Data Network*, или сеть общего пользования. Один из механизмов передачи цифровых данных, предлагаемый операторами связи.

## **PDU**

*Packet Data Unit*, или модуль данных протокола. Термин, принятый ISO и используемый для обозначения пакета или сообщения.

## **PEM**

*Privacy Enhanced Mail*, или почта повышенной секретности. Протокол шифрования сообщений электронной почты для передачи их по открытой сети, такой как Internet.

## **PIM-DM**

*Protocol Independent Multicast Dense Mode*, или уплотненный режим не зависящий от протокола многоадресатной передачи. Протокол многоадресатной маршрутизации, управляемый данными, аналогичный протоколу DVMRP.

## **PIM-SM**

*Protocol Independent Multicast Sparse Mode*, или разреженный режим не зависящий от протокола многоадресатной передачи. Протокол многоадресатной

маршрутизации, управляемый по запросу, в котором расширена идея использования наращиваемых от центра деревьев (CBT).

#### **PING**

*Packet InterNet Groper*, или *межсетевой пакетный тестер*. Название программы, используемой в объединенных сетях на основе протокола TCP/IP для оценки достижимости получателя. Для этого программа посыпает по указанному в командной строке адресу эхо-запрос протокола ICMP и ожидает поступления на него ответа.

#### **POP**

*Post Office Protocol*, или *почтовый протокол*. Протокол, который используется для выборки сообщений из почтового ящика, расположенного на сервере.

#### **POTS**

*Plain Old Telephone Service*, или *обычная телефонная сеть*. Название стандартной системы передачи голоса посредством аналоговой телефонной системы.

#### **PPP**

*Point to Point Protocol*, или *протокол двухточечного соединения*. Протокол передачи IP-фреймов по последовательному каналу связи. См. также *SLIP*.

#### **PSN**

*Packet Switching Node*, или *узел коммутации пакетов*. Официальное название коммутатора пакетов сети ARPANET, который раньше назывался интерфейсным процессором сообщений (*IMP*).

#### **PSTN**

*Public Switched Telephone Network*, или *коммутируемая телефонная сеть общего пользования*. Типовая система передачи голосовых данных посредством телефонной связи.

#### **PUP**

*Parc Universal Packet*, или *универсальный пакет PARC*. В системе межсетевого взаимодействия, созданной фирмой Xerox Corporation, PUP является основной единицей передачи данных, также как IP-пакет в сетях на основе протокола TCP/IP. Пакет был назван в честь лаборатории фирмы Xerox, разработавшей технологию межсетевого взаимодействия — исследовательский центр Пало Альто (Palo Alto Research Center, PARC).

#### **PVC**

*Permanent Virtual Circuit*, или *постоянный виртуальный канал*. Один из двух типов виртуальных каналов, конфигурируемых вручную сетевым администратором, а не автоматически компьютером по запросу программы. В отличие от *SVC* постоянный виртуальный канал предназначен для длительного использования (недели или месяцы).

#### **QoS**

*Quality of Service*, или *качество обслуживания*. Гарантированные значения величины потери пакетов, задержки, дрожания и минимальной скорости передачи данных, обеспечиваемые системой доставки. Ряд приверженцев QoS считают, что без него нельзя обойтись при передаче данных в реальном масштабе времени.

## **RA**

См. *арбитражная система маршрутизации*.

## **RARP**

*Reverse Address Resolution Protocol*, или *протокол обратного преобразования адресов*. Один из протоколов, который может использовать узел сети в процессе начальной загрузки для определения своего IP-адреса. Несмотря на то что протокол RARP был некогда очень популярным, в настоящее время на большинстве узлов сети используются протоколы *BOOTP* или *DHCP*.

## **RDP**

*Reliable Datagram Protocol*, или *протокол надежной передачи дейтаграмм*. Протокол, используемый для реализации надежной службы доставки дейтаграмм на основе одной из служб протокола IP, не гарантирующих доставку пакетов. Протокол RDP не относится к разряду популярных протоколов, поддерживаемых во всех реализациях семейства протоколов TCP/IP.

## **RED**

*Random Early Discard*, или *произвольное раннее обнаружение*. Специальный алгоритм, который используется в маршрутизаторе для управления очередями пакетов в случае возникновения перегрузки. Такой алгоритм более эффективен, чем простое усечение хвоста очереди. Суть его состоит в том, что при достижении предельного размера очереди маршрутизатор начинает случайным образом удалять из нее дейтаграммы.

## **RFC**

*Request For Comments*, или *запрос на комментарии*. Название серии документов, посвященных семейству протоколов TCP/IP (и не только). В RFC содержатся обзоры, данные измерений, описание идей и методик, предлагаемых и принятых стандартов, а также данные результатов экспериментов. Документы RFC доступны для свободного использования, их можно загрузить из Internet.

## **RIP**

*Routing Information Protocol*, или *протокол маршрутной информации*. Один из протоколов, используемых для распространения маршрутной информации внутри автономной системы. Протокол RIP был создан на основе одного из первых протоколов с тем же названием, разработанным фирмой Xerox.

## **RJE**

*Remote Job Entry*, или *дистанционный ввод заданий*. Служба, которая позволяет принять пакетное задание на обработку от удаленного узла сети.

## **rlogin**

*Remote LOGIN*, или *удаленный вход в систему*. Протокол удаленной регистрации пользователей, используемый в системе UNIX, выпущенной Калифорнийским университетом в Беркли. Служба rlogin имеет практически те же возможности, что и TELNET.

## **ROADS**

*Running Out of Address Space*, или *исчерпание адресного пространства*. Название проблемы исчерпания адресного пространства протокола IPv4, которая может случиться в ближайшем будущем.

**routed**

*Route Daemon*, или *демон маршрутизации*. Программа, созданная для системы UNIX, в которой реализован протокол *RIP*.

**RP**

*Rendezvous Point*, или *точка сбора*. Маршрутизатор, которому отсылаются запросы на присоединение к многоадресатной группе в случае использования протокола многоадресатной маршрутизации, управляемого по запросу.

**RPB**

*Reverse Path Broadcast*, или *широковещание по обратному маршруту*. Синоним термина *RPF*.

**RPC**

*Remote Procedure Call*, или *дистанционный вызов процедур*. Технология вызова по сети процедур прикладными программами, работающими на другой машине. Когда клиентская программа вызывает одну из удаленных процедур, механизм RPC автоматически считывает значения переданных ей параметров, формирует сообщение и посылает его на удаленный сервер, ожидает от него ответа, и сохраняет возвращенные значения в соответствующих параметрах. В системе NFS используется RPC особого типа.

**RPF**

*Reverse Path Forwarding*, или *пересылка по обратному маршруту*. Методика, используемая для распространения широковещательных пакетов, которая гарантирует отсутствие маршрутных петель. В протоколе IP пересылка по обратному маршруту используется для распространения к указанной подсети широковещательных пакетов и многоадресатных дейтаграмм.

**RPM**

*Reverse Path Multicast*, или *многоадресатная передача по обратному маршруту*. Обобщенный подход к многоадресатной маршрутизации с использованием алгоритма TRPB.

**RS**

См. *сервер маршрутизации*.

**RS232**

Стандарт, принятый EIA, в котором описаны электрические параметры линии последовательной передачи данных, соединяющей терминал с компьютером, или два компьютера между собой. Хотя для последовательной передачи данных чаще всего используется другой стандарт — RS232C, большинство пользователей продолжает называть его RS232.

**RST**

*Reset*, или *сброс*. Общеиспользуемая аббревиатура, обозначающая TCP-сегмент, в котором передается команда на сброс соединения.

**RSVP**

*Resource Reservation Protocol*, или *протокол резервирования ресурсов*. Протокол, который позволяет конечной точке соединения запрашивать поток данных с заданным качеством обслуживания. При этом все маршрутизаторы, находящиеся по пути следования пакета, должны или отклонить этот запрос или удовлетворить его.

## **RTCP**

*RTP Control Protocol*, или *протокол управления в реальном масштабе времени*. Родственный протоколу RTP протокол, который используется для управления сеансом связи.

## **RTO**

*Round trip Time-Out*, или *тайм-аут времени кругового обращения*. Величина задержки перед повторной передачей пакета. В протоколе TCP значение RTO вычисляется как функция времени кругового обращения и величины его отклонения.

## **RTP**

*Real-time Transport Protocol*, или *протокол передачи данных в реальном масштабе времени*. Основной протокол, используемый для передачи голосовых и видеоданных в реальном масштабе времени по сети IP.

## **RTT**

*Round Trip Time*, или *время кругового обращения*. Характеристика задержки распространения пакета между двумя узлами сети. Общее время, необходимое для передачи пакета по сети от отправителя до конечного получателя и его возврата отправителю. В протоколе TCP оценка времени кругового обращения используется для вычисления текущего значения таймера повторной передачи. В большинстве сетей с коммутацией пакетов, значение задержки является случайной величиной и зависит от степени перегрузки сети. Поэтому при оценке времени кругового обращения вычисляется его среднее значение и величина стандартного отклонения, которая является довольно большой.

## **SA**

*Security Association*, или *ассоциация обеспечения безопасности*. Абстрактное понятие, используемое в протоколе IPsec для установки соответствия между набором параметров системы безопасности и числовым идентификатором (индексом), передаваемым в заголовке дейтаграммы. Набор параметров системы безопасности выбирается узлом сети, он не является всеобщим стандартом. См. также *SPI*.

## **SACK**

*Selective Acknowledgement*, или *избирательное подтверждение приема*. Механизм подтверждения приема, используемый в протоколе движущихся окон, который позволяет получателю посыпать уведомления о получении пакетов, находящихся в текущем окне и прибывших вне установленного порядка. Его называют также расширенным механизмом подтверждения приема. Ср. с *кумулятивной системой подтверждения приема*, используемой в протоколе TCP.

## **SAR**

*Segmentation And Reassembly*, или *сегментация и сборка*. Процесс разделения сообщения на последовательность ячеек для передачи их по сети ATM и последующей сборки и восстановления первоначального сообщения. При отправке дейтаграммы по сети ATM процесс сегментации выполняется протоколом адаптации ATM уровня 5.

## **SGMP**

*Simple Gateway Monitoring Protocol*, или *простой протокол мониторинга шлюза*. Предшественник протокола SNMP.

**SIP**

*Session Initiation Protocol*, или *протокол иницирования сеанса связи*. Протокол, разработанный IETF для поддержки системы сигнализации IP-телефонии. (Примечание. Аббревиатура SIP ранее расшифровывалась как Simple IP, т.е. протокол, который был взят за основу при разработке протокола IPv6.)

**SIPP**

*SIP Plus*. Расширение протокола Simple IP, который был предложен в качестве основы при разработке протокола IPv6. См. *IPv6*.

**SLIP**

*Serial Line IP*, или *межсетевой протокол для последовательного канала связи*. Протокол передачи IP-фреймов по последовательному каналу связи. Протокол SLIP был ранее популярен при подключении к объединенной сети через коммутируемые телефонные линии связи с помощью модема. В настоящее время заменен протоколом *PPP*.

**SMDS**

*Switched Multimegabit Data Service*, или *служба высокоскоростной коммутации данных*. Служба коммутации пакетов, не требующая установки соединения с получателем, предлагаемая региональными телефонными компаниями.

**SMI**

*Structure of Management Information*, или *структура управляющей информации*. Правила, используемые для определения и идентификации переменных базы MIB.

**SMTP**

*Simple Mail Transfer Protocol*, или *простой протокол передачи электронной почты*. Стандарт семейства протоколов TCP/IP, используемый для передачи сообщений электронной почты с одного компьютера на другой. В протоколе SMTP определяется порядок взаимодействия двух почтовых систем и формат управляющих сообщений, которыми они обмениваются при передаче почты.

**SNA**

*System Network Architecture*, или *системная сетевая структура*. Этот термин используется при описании структуры, форматов и протоколов, созданных фирмой IBM для передачи информации между программным и аппаратным обеспечением фирмы IBM Corporation. В SNA не поддерживается режим взаимодействия с семейством протоколов TCP/IP.

**SNAP**

*SubNetwork Attachment Point*, или *точка подключения к подсети*. Стандартный заголовок небольшого размера принятый IEEE, который добавляется к данным, посылаемым по сети, не поддерживающей автоматически опознаваемых фреймов. В заголовке SNAP указывается тип передаваемых данных.

**SNMP**

*Simple Network Management Protocol*, или *простой протокол сетевого управления*. Протокол, который используется для управления устройствами, такими как узлы сети, маршрутизаторы и принтеры. После аббревиатуры “SNMP” обычно указывается суффикс, обозначающий номер версии протокола, например SNMPv3. См. также *MIB*.

**SOA**

*Start Of Authority*, или *начало полномочия*. Ключевое слово, используемое в зональном файле DNS для обозначения начала группы записей, полномочия по управлению которыми переданы данному серверу. Информация из любых других записей, не входящих в группу SOA, возвращается данным сервером с пометкой *non-authoritative* (т.е. что информация получена не из достоверного источника).

**SPF**

*Shortest Path First*, или  *поиск первого кратчайшего пути*. Класс протоколов обновления маршрутной информации, в которых используется алгоритм Дейкстры для вычисления кратчайших путей к получателю. См. *маршрутизация на основе отслеживания состояния соединения*.

**SPI**

*Security Parameters Index*, или *индекс параметров безопасности*. Числовой идентификатор, используемый в протоколе IPsec для обозначения набора параметров системы безопасности, на основе которого должна обрабатываться дейтаграмма. См. также *SA*.

**SS7**

*Signaling System 7*, или *система общеканальной сигнализации №7*. Стандарт системы сигнализации, используемый в обычной телефонной системе.

**SSL**

*Secure Sockets Layer*, или *протокол защищенных сокетов*. Технология создания зашифрованного канала связи между получателями, разработанная фирмой Netscape, Inc., а затем ставшая фактическим стандартом. Стандарт SSL существовал в виде рабочего документа, но так никогда и не стал документом RFC.

**SSR**

*Strict Source Route*, или *строгая маршрутизация от источника*. Один из параметров протокола IP, содержащий список адресов маршрутизаторов, через которые в указанном порядке обязательно должна пройти дейтаграмма. См. также *LSR*.

**STD**

*Standard*, или *стандарт*. Обозначение, используемое в RFC, которое относится к группе документов, содержащих стандарты протоколов.

**SVC**

*Switched Virtual Circuit*, или *коммутируемый виртуальный канал*. Один из двух типов виртуальных каналов, устанавливаемых динамически и разрываемый, если в нем больше нет необходимости. Как правило запросы на создание SVC поступают от программного обеспечения компьютера. В отличие от PVC, время использования SVC может быть очень коротким.

**SWS**

*Silly window syndrome*, или *синдром полного окна*.

**SYN**

*Synchronizing segment*, или *シンхронизирующий сегмент*. Первый служебный сегмент, посыпаемый программами поддержки протокола TCP. Он используется для синхронизации сторон соединения при выполнении операции открытия потока данных.

### **T3**

Термин, используемый в телефонии, для обозначения протокола передачи данных по каналам DS3. Этот термин часто используют в качестве синонима DS3, что не совсем корректно.

### **TCP**

*Transmission Control Protocol*, или *протокол управления передачей*. Стандартный протокол транспортного уровня, входящий в семейство протоколов TCP/IP и обеспечивающий надежную дуплексную потоковую передачу данных. Протокол TCP используется во многих прикладных программах. Протокол TCP позволяет одному компьютеру посыпать поток данных на обработку другому компьютеру. При использовании этого протокола отправитель и получатель перед отправкой данных должны установить друг с другом соединение. В протоколе TCP данные передаются в виде сегментов, которые перед отправкой по объединенной сети помещаются в IP-дейтаграммы. Поскольку протоколы TCP и IP являются основными, в их “честь” было названо все семейство протоколов TCP/IP.

### **TDM**

*Time Division Multiplexing*, или *мультплексная передача с временным разделением каналов*. Метод передачи нескольких независимых потоков данных в одной среде (например, кабеле или эфире), который предусматривает передачу каждого потока данных по каналу связи в течение фиксированного короткого промежутка времени, называемого *временным окном*. См. также *FDM*.

### **TDMA**

*Time Division Multiple Access*, или *множественный доступ с временным разделением*. Метод доступа к сетевой среде, при использовании которого каждому узлу сети отводится для передачи данных короткий временной интервал. Поскольку при таком доступе к среде передачи данных работа всех узлов сети должна быть четко синхронизирована, а также должна быть учтена задержка распространения сигнала, вносимая сетью, сеть на основе технологии TDMA довольно трудно реализовать. Кроме того, стоимость оборудования для такой реализации очень высока.

### **TELNET**

Один из стандартов, входящих в семейство протоколов TCP/IP, предназначенный для поддержки сеанса связи с удаленным терминалом. Протокол TELNET позволяет пользователю, находящемуся на одном конце соединения, взаимодействовать в режиме разделения времени с компьютером, находящемся на другом конце соединения так, как будто бы клавиатура и монитор пользователя напрямую подключены к удаленному компьютеру.

### **TFTP**

*Trivial File Transfer Protocol*, или *простейший протокол передачи файлов*. Один из стандартов, входящих в семейство протоколов TCP/IP, предназначенный для реализации режима простейшей передачи файлов. Поэтому в данном протоколе поддерживается только минимально необходимый набор средств, что делает его использование очень эффективным. Поскольку в протоколе TFTP используется недостаточно надежная, не требующая соединения служба доставки дейтаграмм (протокол UDP), он может использоваться только в пределах одной локальной сети.

### **TLA**

*Top Level Aggregation*, или *группировка верхнего уровня*. Второе слева поле в избирательном адресе протокола IPv6. См. также *NLA*.

**TLI**

*Transport Layer Interface*, или *интерфейс транспортного уровня*. Стандарт сетевого взаимодействия, разработанный для системы UNIX System V, который является альтернативой интерфейсу сокетов.

**TLV-кодирование (TLV encoding)**

*Type-Length-Value*. Один из форматов представления закодированных данных в виде трех полей: типа, длины и значения. В протоколе IP TLV-кодирование используется при определении параметров протокола.

**tn3270**

Версия протокола TELNET, которая используется совместно с терминалами 3270 фирмы IBM.

**TOS**

*Type Of Service*, или *тип сервиса*. Оригинальное название одного из полей заголовка протокола IPv4, с помощью которого отправитель мог указывать требуемый тип обслуживания. В настоящее время это поле заменено полем дифференцированного обслуживания (DiffServe).

**TP-4**

Протокол, созданный ISO, который является аналогом протокола TCP.

**traceroute**

Программа, которая выводит трассировку маршрута до конечного получателя. Программа traceroute отправляет получателю последовательность дейтаграмм, в каждой из которых значение времени жизни увеличивается на единицу, начиная с 1 (1, 2, 3 и т.д.). Это приводит к тому, что каждый из маршрутизаторов, расположенных по пути следования дейтаграммы, вернет отправителю ICMP-сообщение о том, что время жизни дейтаграммы истекло. По ним программа traceroute и узнает маршрут до конечного получателя.

**TRPB**

*Truncated Reverse Path Broadcast*, или *усеченное широковещание по обратному маршруту*. Методика, используемая в многоадресатных протоколах маршрутизации, управляемых данными для пересылки многоадресатных дейтаграмм. См. *широковещание и отсечение*.

**TRPF**

*Truncated Reverse Path Forwarding*, или *усеченное широковещание по обратному маршруту*. Синоним термина TRPB.

**TTL**

*Time To Live*, или *время жизни*. Методика, применяемая в системах негарантированной доставки и позволяющая избежать бесконечных циклов передачи пакетов. Например, каждой IP-дейтаграмме при создании назначается целое число, обозначающее ее время жизни в сети. При прохождении дейтаграммы через маршрутизатор он уменьшает на единицу значение ее времени жизни. Когда значение времени жизни становится равным нулю, маршрутизатор аннулирует дейтаграмму.

**UART**

*Universal Asynchronous Receiver and Transmitter*, или *универсальный асинхронный приемопередатчик*. Электронное устройство, состоящее из одной интегральной

микросхемы, которая может посыпать и принимать символы по асинхронным последовательным каналам связи, таким как RS232. UART является программируемым устройством, поскольку в нем предусмотрены специальные управляющие сигналы, позволяющие разработчику задать параметры работы устройства, такие как скорость передачи данных, четность, количество стоповых битов и сигналы управления модемом. UART применяется в терминалах, модемах, портах ввода/вывода, а также в различных периферийных устройствах компьютера.

#### **UCBCAST**

См. широковещательный адрес Беркли.

#### **UDP**

*User Datagram Protocol*, или протокол передачи пользовательских дейтаграмм. Один из протоколов транспортного уровня, позволяющий прикладным программам, запущенным на разных компьютерах, обмениваться между собой дейтаграммами. Для доставки дейтаграмм в протоколе UDP используется протокол IP. Принципиальная разница между дейтаграммами протоколов UDP и IP заключается в том, что в UDP-дейтаграммах указывается номер порта протокола. Это позволяет отправителю указывать, какой из прикладных программ, запущенных на удаленном компьютере, предназначена та или иная дейтаграмма.

#### **URI**

*Uniform Resource Identifier*, или унифицированный идентификатор ресурса. Общий термин, используемый для обозначения URN или URL.

#### **URL**

*Uniform Resource Locator*, или унифицированный указатель информационного ресурса. Текстовая строка, с помощью которой описывается размещение источника информации. В начале строки указывается тип используемого протокола (например, `http://`), за которым следует идентификатор указанного источника (т.е. доменное имя сервера и полный путь к файлу, расположенному на данном сервере).

#### **URN**

*Uniform Resource Name*, или унифицированное имя ресурса. Текстовая строка, с помощью которой описывается размещение источника информации. В отличие от URL, URN гарантирует, что источник информации будет существовать на протяжении длительного времени.

#### **UUCP**

*Unix to Unix Copy Program*, или протокол, используемый для обмена файлами между согласованными UNIX-системами. Служебная программа, разработанная в середине 1970-х годов для системы UNIX версии 7. Она позволяла копировать файлы с одной UNIX-машины на другую в режиме разделения времени по проложенному между ними единственному каналу связи (как правило коммутируемому). Поскольку программа UUCP обычно использовалась для обмена электронной почтой между UNIX-машинами, этот термин часто некорректно используют для названия устаревшего типа почтовой системы.

#### **vBNS**

*Very High Speed Backbone Network Service*, или сверхвысокоскоростной магистральный канал сетевых служб. Магистральная сеть со скоростью передачи данных 155 Мбит/с, запущенная в эксплуатацию в 1995 году и используемая в настоящее время для проведения исследований в области сетевых технологий.

## **VC**

*Virtual Circuit*, или *виртуальный канал*. Маршрут, проложенный по сети между двумя прикладными программами, запущенными на разных компьютерах, который используется для пересылки данных между ними. Виртуальный канал создается либо по запросу программы поддержки протокола, либо вручную. При его использовании создается иллюзия, что непосредственно между двумя компьютерами проложен кабель. Идея использования виртуальных каналов в несколько расширенном виде нашла воплощение в сети ATM.

## **VLSM**

*Variable Length Subnet Mask*, или *маска подсети переменной длины*. Маска подсети, которая используется при адресации подсетей переменной длины.

## **VPI/VCI**

*Virtual Path Identifier/Virtual Circuit Identifier*, или *идентификатор виртуального маршрута/идентификатор виртуального канала*. Пара идентификаторов, которые используются в сетях ATM для обозначения виртуального соединения. При открытии соединения узлом сети ему назначается уникальный идентификатор, состоящий из пары VPI/VCI.

## **VPN**

*Virtual Private Network*, или *виртуальная частная сеть*. Технология, позволяющая соединить между собой два или несколько сетевых центров через Internet так, чтобы для пользователей этих центров создавалась иллюзия работы в одной частной локальной сети. Поскольку пакеты между сетевыми центрами передаются по открытой сети Internet, для обеспечения требуемого уровня секретности в программах поддержки VPN используются алгоритмы шифрования.

## **WAN**

*Wide Area Network*, или *глобальная сеть*. Технология построения физических сетей, охватывающих большие географические территории. Эти сети часто называют протяженными (long-haul), или распределенными. В глобальной сети стоимость пересылки единицы информации и задержка распространения пакетов существенно выше, чем в сетях, охватывающих небольшие территории. См. также *LAN* и *MAN*.

## **World Wide Web**

Самая крупная гипермейдийная служба в Internet, которая позволяет пользователям получать информацию с помощью специальной программы просмотра, называемой *Web-браузером*.

## **WWW**

См. *World Wide Web*.

## **X**

См. *X-Window System*.

## **X.25**

Устаревший стандарт протокола, принятый ITU, который был популярен в Европе до того, как стало широко использоваться семейство протоколов TCP/IP

## **X.400**

Протокол, разработанный ITU, для обмена сообщениями по электронной почте.

## X25NET

*X.25 NETwork*, или *сеть X.25*. Один из видов услуг, предлагаемый CSNET для передачи IP-трафика по протоколу X.25 между сетевым центром заказчика и Internet.

## XDR

*External Data Representation*, или *представление внешних данных*. Универсальный, не зависящий от аппаратной платформы стандарт представления данных. При обмене данными, отправитель преобразует их из формата, принятого на локальном компьютере в формат XDR, а получатель выполняет обратное преобразование.

## X-Window System

Система программного обеспечения, разработанная в Массачусетском технологическом институте, для отображения информации в системе UNIX на графических мониторах. Каждое окно занимает определенную прямоугольную область пространства монитора, которая закрепляется в качестве устройства вывода текстовой и графической информации за каждой прикладной программой. Причем прикладные программы могут работать как на локальном, так и на удаленном компьютере. Кроме того, существует также специальная служебная программа, называемая диспетчером окон, с помощью которой пользователь может создавать, удалять и перемещать окна, а также изменять их размер и расположение на экране.

## Автоматически опознаваемый фрейм (self-identifying frame)

Одна из характеристик фрейма или пакета, в который включено специальное поле, идентифицирующее тип передаваемых в нем данных. Например, в сети Ethernet используются автоматически опознаваемые фреймы, а в сети ATM — нет.

## Агент (agent)

В сетевом управлении агент — это серверная программа, которая запускается на подконтрольном узле сети или маршрутизаторе и отвечает на запросы клиента управления.

## Агент ARP (proxy ARP)

Методика, заключающаяся в том, что одна из машин (обычно маршрутизатор) отвечает на ARP-запросы от имени другой машины, подставляя в них свой физический адрес. При этом все пакеты, посылаемые конкретной машине, сначала поступают на маршрутизатор, который затем пересыпает их конечному получателю. Агенты ARP используются для того, чтобы можно было назначить нескольким физическим сетям один адрес IP-сети.

## Адаптивный алгоритм повторной передачи (adaptive retransmission)

Механизм протокола TCP, который используется для нахождения текущего значения величины задержки повторной передачи, которая зависит от скорости работы соединения и среднего значения полного времени доставки пакета.

## Административная граница (administrative scoping)

Механизм, использующийся для ограничения области рассылки многоадресатных дейтаграмм. Для его реализации резервируется часть адресного пространства, которая затем может использоваться в пределах сетевого центра либо в пределах организации.

## Администратор зоны (area manager)

Человек, ответственный за зону IETF. Совокупность администраторов зоны образуют IESG.

### **Адрес Internet (Internet address)**

См. *IP-address*.

### **Адрес (address)**

Целочисленное значение, которое используется для идентификации компьютера в сети. Значение адреса указывается в заголовке каждого пакета, посылаемого по сети определенному компьютеру.

### **Адрес для передачи (care-of address)**

Временный IP-адрес, используемый мобильным узлом при подключении к внешней сети.

### **Адрес петли обратной связи (loopback address)**

Сетевой адрес, используемый для тестирования работоспособности узла сети. Пакеты, посылаемые по этому адресу, обрабатываются локальным модулем протокола и никогда не покидают пределов локальной машины (т.е. не посылаются в сеть). В протоколе IP в качестве префикса адреса петли обратной связи используется 127.0.0.0.

### **Адрес, локальный в пределах канала связи (link-local address)**

Адрес, используемый в протоколе IPv6, для рассылки дейтаграмм в пределах локальной сети.

### **Адрес, локальный в пределах сетевого центра (site-local address)**

Адрес, используемый в протоколе IPv6, для рассылки дейтаграмм в пределах одного сетевого центра или одной организации.

### **Адреса из сети 10**

Общее название группы немаршрутизуемых IP-адресов (т.е. таких адресов, которые зарезервированы для использования во внутренних сетях предприятий и не используются в глобальной сети Internet). Префикс сети 10.0.0.0 раньше был назначен сети ARPANET. После того, как сеть ARPANET перестала функционировать, этот блок адресов был переведен в категорию немаршрутизуемых.

### **Адресация подсетей (subnet addressing)**

Расширение системы IP-адресации, позволяющее одному сетевому центру использовать для всех своих физических сетей одинаковый сетевой префикс IP-адреса. С точки зрения внешних узлов сети, использование такой системы адресации не влияет на процесс маршрутизации. Для них, как и раньше, сетевой адрес разбивается на две части: префикс сети и суффикс узла. Однако маршрутизаторы и узлы сети, находящиеся внутри сетевого центра, должны интерпретировать суффикс узла несколько иначе. Из него выделяется часть, идентифицирующая физическую сеть, и часть, идентифицирующая узел в этой сети.

### **Адресация суперсетей (supernet addressing)**

Синоним термина *CIDR*.

### **Адресная привязка (address binding)**

Информация, используемая при *преобразовании* сетевого адреса высокого уровня в эквивалентный ему низкоуровневый физический адрес, например при преобразовании IP-адреса компьютера в физический адрес его сетевой платы Ethernet.

### **Алгоритм Беллмана–Форда (Bellman-Ford algorithm)**

Алгоритм, используемый при *дистанционно-векторной* маршрутизации.

### **Алгоритм Карна (Karn's Algorithm)**

Алгоритм, позволяющий протоколу транспортного уровня отличать корректные значения замеров полного времени доставки пакета от некорректных, т.е. точно определять ожидаемое время полной доставки пакета.

### **Алгоритм Форда–Фалкерсона (Ford-Fulkerson algorithm)**

Синоним дистанционно-векторного алгоритма, названного в честь двух учёных, которые его придумали.

### **Альтернативный адрес (anycast address)**

Форма адресации, появившаяся в протоколе IPv6. Дейтаграмма, посланная по альтернативному адресу, направляется одному из компьютеров выбранной группы. В качестве критерия выбора этого компьютера используется минимальное маршрутное расстояние до отправителя. *Альтернативный адрес* часто называют *клUSTERным адресом*.

### **Анонимная сеть (anonymous network)**

Синоним *ненумерованной сети*.

### **Анонимный сеанс FTP (anonymous: FTP)**

Сеанс работы по протоколу FTP, в котором для доступа к открытым файлам используется имя пользователя *anonymous* или *ftp*. В качестве пароля для регистрации на анонимном FTP-сервере часто используется слово *guest*.

### **Анонс окна (window advertisement)**

Сообщение протокола TCP, с помощью которого отправитель уведомляется о количестве доступного пространства в приемном буфере получателя.

### **Аппаратный адрес (hardware address)**

Низкоуровневый адрес, который используется для адресации устройств сети на физическом уровне. Является синонимом терминов *физический адрес* и *MAC-адрес*. Схема адресации, используемая на физическом уровне, зависит от типа сетевого оборудования (например, в сети Ethernet используется 48-битовый аппаратный адрес).

### **Арбитражная система маршрутизации (routing arbiter system)**

Основу этой системы составляет реплицируемая база данных, содержащая достоверную информацию о достижимости всех получателей в Internet. При этом перед обновлением информации в базе данных проверяется ее подлинность. База данных располагается в точке доступа к сети на специальном сервере маршрутизации, который поддерживает протокол BGP для обмена информацией о достижимости сетей между провайдерами.

### **База управляющей информации (Management Information Base)**

См. *MIB*.

### **Бастионный узел (bastion host)**

Защищенный компьютер, являющийся частью брандмауэра, на котором запускается программное обеспечение, взаимодействующее с компьютерами, расположеннымными за пределами локальной сети организации.

### **Бесклассовая адресация (classless addressing)**

Расширение первоначальной системы адресации протокола IPv4, устраняющее ранее установленные границы классов адресов. Принятие бесклассовой адресации было вызвано проблемой нехватки адресного пространства протокола IPv4.

### **Беспорядочный режим (promiscuous mode)**

Возможность сетевого интерфейса, позволяющая компьютеру получать все пакеты, передающиеся по сети.

### **Бод (baud)**

Формально, количество изменений параметров электрического сигнала в канале передачи данных за одну секунду. Обычно при передаче данных по каналу связи используют только два состояния электрического сигнала (например, два значения напряжения или тока). При этом скорость передачи данных в бодах будет совпадать со скоростью передачи в битах за секунду. При пересылке данных часть полосы пропускания канала связи используется для передачи служебных сигналов. Поэтому реальная скорость передачи пользовательских данных будет несколько ниже, чем максимальная скорость канала связи. Например, при передаче 8-битовых символов по последовательному каналу связи на самом деле передается 10-битовая последовательность (первый и последний биты — служебные, так называемые стартовый и стоповый биты). Поэтому по каналу связи, максимальная скорость которого равна 9600 бит/с, можно передавать всего 960 символов в секунду.

### **Брандмаэр (firewall)**

Специальное устройство, через которое внутренняя сеть организации подключается ко внешнему каналу, ведущему в Internet. Брандмаэры используются для защиты внутренней сети от несанкционированного доступа извне. Брандмаэр может быть реализован как на программном, так и аппаратном уровне.

### **Быстрый Ethernet (Fast Ethernet)**

Популярное название сети Ethernet стандарта *100Base-T*.

### **Векторно-дистанционный (vector-distance)**

Устаревшее название дистанционно-векторных протоколов маршрутизации.

### **Виртуальное соединение (virtual connection)**

Абстрактное понятие, принятое в протоколах, требующих установки соединения с получателем, например TCP. После того как соединение установлено, оно остается в открытом состоянии до тех пор, пока не будет выдан запрос на его аннулирование.

### **Внеочередные данные (out of band data)**

Данные, посылаемые получателю в обход основных каналов доставки. Обычно они используются для передачи сигналов, свидетельствующих о ненормальном завершении операции или сообщений об ошибках. В протоколе TCP для отправки внеочередных данных используется признак срочных данных.

### **Внимание маршрутизатора (router alert)**

Параметр протокола IP, наличие которого в дейтаграмме приводит к тому, что ее содержимое будет анализироваться всеми промежуточными маршрутизаторами, даже если дейтаграмма явно ими не должна обрабатываться.

### **Внутренняя сеть предприятия (intranet)**

Закрытая корпоративная сеть, состоящая из нескольких локальных сетей, объединенных друг с другом посредством маршрутизаторов, в которых используется семейство протоколов TCP/IP. Внутренняя сеть предприятия может иметь выход в глобальную сеть Internet.

### **Воспроизведение (replay)**

Ошибка ситуация, при которой пакеты предыдущего сеанса связи по ошибке интерпретируются как принадлежащие текущему или последующему сеансу. Использование протоколов в Internet, не препятствующих ошибочному воспроизведению, не обеспечивает требуемый уровень безопасности.

### **Время кругового обращения (round trip time)**

См. *RTT*.

### **Всемирное время (universal time)**

Международный стандарт обозначения времени, который раньше назывался средним временем по Гринвичу (Greenwich Mean Time, или GMT). Он также называется всемирным координированным временем (universal coordinated time, или UTC)

### **Выделенные номера (Assigned Numbers)**

Документ RFC, в котором описаны все величины (как правило, числовые), используемые в семействе протоколов TCP/IP.

### **Выравнивание трафика (traffic shaping)**

Название одного из механизмов, используемого в системах, гарантирующих требуемое качество обслуживания. Суть его состоит в том, что входящий трафик помещается в буфер, откуда он считывается с фиксированной скоростью.

### **Гбит/с (Gbps)**

*Gigabits Per Second*, или *гигабит в секунду*. Единица измерения скорости передачи данных в канале связи, равная  $2^{30}$  бит/с. См. также *Кбит/с*, *Мбит/с* и *бод*.

### **Группа всех маршрутизаторов (all routers group)**

Стандартная многоадресатная группа протокола IP, в которую включены все маршрутизаторы, подключенные к данной локальной сети.

### **Группа всех машин (all systems group)**

Стандартная многоадресатная группа протокола IP, в которую включены все узлы сети и маршрутизаторы, подключенные к данной локальной сети.

### **Движущееся окно (sliding window)**

Одна из характеристик протокола, позволяющего отправителю посыпать более одного пакета до получения сигнала подтверждения приема предыдущего пакета. Получив сигнал подтверждения приема, отправитель “сдвигает” окно передачи пакетов и посыпает следующий по порядку пакет. Количество посыпаемых пакетов без ожидания сигнала подтверждения приема зависит от размера окна. Увеличение размера окна приводит к повышению производительности работы сети.

### **Двойная экспоненциальная задержка (binary exponential backoff)**

Методика, используемая для управления передачей данных в сети, а также для быстрой ликвидации перегрузки. Суть ее состоит в том, что после каждой успешной попытки доступа в сеть отправитель должен удвоить значение величины задержки перед попыткой последующего доступа в сеть.

### **Дейтаграмма (datagram)**

См. *IP-дейтаграмма*.

## **Демультиплексирование (demultiplex)**

Процесс разделения общего входящего потока данных на несколько отдельных исходящих потоков. Демультиплексирование выполняется на нескольких уровнях. Например, демультиплексирование на аппаратном уровне позволяет выделить из общего канала передачи данных сигнал, соответствующий одному потоку данных. Этот процесс выполняется на основании временных или частотных характеристик сигнала. Благодаря демультиплексированию по одному физическому кабелю можно передавать несколько потоков данных. Программы поддержки протокола IP также выполняют демультиплексирование входящих дейтаграмм и перенаправляют их на обработку соответствующему модулю протокола более высокого уровня или прикладной программе. См. [мультиплексирование](#).

## **Дерево источников (source tree)**

Синоним термина *дерево кратчайшего пути*.

## **Дерево кратчайшего пути (shortest path tree)**

Дерево пересылки многоадресатных пакетов по оптимальным маршрутам от отправителя до всех членов группы. Алгоритм с использованием дерева кратчайшего пути является альтернативой алгоритму с общим деревом пересылки.

## **Дистанционно-векторный протокол маршрутизации (distance-vector routing protocol )**

Класс протоколов обновления маршрутной информации, использующих распределенный алгоритм поиска кратчайшего пути (SPF). При этом каждый задействованный маршрутизатор посылает соседним маршрутизаторам список сетей, которые могут быть через него достигнуты, и метрику расстояния до каждой из них.

## **Домен (domain)**

Часть иерархии имен системы DNS. Синтаксически доменное имя состоит из последовательности имен (меток), разделенных точкой.

## **Дополнительный заголовок (extension header)**

Один из необязательных заголовков протокола IPv6, который указывается после основного заголовка.

## **Дополнительный заголовок фрагментации (fragment extension header)**

Один из необязательных заголовков протокола IPv6, который используется для идентификации фрагмента дейтаграммы.

## **Достижимость (reachability)**

Считается, что сеть *достигима* с некоторого узла, если дейтаграмма, отправленная с этого узла гарантированно попадет в сеть, к которой подключен конечный получатель. Обмен информацией о достижимости сетей осуществляется с помощью одного из внешних протоколов маршрутизации.

## **Дрожание (jitter)**

Технический термин, означающий изменение задержки в процессе доставки упорядоченной последовательности пакетов. Одной из основных причин возникновения эффекта дрожания является неравномерность трафика, передающегося по объединенной сети.

## **Дуплексный режим (full duplex)**

Одна из характеристик метода передачи данных, при котором возможна одновременная передача данных в обоих направлениях. Дуплексный режим реализован в протоколе TCP. Ср. с *половудуплексным режимом*.

### **Заголовок (header)**

Служебная информация, помещаемая в начало каждого пакета или сообщения, с помощью которой описывается содержимое пакета или указывается адрес получателя.

### **Задержка (delay)**

Один из двух основных параметров сети. Величина задержки характеризует время, в течение которого бит данных передается по сети от отправителя до конечного получателя.

### **Закрытое окно (closed window)**

Ситуация, возникающая при использовании протокола TCP; при исчерпании пространства внутреннего буфера, использующегося для временного хранения поступающих по сети данных, получатель анонсирует отправителю окно нулевого размера. При этом отправитель должен немедленно прекратить отправку данных до открытия окна получателем.

### **Замораживание изменений (hold down)**

Методика запрета внесения изменений в таблицу маршрутизации на протяжении небольшого интервала времени после ее первоначального изменения. Замораживание препятствует возникновению маршрутных петель.

### **Зарезервированный адрес (reserved address)**

Синоним термина *немаршрутизуемый адрес*.

### **Зона (area)**

В протоколе OSPF группа маршрутизаторов, обменивающихся маршрутной информацией.

### **Зона полномочий (zone of authority)**

Термин, используемый в системе доменных имен для обозначения части пространства имен, право на управление которыми передано конкретному серверу имен. Для управления каждой зоной должны выделяться как минимум два сервера имен, у которых не должно быть общих предпосылок для сбоя.

### **Зона полномочий (authority zone)**

Часть иерархии доменных имен, полномочия на обслуживание которой переданы одному серверу имен.

### **Иерархическая адресация (hierarchical addressing)**

Схема адресации, в которой сетевой адрес разделяется на части, каждая из которых идентифицирует отдельный уровень иерархии. В IP-адресации используется двухуровневая иерархия, при которой первая часть адреса идентифицирует сеть, а вторая — конкретный узел в этой сети. Сетевая часть адреса используется маршрутизатором для перенаправления дейтаграммы следующему маршрутизатору, расположенному по пути следования дейтаграммы к конечному получателю. Так продолжается до тех пор, пока дейтаграмма не достигнет маршрутизатора, непосредственно подключенного к сети конечного получателя, после чего она напрямую пересыпается получателю. При использовании подсетей в схему адресации протокола IP вносится дополнительный уровень иерархии.

### **Избирательное подтверждение приема (selective acknowledgement)**

См. *SACK*.

### **Издержки использования ячеек (cell tax)**

Проблема, возникающая в сети ATM вследствие того, что размер заголовка ячейки составляет 10% от ее общей длины.

### **Изохронный (isochronous)**

Одна из характеристик сетевой системы, в которой не возникает дрожание. Это означает что вся система, включая ее цифровые схемы, должна проектироваться так, чтобы доставка цифрового потока данных выполнялась с той же самой тактовой частотой, которая была использована для его генерации (т.е. синхронно с отправителем). Таким образом, если в изохронной системе существует несколько путей между любыми двумя конечными точками, то система должна обеспечивать для всех путей одинаковое время задержки передачи данных. Обычная телефонная система является изохронной.

### **Инкапсуляция (encapsulation)**

Методика, используемая в иерархической системе сетевых протоколов для передачи пакетов данных с уровня на уровень. При приеме сообщения от модуля протокола более высокого уровня оно помещается в область данных фрейма протокола текущего уровня. Процесс инкапсуляции выполняется на каждом из уровней протоколов. А это означает, что при передаче по физической сети в начале дейтаграммы будет находиться последовательность заголовков, первый из которых соответствует физическому сетевому фрейму, второй — протоколу IP, третий — транспортному протоколу и т.д.

### **Инкапсуляция трейлера (trailer encapsulation)**

Не совсем обычный метод инкапсуляции IP-дейтаграмм для передачи по физической сети, при использовании которого служебная информация помещается в конце пакета. Трейлеры применяются в сети Ethernet для выравнивания данных на границу страницы. Кроме того, трейлеры активно используются в протоколе адаптации ATM уровня 5.

### **Интерфейс Windows Sockets (Windows Sockets Interface)**

Версия API сокетов, разработанная компанией Microsoft. Обычно его называют WINSOCK.

### **Исторический (historic)**

Классификация, принятая IETF, которая присваивается протоколу, если его дальнейшее использование нежелательно. Протокол, которому присвоен статус “исторический”, является устаревшим.

### **Кабель пятой категории (category 5 cable)**

Стандарт кабеля типа витая пара, который используется при прокладке сетей Ethernet.

### **Класс адреса (class of address)**

Категория, к которой относится IP-адрес. Класс адреса определяет границу его раздела на префикс сети и суффикс узла.

### **Класс трафика (traffic class)**

Название набора служб, поддерживаемых при интерпретации поля дифференцированного обслуживания.

### **Классовая адресация (classful addressing)**

Первоначальная система адресации протокола IPv4, в которой адрес узла относился к одному из трех классов: А, В или С.

### **Кластерный адрес (cluster address)**

Термин, первоначально использовавшийся для обозначения *альтернативного (anycast) адреса*.

### **Кодек (codec)**

*Coder/decoder*, или *кодер/декодер*. Устройство, использующееся для преобразования аналогового сигнала в поток цифровых данных.

### **Контрольная сумма (checksum)**

Целое число, полученное в результате суммирования последовательности октетов сообщения, которые рассматриваются как целые числа. Контрольная сумма используется для обнаружения ошибок, возникающих в результате передачи данных по каналу связи от одной машины до другой. Обычно контрольная сумма вычисляется программами поддержки протоколов и помещается в пакет непосредственно перед его отправкой в сеть. После получения пакета программы поддержки протокола снова вычисляют его контрольную сумму и сравнивают ее с той, что указана в пакете. При несовпадении фиксируется факт ошибки. Во многих протоколах семейства TCP/IP применяется 16-битовое поле контрольной суммы, а для его вычисления используется двоичная арифметика с представлением отрицательных чисел в дополнительном коде. При этом значения всех целых полей пакета представляются в сетевом порядке следования байтов. Ср. с *CRC*.

### **Концентратор (hub)**

Недорогое электронное устройство, к которому подключается несколько компьютеров (обычно по кабелю типа “витая пара”) для обмена пакетами между собой. Концентраторы работают на втором уровне и выполняют повторение сигналов. Чаще всего используются концентраторы сети Ethernet.

### **Корректный разрыв соединения (graceful shutdown)**

Предусмотренный в протоколе специальный механизм, который позволяет двум взаимодействующим сторонам четко согласовывать между собой момент разрыва соединения, даже если часть пакетов будет утеряна, задержана или продублирована. В протоколе TCP используется трехэтапный метод квитирования, который позволяет гарантировать корректность разрыва соединения.

### **Край (leaf)**

Термин, взятый из теории графов и используемый для обозначения последнего маршрутизатора, расположенного на каждом из ведущих от отправителя дейтаграммы маршрутов или сети, подключенной к краевому маршрутизатору.

### **Кумулятивная система подтверждения приема (cumulative acknowledgement)**

Альтернатива селективной системе подтверждения приема, используемая в протоколе TCP. При кумулятивной системе отправителю посыпается уведомление об успешном приеме всего блока данных, а не его отдельных фрагментов.

### **Логическая подсеть (logical subnet)**

См. *LIS*.

## **Магистральная сеть (backbone network)**

Сеть, являющаяся основой объединенной сети; к ней подключаются другие сети. Национальная магистральная сеть является глобальной распределенной сетью, а корпоративная магистраль может быть обычной локальной сетью.

## **Максимальная единица передачи данных в сети (maximum transfer unit)**

См. *MTU*.

## **Максимальное количество переходов (hop limit)**

Название одного из полей заголовка дейтаграммы протокола IPv6, которое в протоколе IPv4 называлось *временем жизни дейтаграммы*. Данное поле предназначено для предотвращения возникновения маршрутных петель. При прохождении дейтаграммы через каждый из маршрутизаторов, расположенных по пути ее следования до конечного получателя значение этого поля уменьшается на единицу.

## **Маркерное кольцо (token ring)**

В широком смысле — тип сетевой технологии, в которой для управления доступом к среде передачи данных используется специальный тип пакета, называемый маркером. Маркер последовательно передается от машины к машине. Компьютер может передавать данные только в те моменты времени, когда он владеет маркером. В узком смысле — это название сетевого оборудования, производимого фирмой IBM, с помощью которого реализуются сети с маркерным кольцом (их еще называют сетями с эстафетной передачей данных, или *token ring*).

## **Марсианин (martians)**

Юмористический термин, применяемый к пакету, который по ошибке попал не в свою сеть, обычно из-за ошибок в таблицах маршрутизации.

## **Маршрут (route)**

В широком смысле маршрут — это путь, по которому сетевой трафик проходит от отправителя до конечного получателя. В объединенной сети на основе протокола TCP/IP маршрутизация каждой IP-дейтаграммы выполняется независимо от других дейтаграмм. Кроме того, некоторые маршруты могут динамически изменяться.

## **Маршрутизатор (router)**

Специализированный выделенный компьютер, который подключается к двум или более сетям и выполняет пересылку пакетов между ними. В частности, IP-маршрутизатор пересыпает IP-дейтаграммы между сетями, к которым он подключен. Для выбора адреса ближайшей точки перехода, по которому будет передаваться дейтаграмма, маршрутизатор использует адрес получателя дейтаграммы. Разработчики сетей сначала называли маршрутизатор *шлюзом*.

## **Маршрутизация между автономными системами**

Синоним внешней маршрутизации. В настоящее время в качестве протокола для внешней маршрутизации чаще всего используется протокол *BGP-4*.

## **Маршрутизация методом поиска кратчайшего пути (shortest path routing)**

Алгоритм маршрутизации, в котором дейтаграммы направляются к конечному получателю по кратчайшему пути. Во всех протоколах маршрутизации выполняется поиск кратчайших путей к получателю, однако критерии этого поиска разные. См. *SPF*.

### **Маршрутизация на основе отслеживания состояния соединения (link state routing)**

Один из двух возможных методов, используемых в протоколах маршрутизации, когда маршрутизатор рассыпает в режиме широковещания сообщения о состоянии соединения, а для поиска кратчайших путей к получателю дейтаграммы использует алгоритм Дейкстры. См. *дистанционно-векторный протокол маршрутизации*.

### **Маршрутизация на основе типа обслуживания (type of service routing)**

Одна из систем маршрутизации, в которой, кроме расстояния до конечного получателя, учитываются также характеристики физических сетей, по которым будет передаваться дейтаграмма.

### **Маршрутизация от источника (source route)**

Маршрут следования пакета, который заранее определен отправителем. В протоколе IP маршрут от источника задается в виде параметра протокола. В нем указывается список адресов маршрутизаторов, через которые дейтаграмма должна обязательно пройти. Маршрутизация от источника часто используется при выявлении неполадок в сети. См. также *LSR* и *SSR*.

### **Маршрутная петля (routing loop)**

Ошибка ситуация при которой несколько маршрутизаторов “перебрасывают” пакеты между собой по кругу, поскольку этот путь указан в их таблицах маршрутизации как кратчайший, ведущий к конечному получателю.

### **Маска (mask)**

См. *маска подсети*.

### **Маска адреса (address mask)**

Синоним *маски подсети*.

### **Маска подсети (subnet mask)**

Битовая маска, используемая для выделения из IP-адреса той части, которая соответствует адресу сети. Размер маски подсети составляет 32 бита. Биты, имеющие единичное значение, соответствуют сетевой части IP-адреса, а нулевое значение — части, которая идентифицирует узел в данной сети.

### **Мгновенное изменение (triggered update)**

Эвристический алгоритм, используемый в дистанционно-векторных протоколах, таких как RIP. При изменении информации в таблице маршрутизации маршрутизатор немедленно посыпает сообщение об обновлении маршрутной информации, не дожидаясь наступления очередного момента отсылки следующего периодического сообщения.

### **Медленная сходимость (slow convergence)**

Проблема, возникающая при использовании дистанционно-векторных протоколов маршрутизации. Ее называют также проблемой подсчета до бесконечности. Из-за медленного распространения по сети сообщений об обновлении, в маршрутной информации возникают противоречия. При этом образуются маршрутные петли, в которых действуются два или более маршрутизаторов. Для частичного решения проблемы медленной сходимости необходимо ограничить значение, соответствующее бесконечности (в случае протокола RIP это 16).

### **Медленный запуск (slow-start)**

Один из способов предотвращения перегрузки в сети, используемый в протоколе TCP. При получении сигнала подтверждения приема размер окна протокола TCP увеличивается на один сегмент. Термин может немного сбить с толку, поскольку медленный запуск предназначен для быстрого достижения высокой пропускной способности сети путем экспоненциального увеличения размера окна.

### **Международная организация по стандартизации**

См. ISO.

### **Международный телекоммуникационный союз (International Telecommunications Union, ITU)**

Международная организация, разрабатывающая стандарты обмена данными в области телефонии. Ею был принят стандарт для сетевых протоколов семейства X.25. Следует отметить, что в Европе службы почтово-телеграфной и телефонной связи (PTT) предлагают сразу два вида услуг: передача голоса (телефон) и передача данных (сети X.25)

### **Многоадресатная передача, управляемая данными (data-driven multicast)**

Одна из схем многоадресатной пересылки пакетов, в которой используется алгоритм широковещания и отсечения. См. многоадресатная передача, управляемая по запросу.

### **Многоадресатная передача, управляемая по запросу (demand-driven multicast)**

Одна из схем многоадресатной пересылки пакетов, в которой для доставки пакетов используется алгоритм анализа маршрутизатором общего дерева пересылки. См. многоадресатная передача, управляемая данными.

### **Многоадресатная рассылка (multicast)**

Методика, благодаря которой копию одного пакета можно доставлять выбранному подмножеству машин. Многоадресатная рассылка поддерживается на аппаратном уровне в некоторых сетевых технологиях (например, Ethernet) путем подключения сетевой интерфейсной платы к одной или нескольким группам. В протоколе IP многоадресатная рассылка поддерживается в пределах всей объединенной сети.

### **Многоадресатная рассылка IP-дейтаграмм (IP multicast)**

Одна из схем адресации и перенаправления, которая позволяет передавать IP-дейтаграммы одновременно некоторому подмножеству узлов сети. В настоящее время в Internet маршрутизация многоадресатных IP-дейтаграмм повсеместно не поддерживается.

### **Многоадресный узел (multi-homed host)**

Узел сети, работающий под управлением протокола TCP/IP, который подключен к двум или более физическим сетям.

### **Мобильный протокол IP (mobile IP)**

Технология, разработанная IETF, позволяющая перемещать компьютер в другую сеть без изменения его первоначального IP-адреса. При подключении к новой сети, компьютер обращается к местному серверу за получением второго (временного) IP-адреса, после чего связывается с агентом, находящимся в "домашней" сети, и дает ему указание пересыпать все дейтаграммы по временному адресу.

### **Модель взаимодействия типа клиент/сервер (client-server model)**

Модель взаимодействия в распределенной системе, в которой программа, запущенная на одном компьютере, посыпает запросы программе, работающей на другом компьютере, и ожидает от нее ответа. Программа, посыпающая запросы, называется клиентом, а программа, отвечающая на запросы, — сервером. Обычно клиентская часть программы намного проще, чем серверная.

### **Мост (bridge)**

Устройство, соединяющее два или более сегмента сети и пересылающее пакеты между ними. Мости взаимодействуют с сетью на физическом уровне. Например, в сети Ethernet с помощью моста можно соединить два физических отрезка кабеля (или сегмента) так, чтобы с точки зрения сетевого программного обеспечения это выглядело как один сплошной физический сегмент. При этом мост будет пересыпать только те пакеты, которые не предназначены для получателей, находящихся в текущем сегменте. Следует различать мости и повторители. Мости выполняют прием, хранение и пересылку данных пакета, тогда как повторители — только передачу аналоговых сигналов (включая помехи) из одного кабеля в другой. В то же время мости отличаются от маршрутизаторов тем, что мости функционируют на физическом уровне сети и оперируют физическими адресами устройств, тогда как маршрутизаторы работают на сетевом уровне и оперируют IP-адресами.

### **Мультиплексирование (multiplex)**

Процесс объединения нескольких потоков данных в единый поток для передачи по одному каналу связи так, чтобы впоследствии эти потоки можно было легко разделить. Мультиплексирование данных может выполняться на нескольких уровнях. См. *демультиплексирование*.

### **Мультипликативное уменьшение (multiplicative decrease)**

Методика, используемая в протоколе TCP для снижения потока данных при возникновении перегрузки в сети. Каждый раз при обнаружении потери сегмента в протоколе TCP текущий размер окна уменьшается в два раза.

### **Набор стандартов ОС**

Серия стандартов для высокоскоростной передачи данных по оптоволоконным сетям. Например, широко распространенный стандарт OC3 позволяет передавать данные со скоростью 155 Мбит/с.

### **Надежная многоадресатная система доставки (reliable multicast system)**

Система доставки многоадресатных дейтаграмм, которая гарантирует их доставку каждому члену группы.

### **Надежная пересылка (reliable transfer)**

Одна из характеристик транспортного протокола, который обеспечивает гарантированную доставку дейтаграмм получателю без потерь, повреждения, дублирования и нарушения их порядка следования. Если такая доставка невозможна, отправитель оповещается об этом.

### **Направленный широковещательный адрес (directed broadcast address)**

Особый тип IP-адреса, которому соответствуют все узлы в данной подсети. Пакет, посланный по этому адресу, вначале доставляется в указанную сеть, а затем его копия рассыпается всем узлам этой сети.

### **Наращивание (graft)**

Операция, обратная *отсечению*, при выполнении которой многоадресатный маршрутизатор присоединяется к ветви общего дерева пересылки.

### **Насыщение сети Ethernet (Ethernet meltdown)**

Состояние перегрузки или близкое к ней в сети Ethernet. Обычно состояние насыщения является следствием отправки пакета по некорректному адресу или неправильной работы системы маршрутизации и, как правило, продолжается в течение короткого промежутка времени.

### **Негарантированная доставка (best-effort delivery)**

Одна из характеристик сетевой технологии, которая не обеспечивает надежность передачи данных на канальном уровне. Протокол IP также не обеспечивает надежность передачи данных на сетевом уровне, поскольку он рассчитан на использование сетевого оборудования, не гарантирующего доставку пакетов. Еще одним примером протокола, не гарантирующего доставку данных на уровне приложений, является UDP.

### **Нэйгла, алгоритм (Nagle, algorithm)**

Самосинхронизирующийся эвристический алгоритм, позволяющий сгруппировать выходные данные в буфере для улучшения пропускной способности сети и избежания синдрома полного окна.

### **Немаршрутизуемый адрес (nonroutable address)**

Адрес, в котором используется префикс сети, зарезервированный для использования во внутренних сетях предприятий. Если маршрутизатор в глобальной сети Internet получит дейтаграмму, содержащую немаршрутизуемый адрес, он должен ее аннулировать и переслать отправителю сообщение об ошибке. См. *адреса из сети 10*.

### **Ненадежная доставка (unreliable delivery)**

Характеристика механизма доставки, который не гарантирует отсутствие потерь, повреждения и дублирования данных, а также то, что данные поступают получателю точно в том порядке, в каком они были отправлены. В протоколе IP реализован ненадежный механизм доставки.

### **Ненумерованная сеть (unnumbered network)**

Одна из методик экономии адресного пространства протокола IP, при использовании которой двухточечной сети, соединяющей два маршрутизатора, не выделяется отдельного префикса IP-адреса (т.е. сеть остается ненумерованной, или анонимной).

### **Непрямая доставка (indirect delivery)**

Процесс доставки дейтаграммы посредством маршрутизатора. Существует также прямая доставка, при которой дейтаграмма посылается отправителем непосредственно конечному получателю.

### **Неустойчивое состояние (soft state)**

Одна из технологий управления кэшированием, при использовании которой информация, хранящаяся у получателя, считается устаревшей по истечении определенного времени без явного уведомления об этом ее отправителя. Данная технология применяется в тех случаях, когда отправитель и получатель периодически отключаются от сети.

### **Обновление разделенного горизонта (split horizon update)**

Эвристический алгоритм, используемый в дистанционно-векторных протоколах маршрутизации, таких как RIP, для решения проблемы *медленной сходимости* (т.е. возникновения маршрутных петель). При использовании этой методики маршрутизатор не должен распространять маршрутную информацию обратно по тому же интерфейсу, из которого она была первоначально получена.

### **Оборудование 10/100 (10/100 hardware)**

Этот термин применяется к любому сетевому оборудованию стандарта Ethernet, которое может работать как на скорости 10 Мбит/с, так и на скорости 100 Мбит/с.

### **Обратное исправление (poison reverse)**

Эвристический алгоритм, используемый в дистанционно-векторных протоколах маршрутизации, таких как RIP, который позволяет избежать образования маршрутных петель. Когда обрывается связь с какой-либо сетью, анонсирующий ее маршрутизатор сохраняет в своей таблице данные об этой сети на время передачи нескольких периодических сообщений об обновлении. При этом в широковещательных сообщениях указывается бесконечная стоимость маршрута к сети, с которой отсутствует связь.

### **Обратный порядок следования байтов (битов) (big endian)**

Формат, используемый для хранения или передачи бинарных данных, в котором старший по значимости байт (или бит) хранится в памяти по младшему адресу, или передается первым. Подобный порядок следования байтов принят в протоколе TCP/IP. Ср. с *прямым порядком* следования байтов.

### **Общее дерево (shared tree)**

Один из методов пересылки пакетов, используемый в многоадресатных протоколах маршрутизации, управляемых по запросу. Алгоритм с общим деревом пересылки является альтернативой алгоритму с деревом кратчайшего пути.

### **Объединение маршрутов (route aggregation)**

Методика, используемая в протоколах маршрутизации для уменьшения размера таблицы маршрутизации. Все маршруты к конечным получателям, имеющие общую ближайшую точку перехода, объединяются и указываются в таблице маршрутизации в виде одного элемента. Высшая степень объединения маршрутов достигается в элементе таблицы маршрутизации, содержащем сведения о стандартном маршруте.

### **Объединенная сеть (internet)**

Конструктивно — это набор сетей с коммутацией пакетов, соединенных между собой посредством маршрутизаторов. Благодаря использованию семейства протоколов TCP/IP для пользователя создается впечатление, что объединенная сеть работает как одна большая виртуальная сеть. При написании слова “Internet” с прописной буквы имеется в виду глобальная сеть.

### **Ограничение трафика (traffic policing)**

Название одного из механизмов, используемого в системах, гарантирующих требуемое качество обслуживания. Суть его состоит в том, что постоянно изменяется величина входного трафика и, если она превышает установленные пределы, “лишний” трафик аннулируется.

### **Одноадресатная рассылка (unicast)**

Метод адресации и маршрутизации дейтаграмм, когда каждый пакет доставляется только одному получателю. Большинство IP-дейтаграмм являются одноадресатными. См. также *многоадресатная рассылка*.

### **Одноадресный маршрутизатор (one-armed router)**

IP-маршрутизатор, который поддерживает два домена адресов, но имеет только одно физическое подключение к сети. Такие маршрутизаторы, как правило, используются для преобразования адресов или повышения системы безопасности, а не для пересылки пакетов между сетями. Их также называют *одноадресными брандмауэрами*.

### **Одноуровневое пространство имен (flat namespace)**

Одна из характеристик системы присвоения имен, при использовании которой имена объектов фиксируются в виде одномерного списка строк. В качестве примера одноуровневой системы имен можно назвать список улиц обычного города. Одноуровневая система присвоения имен отличается от иерархической системы тем, что в последней все пространство имен делится на зоны, соответствующие иерархии административного органа, который ими управляет.

### **Окно (window)**

См. *движущееся окно*.

### **Окно нулевого размера (zero window)**

См. *закрытое окно*.

### **Октет (octet)**

Единица передачи данных в сети, состоящая из 8 бит. Несмотря на то что разработчики часто используют термин *байт* как синоним слова *октет*, строго говоря размерность байта может быть больше или меньше 8 бит.

### **Основной заголовок (base header)**

В предложенном стандарте протокола IPng обязательный заголовок, расположенный в начале каждой дейтаграммы.

### **Открытие соединения в активном режиме (active open)**

Операция, которую выполняет клиент при установке TCP-соединения с сервером, если ему известен адрес сервера.

### **Отсечение (prune)**

Операция, обратная *наращиванию*, при выполнении которой многоадресатный маршрутизатор удаляет свой адрес из общего дерева пересылки.

### **Отсроченное подтверждение приема (delayed acknowledgement)**

Эвристический алгоритм, используемый получателем пакетов протокола TCP для устранения проблемы, называемой синдромом полного окна.

### **Пакет (packet)**

Неформальный термин, часто используемый для обозначения небольшого блока данных, посылаемых по сети с коммутацией пакетов.

### **Партнерское соглашение (peering arrangement)**

Соглашение о сотрудничестве, заключенное между двумя провайдерами, в котором оговариваются условия обмена информацией о достижимости сетей и пе-

редачи пакетов данных. Обычно обмен трафиком происходит в точках доступа к сети, однако крупные провайдеры Internet часто заключают между собой дополнительные соглашения.

#### **Переадресация (redirect)**

Сообщение протокола ICMP, посыпаемое маршрутизатором узлу локальной сети, в котором содержится запрос на изменение информации в таблице маршрутизации узла сети.

#### **Перегрузка (congestion)**

Ситуация, при которой величина пересылаемого по сети потока данных временно превышает пропускную способность канала связи или маршрутизатора. В протокол TCP включен специальный механизм устранения перегрузки, который позволяет быстро вывести систему из этого состояния.

#### **Перегрузка уведомлениями (ACK implosion)**

Проблема, возникающая при использовании надежного многоадресатного протокола, когда многочисленные сигналы подтверждения приема посыпаются обратно отправителю пакета. В большинстве схем многоадресатной рассылки для обработки этих сигналов используются специализированные маршрутизаторы.

#### **Перенаправление (forwarding)**

Процесс переброски входящего пакета из сети в сеть. Поиск нужной сети выполняется на основании просмотра таблицы маршрутизации, в которой указан адрес ближайшей точки перехода. Пересылку дейтаграмм выполняют маршрутизаторы протокола IP.

#### **Пересылка с промежуточным хранением (store-and-forward)**

Принцип перенаправления дейтаграмм, используемый в IP-маршрутизаторах, при использовании которого входящая дейтаграмма сохраняется в памяти маршрутизатора до тех пор, пока она полностью не будет передана следующему получателю.

#### **Периметр безопасности (perimeter security)**

Одна из методик обеспечения безопасности в сетях, при которой на каждое внешнее соединение организации устанавливается брандмауэр, защищающий ее внутреннюю сеть от несанкционированного доступа из внешних сетей.

#### **Повторитель (repeater)**

Электронное устройство, с помощью которого можно удлинить сегмент локальной сети. Повторитель обеспечивает усиление всех аналоговых сигналов, передающихся в сетевом кабеле (в том числе и помех) и их передачу из одного сегмента сети в другой. В настоящее время используется редко.

#### **Подавление источника (source quench)**

Один из методов устранения перегрузки узла сети. В случае обнаружения перегрузки узел сети посылает специальное сообщение источнику данных, предписывающее последнему прекратить отправку пакетов. В объединенной сети на основе протокола TCP/IP маршрутизатор посылает отправителю пакетов ICMP-сообщение о подавлении источника данных, если его входная очередь переполнена.

#### **Подсеть переменной длины (variable-length subnetting)**

Одна из схем назначения адресов подсетей, при использовании которой в некоторых физических сетях организации могут использоваться разные маски

подсети. Существует также альтернативная схема с использованием *подсетей постоянной длины*.

#### **Подсеть постоянной длины (fixed-length subnetting)**

Одна из схем назначения адресов подсетей, при использовании которой во всех физических сетях организации используется одна и та же маска подсети. Существует также альтернативная схема с использованием *подсетей переменной длины*.

#### **Подтверждение приема (acknowledgement)**

Ответный сигнал, посылаемый получателем для индикации факта успешного получения информации. Сигналы подтверждения приема могут использоваться на любом уровне. На физическом уровне они реализуются в виде электрических сигналов, передаваемых по одному или нескольким проводам, которые координируют передачу данных. На канальном уровне сигналы подтверждения приема используются для индикации факта успешной передачи данных по одному физическому каналу связи. На более высоких уровнях сигналы подтверждения приема позволяют программе-получателю послать сигнал программе-отправителю.

#### **Поиск значения MTU по пути следования пакетов (path MTU)**

Процесс определения минимального значения MTU по пути следования пакета от отправителя до конечного получателя. Минимальное значение MTU определяет максимальный размер дейтаграммы, которая может передаваться от отправителя до конечного получателя без фрагментации. В стандарте протокола IP рекомендуется использовать механизм поиска значения MTU по пути следования пакетов.

#### **Положительное уведомление (positive acknowledgement)**

Синоним термина *подтверждение приема*.

#### **Полудуплексный (half duplex)**

Одна из характеристик метода передачи данных, при котором в заданный момент времени возможна передача данных только в одном направлении. Ср. с *дуплексным режимом*.

#### **Порт (port)**

См. *порт протокола*.

#### **Порт протокола (protocol port)**

Абстрактное понятие, используемое в транспортных протоколах семейства TCP/IP для того, чтобы можно было различать получателей дейтаграмм в пределах одного узла сети. В семействе протоколов TCP/IP для идентификации номеров портов используются небольшие положительные целые числа. Как правило, в операционной системе предусмотрены средства, с помощью которых прикладная программа может запросить свободный номер порта, который затем она будет использовать. Часть номеров портов зарезервирована для использования стандартными сетевыми службами, такими как, например, служба электронной почты.

#### **Поток (flow)**

Обобщенный термин, используемый для характеристики последовательности пакетов, посылаемых от отправителя до конечного получателя. В некоторых сетевых технологиях для обмена данными между любыми двумя взаимодействующими приложениями используется отдельный поток, в других — все пакеты, передаваемые между двумя узлами сети, пересыпаются в одном потоке.

### **Почтовый мост (mail bridge)**

Неофициальный термин, используемым как синоним *почтового шлюза*.

### **Почтовый сервер (mail exchanger)**

Компьютер, который получает электронную почту. Некоторые из почтовых серверов выполняют пересылку почты другим почтовым серверам. В системе DNS для почтовых серверов предусмотрен специальный тип записи MX.

### **Почтовый шлюз (mail gateway)**

Специализированный компьютер, подключенный к двум или более системам электронной почты (обычно разнотипным, находящимся в разных сетях) и выполняющий пересылку сообщений между ними. Почтовые шлюзы обычно получают сообщение целиком, переформатируют его в соответствии с правилами, принятыми в почтовой системе получателя, и пересылают получателю.

### **Представление внешних данных (External Data Representation)**

См. *XDR*.

### **Преобразование (resolution)**

См. *преобразование адреса*.

### **Преобразование адреса (address resolution)**

Процесс преобразования протокольного адреса в соответствующий ему физический адрес, например преобразование IP-адреса компьютера в физический адрес его сетевой платы Ethernet. В зависимости от применяемого сетевого оборудования для выполнения этого преобразования иногда необходимо послать в локальную сеть широковещательный запрос. См. *ARP*

### **Преобразование имен (name resolution)**

Процесс поиска по имени соответствующего ему адреса, выполняемый системой DNS. Для этого пользовательская программа должна обратиться к специальному удаленному серверу имен, содержащему распределенную базу данных имен и адресов.

### **Префикс провайдера (provider prefix)**

Схема адресации, при которой каждому провайдеру услуг Internet выделяется блок IP-адресов, идентифицируемый одним префиксом, которые затем раздаются его клиентам. Префиксная система адресации поддерживается в протоколе IPv6.

### **Принцип сервера приложений (application-server paradigm)**

Синоним взаимодействия по принципу *клиент/сервер*.

### **Проблема 2Х**

Проблема неэффективности маршрутизации, возникающая при использовании мобильного протокола IP. Она выражается в том, что дейтаграммы, посланные от узла локальной сети мобильному узлу, подключенному к этой же локальной сети, два раза передаются по глобальной сети Internet: сначала из локальной сети — в “домашнюю” сеть мобильного компьютера, где находится агент пересылки, а затем из “домашней” сети — снова в локальную сеть, к которой подключен мобильный компьютер.

### **Проблема дополнительных переходов (extra hop problem)**

Проблема, возникающая в системе маршрутизации, при которой дейтаграммы передаются к получателю по неоптимальному (т.е. не кратчайшему маршруту). Данную проблему очень трудно обнаружить, поскольку внешне никаких дефектов не наблюдается и все работает как надо.

### **Проблема подсчета до бесконечности (count to infinity)**

Синоним, который часто используется для обозначения *проблемы медленной сходимости*.

### **Проверка соединения (keepalive)**

Сообщение небольшого размера, периодически посылаемое между двумя взаимодействующими по сети объектами для проверки канала связи между ними и работоспособности обеих сторон. Механизм проверки соединения используется в протоколе *BGP*.

### **Прокси-сервер (proxy)**

Программное или аппаратное устройство, которое может выполнять функции другого устройства. Например, Web-сервер (сервер-посредник), который выполняет функции другого Web-сервера.

### **Проталкивание (push)**

Операция, выполняемая прикладной программой над открытым TCP-соединением, вызывающая немедленную передачу данных по каналу связи. Для запроса на проталкивание используется специальный бит заголовка TCP-сегмента.

### **Протокол Internet**

См. *IP*.

### **Протокол (protocol)**

Формальное описание формата сообщений и правил, которым должны следовать две или более машин для обмена этими сообщениями. Протоколы бывают низкоуровневые и высокоуровневые. С помощью первых описывается взаимодействие на низком уровне между сетевыми интерфейсами двух машин (т.е. способы формирования из последовательности октетов сообщения битового потока и передачи его по проводам). Вторая группа протоколов определяет порядок обмена сообщениями между прикладными программами (т.е. методы обмена файлами между двумя программами через объединенную сеть). В спецификацию большинства протоколов входят как словесное описание взаимодействия, так и строгий алгоритм работы протокола, в основу которого положена модель конечного автомата.

### **Протокол адаптации ATM уровня 5 (ATM Adaptation Layer 5, или AAL)**

Один из протоколов семейства ATM, определяющий способы передачи и получения информации прикладными программами с помощью сети ATM. Для передачи данных используется пятый уровень адаптации.

### **Протяженные сети (long haul network)**

Старое название глобальных сетей (*WAN*).

### **Прямой порядок следования байтов (little endian)**

Один из форматов представления двоичных чисел, передаваемых по сети, в котором младший по значимости байт (или бит) располагается по младшему адресу. См. *обратный порядок следования байтов*.

### **Псевдозаголовок (pseudo header)**

Дополнительный заголовок, содержащий IP-адреса отправителя и получателя, полученные из заголовка IP-дейтаграммы, который добавляется к пакету в процессе вычисления контрольной суммы в протоколах TCP или UDP.

### **Пятиуровневая эталонная модель**

Иерархическая модель, используемая в семействе протоколов TCP/IP. Хотя на момент принятия модель была весьма спорной, успешное использование семейства протоколов TCP/IP привело к тому, что она была широко принята.

### **Рабочая группа (working group)**

Группа лиц в IETF, которые работают над решением проблемы конкретного протокола или структуры сети.

### **Рабочие документы Internet (Internet Draft)**

Черновики документов, создаваемые IETF. После одобрения эти документы переходят в ранг RFC.

### **Разборщик почты (mail exploder)**

Часть системы электронной почты, которой в качестве входных параметров передается тело сообщения и список адресов получателей, выполняющая рассылку копии сообщения всем получателям, указанным в списке. В большинстве систем электронной почты имеется встроенный разборщик почты, что позволяет пользователям создавать собственные списки рассылки.

### **Региональная сеть (regional network)**

Сеть, которая покрывает средние расстояния, например несколько городов или географических областей.

### **Ретрансляция фреймов (Frame Relay)**

Название одной из сетевых технологий, ориентированных на установку соединения, обычно используемой телефонными компаниями.

### **Самовосстанавливающаяся (self-healing)**

Одна из характеристик системы, которая автоматически восстанавливает свою работоспособность после возникновения сбоя. Сеть с двумя кольцами FDDI является самовосстанавливающейся, поскольку при отказе одного из узлов или обрыве одного из колец работа сети не нарушается.

### **Самосинхронизирующаяся (self clocking)**

Одна из характеристик системы, работающей в циклическом режиме, для которого не требуется внешний источник синхронизации, например, при получении пакета выполняется какое-либо действие.

### **Сборка (reasembly)**

Накопление всех фрагментов IP-дейтаграммы и восстановление из них первоначальной IP-дейтаграммы. Сборка выполняется на машине конечного получателя.

### **Сборный пункт уведомлений (acknowledgement aggregator)**

Используется в надежных протоколах многоадресатной рассылки для устранения проблемы перегрузки уведомлениями.

**Сброс (reset)**

Сегмент протокола TCP, посылаемый отправителю дейтаграмм в случае возникновения ошибочной ситуации.

**Сверхвысокоскоростной магистральный канал  
(very high speed Backbone Network Service)**

См. *vBNS*.

**Сегмент (segment)**

Единица передачи данных между модулями протоколов TCP двух машин. Каждый сегмент является частью потока данных, передаваемого между двумя машинами. Текущее положение сегмента в потоке отмечается в его заголовке. Для обеспечения целостности передаваемых данных для каждого сегмента вычисляется значение контрольной суммы.

**Семейство протоколов TCP/IP (TCP/IP Internet Protocol Suite)**

Официальное название группы протоколов TCP/IP.

**Семиуровневая эталонная модель (seven-layer reference model)**

См. *ISO*.

**Сервер (server)**

Постоянно работающая программа, которая обеспечивает требуемый вид услуг для клиентов, подключающихся к ней с помощью сети. Например, доступ к файлам на сервере или к Web-страницам.

**Сервер маршрутизации (route server)**

Сервер, который находится в точке доступа к сети и поддерживает протокол BGP для обмена информацией о достижимости сетей между провайдерами, полученной из своей копии арбитражной базы данных.

**Сетевое управление (network management)**

См. *MIB* и *SNMP*.

**Сетевой порядок следования байтов (network byte order)**

Стандарт, входящий в семейство протоколов TCP/IP, для передачи целых чисел. В нем говорится, что старший по значимости байт должен передаваться по каналу связи первым (т.е. в сети принят *обратный порядок следования байтов*). При передаче отправитель должен преобразовать бинарные данные их из локального порядка следования в сетевой. В свою очередь получатель должен выполнить обратную операцию — преобразовать данные из сетевого порядка следования в локальный.

**Сеть Ethernet на основе витой пары (twisted pair Ethernet)**

Сеть Ethernet типа 10Base-T, при прокладке которой используется кабель на основе витой пары, соединяющий каждый компьютер с концентратором. См. *сеть Ethernet с толстым коаксиальным кабелем* и *сеть Ethernet с тонким коаксиальным кабелем*.

**Сеть Ethernet с толстым коаксиальным кабелем (thicknet)**

Название оригинального стандарта передачи данных по сети 10Base5 Ethernet с использованием толстого коаксиального кабеля. См. *сеть Ethernet с тонким коаксиальным кабелем*, *10Base2* и *10Base-T*.

### **Сеть Ethernet с тонким коаксиальным кабелем (thinnet)**

Название стандарта передачи данных по сети 10Base2 Ethernet с использованием тонкого и гибкого коаксиального кабеля. См. *сеть Ethernet с толстым коаксиальным кабелем, 10Base5* и *10Base-T*.

### **Сеть с двухточечным подключением (point-to-point network)**

Два компьютера, связанные друг с другом посредством любой сетевой технологии, например последовательного канала связи. Поскольку к двухточечной сети подключены только два компьютера, им не нужно назначать физический адрес. Сети с двухточечным подключением обычно используются для создания выделенного канала связи между двумя сетевыми центрами.

### **Сигнализация (signaling)**

Термин, взятый из телефонии, который обозначает набор протоколов, с помощью которых создается виртуальный канал с получателем.

### **Синдром полного окна (silly window syndrome)**

Состояние, которое может возникнуть в протоколе TCP, если получатель несколько раз подряд пошлет отправителю анонсы окна небольшого размера. При этом отправитель будет посылать получателю сегменты небольшого размера, которые должны поместиться в приемном окне. В результате передачи сегментов небольшого размера падает общая пропускная способность сети.

### **Сквозная передача (end-to-end transmission)**

Способ передачи данных, при котором взаимодействие происходит только между отправителем и конечным получателем сообщения, а не с промежуточными системами, как при маршрутизации пакетов. К системам сквозной передачи относится большинство протоколов уровня приложений и транспортные протоколы, наподобие TCP.

### **Служба, не требующая установки соединенияconnectionless service**

Одна из характеристик службы доставки пакетов, в которой каждый пакет или дейтаграмма рассматривается как отдельный самостоятельный объект. С ее помощью данные передаются получателю без предварительной установки соединения. Для идентификации в каждом пакете указывается адрес получателя. Службы, не требующие установки соединения с получателем, реализованы во многих сетевых технологиях и протоколах, таких как IP и UDP.

### **Служба, требующая установки соединения (connection-oriented service)**

Характеристика службы, реализованной на основе любой сетевой технологии, которая перед посылкой данных выполняет обмен служебной информацией для установки соединения. Подобный тип службы обеспечивает протокол TCP на программном уровне и коммутатор ATM — на аппаратном.

### **Смешанный ARP (promiscuous ARP)**

См. *агент ARP*.

### **Соединение (connection)**

Абстрактное понятие, использующееся в программах поддержки протокола. Например, в протоколе TCP предусмотрены средства установки соединения между взаимодействующими прикладными программами, которые работают на разных компьютерах, подключенных к сети.

### **Сообщество Internet (Internet Society, или ISOC)**

Некоммерческая организация, созданная с целью повышения заинтересованности пользователей к Internet. Сообщество Internet выступает в роли головной организации для IAB и продолжает служить делу объединения людей во всем мире посредством использования Internet

### **Соответствие наибольшему префиксу (longest-prefix matching)**

Методика, используемая IP-маршрутизаторами при выполнении поиска в таблице маршрутов. Среди найденных элементов, которые соответствуют адресу получателя, маршрутизатор выбирает тот элемент, у которого маска подсети содержит большее количество битов.

### **Специфичный маршрут к узлу сети (host-specific route)**

Запись в таблице маршрутизации, в которой вместо IP-адреса сети, подсети или стандартного маршрута указан IP-адрес конкретного узла сети.

### **Срочные данные (urgent data)**

Метод, используемый в протоколе TCP для отправки данных вне очереди. В отличие от обычных, срочные данные посылаются в сеть немедленно, а не помещаются в буфер на отправку.

### **Стандартный маршрут (default route)**

Один из элементов таблицы маршрутизации, относящийся ко всем получателям, маршруты к которым не заданы явно. Такой элемент присутствует практически во всех таблицах маршрутизации узлов сети и маршрутизаторов.

### **Стандартный порт (well-known port)**

Номер порта протокола, зарезервированный для использования конкретной сетевой службой, в которой используется один из протоколов транспортного уровня (т.е. TCP или UDP). Поскольку каждый из серверов ожидает поступления запроса по стандартному порту, клиентской программе достаточно знать только адрес компьютера, на котором запущена эта служба, чтобы вступить с ней во взаимодействие.

### **Стандартный порядок следования байтов (standard byte order)**

См. *сетевой порядок следования байтов*.

### **Степень взаимодействия (interoperability)**

Возможность кооперации разнородных систем при решении вычислительных задач. Имеется в виду возможность взаимодействия компьютерных систем от различных производителей. Этим термином как нельзя лучше выражается цель создания объединенной сети, а именно: определение абстрактной, независимой от оборудования сетевой среды, которая позволяет выполнять распределенные вычисления путем взаимодействия компьютеров по сети на транспортном уровне без использования низкоуровневых протоколов.

### **Технически обеспечиваемая услуга (provisioned service)**

Один из видов услуг, предоставляемый телефонной компанией, которая требует ручной настройки оборудования.

### **Тип следующего заголовка (Next Header)**

Поле заголовка дейтаграммы протокола IPv6, в котором указывается тип следующего заголовка (если он присутствует).

### **Точечная десятичная форма записи (dotted decimal notation)**

Синтаксическая форма, используемая для представления 32-разрядных двоичных целых чисел, состоящая из четырех 8-битовых чисел, представленных в десятичной системе счисления и разделенных точками. В большинстве прикладных программ, поддерживающих протокол TCP/IP, вместо имени сервера можно указывать его IP-адрес, представленный в точечной десятичной форме записи.

### **Точечная квадратичная форма записи (dotted quad notation)**

Синтаксическая форма, используемая для представления двоичных значений в виде 16-битовых шестнадцатеричных чисел, разделенных точкой.

### **Точечная шестнадцатеричная форма записи (dotted hex notation)**

Синтаксическая форма, используемая для представления двоичных значений в виде 8-битовых шестнадцатеричных чисел, разделенных точкой.

### **Точка воспроизведения (playback point)**

Минимальное количество данных в приемном буфере, необходимое для их плавного воспроизведения. Используется для подавления эффекта дрожания в сети при передаче аудио- и видеопотоков данных.

### **Точка подключения к подсети (SubNetwork Attachment Point)**

См. *SNAP*.

### **Трансивер (transceiver)**

Электронное устройство (приемопередатчик), с помощью которого сетевой интерфейс узла сети подключается к локальной сети, например Ethernet. Трансиверы Ethernet содержат специальные аналоговые электронные схемы, передающие сигнал в кабель и выполняющие фильтрацию коллизий.

### **Требования к маршрутизатору (router requirements)**

Документ, в котором описаны обновления семейства протоколов TCP/IP, используемые в маршрутизаторах. См. *требования к узлу сети*.

### **Требования к узлу сети (host requirements)**

Достаточно длинный документ, содержащий исправления и изменения, относящиеся к семейству протоколов TCP/IP. Этот документ опубликован среди RFC. См. *требования к маршрутизатору*.

### **Требования к шлюзу (gateway requirements)**

См. *требования к маршрутизатору*.

### **Трехэтапный метод квитирования (three-way handshake)**

Процесс обмена тремя TCP-сегментами, используемый для надежной установки соединения между получателями или корректного разрыва соединения.

### **Туннелирование (tunneling)**

Методика, при которой пакет, принадлежащий одному из протоколов высокого уровня, инкапсулируется в дейтаграмму транспортного протокола и передается по сети. Например, при создании магистрали многоадресатной передачи данных MBONE используется туннелирование IP-в-IP. При этом многоадресатная IP-дейтаграмма инкапсулируется в обычную IP-дейтаграмму и передается от одного узла многоадресатной маршрутизации до другого. Туннелирование используется при создании VPN для передачи зашифрованных дейтаграмм между сетевыми центрами. См. также *IP-в-IP*.

### **Узел (host)**

Персональный компьютер конечного пользователя или любое периферийное устройство, подключенное к сети. Узлом сети могут быть такие периферийные устройства, такие как принтеры, небольшие портативные устройства и даже большие суперкомпьютеры. Ср. с *маршрутизатором*.

### **Узкополосная передача (baseband)**

Способ передачи данных в сетях наподобие Ethernet, который подразумевает использование одной несущей частоты. При этом все станции, подключенные к такой сети, фиксируют факт передачи каждого пакета данных. Ср. с *широкополосной передачей*.

### **Управление потоком (flow control)**

Изменение количества посылаемых в сеть пакетов, которое осуществляется узел сети или маршрутизатор с целью ликвидации возникшей перегрузки.

### **Уровень 1**

Название уровня взаимодействия, относящегося к сетевому интерфейсу. Термин унаследован из семиуровневой эталонной модели OSI. На уровне 1 описывается механизм физического соединения, включая распайку сетевого разъема и величину напряжения сигнала, передающегося по проводам.

### **Уровень 2**

В семиуровневой модели OSI данный уровень относится к каналу связи (т.е. описывается формат фреймов). В пятиуровневой модели протокола TCP/IP уровень 2 относится к физическому формату фрейма и системе адресации. Поэтому адрес второго уровня является MAC-адресом (т.е. физическим адресом сети Ethernet).

### **Уровень 3**

В семиуровневой модели OSI этот уровень относится к сети передачи данных. В пятиуровневой модели протокола TCP/IP уровень 3 относится к межсетевому взаимодействию (т.е. описывается протокол IP и формат IP-дейтаграмм). Поэтому IP-адрес относится к адресу третьего уровня.

### **Усечение хвоста очереди (tail drop)**

Простой алгоритм, применяемый маршрутизаторами при переполнении входной очереди. Суть его состоит в том, что при переполнении очереди маршрутизатор начинает аннулировать все входящие дейтаграммы. Данный алгоритм менее эффективен, чем RED.

### **Файловый сервер**

Специальная прикладная программа, запущенная в виде процесса на отдельном компьютере, с помощью которой другие прикладные программы, запущенные на удаленных компьютерах, получают доступ к файлам на текущем компьютере. Иногда этот термин ошибочно употребляют в отношении физического компьютера, на котором запущена программа доступа к файлам.

### **Физический адрес (physical address)**

Синоним *MAC-адреса* или *аппаратного адреса*.

### **Фильтр пакетов (packet filter)**

Специальное устройство в маршрутизаторе, которое можно сконфигурировать так, чтобы маршрутизатор отклонял некоторые типы пакетов. Фильтры пакетов часто используются для создания защищенных брандмауэров.

### **Фрагментация (fragmentation)**

Процесс разделения IP-дейтаграммы на куски меньшего размера, выполняемый в том случае, если исходная дейтаграмма не может быть непосредственно передана по сети. Каждый фрагмент имеет такой же формат, что и обычная дейтаграмма. Для управления процессом фрагментации в заголовке IP-дейтаграммы предусмотрены специальные поля. В них указывается признак фрагментации и смещение фрагмента относительно исходной дейтаграммы. Процесс восстановления исходной дейтаграммы выполняется модулем протокола IP машины конечного получателя.

### **Фрейм (frame)**

Формально, пакет, который передается по последовательному каналу связи. Этот термин унаследован из символьно ориентированных протоколов, в которых при передаче пакета добавлялись специальные символы, обозначающие начало и конец пакета. В этой книге данный термин используется для обозначения низкоуровневых объектов, передаваемых по физическим сетям.

### **Честная организация очередей (fair queueing)**

Широко используемая методика для устранения перегрузки маршрутизатора. Она названа “честной”, поскольку для доступа к маршрутизатору для каждого узла выделяется одинаковая ширина полосы пропускания. Следует отметить, что методика честной организации очередей не является полностью оптимальной, поскольку в ней не делаются различия между узлами сети с малым и большим трафиком, т.е. между теми узлами, к которым подключено несколько активных соединений, и крупными сетевыми центрами с большим количеством пользователей и активных соединений.

### **Число переходов (hop count)**

Один из методов измерения расстояния между двумя точками объединенной сети. Если значение числа переходов равно  $n$ , то это означает, что отправителя и конечного получателя разделяет  $n$  маршрутизаторов.

### **Широковещание и отсечение (broadcast and prune)**

Методика, используемая в управляемой данными многоадресатной маршрутизации. Она заключается в том, что маршрутизатор не пересыпает многоадресатные дейтаграммы в сеть, если ему достоверно известно, что она не содержит членов группы.

### **Широковещательная передача (broadcast)**

Один из механизмов доставки сообщений, когда копия одного пакета доставляется всем узлам сети. Режим широковещательной доставки может быть реализован как на аппаратном (например, в сети Ethernet), так и на программном уровне (например, широковещательная передача IP-пакета при наличии подсетей).

### **Широковещательный адрес Беркли (Berkeley broadcast address)**

Название нестандартного широковещательного IP-адреса, содержащего в поле идентификатора узла сети все нулевые биты (вместо всех единичных битов). Впервые эта технология широковещательной передачи была применена в системе BSD UNIX в университете в Беркли. Отсюда и происходит ее название.

### **Широкополосная передача (broadband)**

Один из способов передачи данных в высокоскоростных сетях. При этом выполняется мультиплексирование нескольких независимых потоков данных и их передача по одному физическому кабелю (обычно с помощью разделения несущих

сигналов по частоте). Например, по одному 50-мегабитовому широкополосному кабелю можно передать пять 10-мегабитовых потоков данных, каждый из которых можно рассматривать как независимую сеть Ethernet. Широкополосная передача позволяет уменьшить количество сетевого кабеля. К ее недостаткам можно отнести высокую удельную стоимость оборудования, необходимого для организации одного соединения. Ср. с *узкополосной передачей*.

#### **Шифрование методом открытого ключа (public key encryption)**

Методика шифрования, в которой используется два ключа: секретный и открытый. Открытый ключ используется для шифрования сообщений, отсылаемых получателю, для расшифровки которых применяется секретный ключ.

#### **Шлюз (gateway)**

Одно из устройств, позволяющее соединить между собой две или более разнотипных сетевых системы и преобразующее информационные потоки, передающиеся между ними. Первоначально создатели объединенной сети называли такие устройства *IP-шлюзами*. Это были специальные компьютеры, выполнявшие маршрутизацию IP-дейтаграмм. Однако позднее разработчики сетевого оборудования стали использовать другой термин — *IP-маршрутизаторы*.

#### **Шлюз уровня приложений (application gateway)**

Прикладная программа, соединяющая несколько разнородных систем и преобразующая передаваемые между ними данные. В качестве примера можно назвать шлюз электронной почты, который часто используется для обмена электронной почты между сетями с разными протоколами.

#### **Экспоненциальная задержка (exponential backoff)**

См. *двойная экспоненциальная задержка*.

#### **Эпохальная дата (epoch date)**

Историческая дата, выбранная в качестве точки отсчета при исчислении времени. В протоколе TCP/IP в качестве точки отсчета времени выбран момент наступления нулевой секунды 1 января 1900 года по всеобщему скоординированному времени (прежнее название — среднее время по Гринвичу). На запрос даты или времени программы протокола TCP/IP возвращают значение в виде количества секунд, прошедших с момента наступления эпохальной даты.

#### **Эталонная модель (reference model)**

Описание принципов разделения семейства протоколов на уровни и способов взаимодействия между уровнями. В протоколе TCP/IP используется пятиуровневая эталонная модель. Ранее использовалась семиуровневая модель взаимодействия открытых систем (OSI), принятая ISO.

#### **Эхо-запрос и эхо-ответ**

Тип сообщения, используемый для тестирования возможности сетевого взаимодействия. Эхо-запрос протокола ICMP и ответ на него используются в программе ping.

#### **Ядро сети (core architecture)**

Структура объединенной сети, состоящая из центральной системы маршрутизации, к которой подключены локальные системы маршрутизации. Первоначальная структура сети Internet состояла из одной магистральной сети, которая

использовалась в качестве ядра. По мере того как провайдеры Internet создавали новые магистральные сети, структура Internet изменялась и все дальше отходила от системы с одним ядром.

#### **Ячейка (cell)**

Небольшой пакет фиксированного размера. Постоянство размеров пакета позволяет оптимизировать алгоритмы коммутации на аппаратном уровне. Чаще всего ячейки используются в сетях ATM, где их общая длина составляет 53 октета (48 октетов данных и 5 октетов заголовка).

## *Список литературы*

1. ABRAMSON, N., The ALOHA System — Another Alternative for Computer Communications, *Proceedings of the Fall Joint Computer Conference*, 1970.
2. ABRAMSON, N., KUO, F. (EDS.), *Computer Communication Networks*, Prentice Hall, Englewood Cliffs, New Jersey, 1973.
3. ANDREWS, D. W., and SHULTZ G. D., A Token-Ring Architecture for Local Area Networks: An Update, *Proceedings of Fall 82 COMPCON*. IEEE, 1982.
4. ATALLAH M., and COMER D. E., Algorithms for Variable Length Subnet Address Assignment, *IEEE Transactions on Computers*, vol. 47:6, 693–699, June 1998.
5. BALL J.E., BURKE E.J., GERTNER I., LANTZ K.A., and RASHID R.F., Perspectives on Message-Based Distributed Computing, *IEEE Computing Networking Symposium*, 46–51, 1979.
6. BBN, A History of the ARPANET: The First Decade, *Technical Report* Bolt, Beranek, and Newman, Inc., 1981.
7. BBN, Specification for the Interconnection of a Host and an IMP (revised), *Technical Report 1822*, Bolt, Beranek, and Newman, Inc., December 1981.
8. BERTSEKAS, D., and GALLAGER, R., *Data Networks*, 2nd edition, Prentice-Hall, Upper Saddle River, New Jersey, 1991.
9. BIAGIONI, E., COOPER, E., and SANSOM, R., Designing a Practical ATM LAN, *IEEE Network*, 32–39, March 1993.
10. BIRRELL, A., and NELSON, B., Implementing Remote Procedure Calls, *ACM Transactions on Computer Systems*. 2(1), 39–59, February 1984.
11. BLACK, U., *ATM: Foundation for Broadband Networks*, Prentice-Hall, Upper Saddle River, New Jersey, 1995.
12. BOGGS, D., SHOCH, J., TAFT, E., and METCALFE, R., PUP: An Internetwork Architecture, *IEEE Transactions on Communications*., April 1980.
13. BORMAN, D., Implementing TCP/IP on a Cray Computer, *Computer Communication Review*, 19(2), 11–15, April 1989.
14. BROWNBRIDGE, D., MARSHALL, L., and RANDELL, B., The Newcastle Connections or UNIXes of the World Unite!, *Software — Practice and Experience*, 12(12), 1147–1162, December 1982.
15. CASNER, S., and DEERING, S., First IETF Internet Audiocast, *Computer Communications Review*, 22(3), 92–97, July 1992.
16. CERF, V., and CAIN, E., The DOD Internet Architecture Model, *Computer Networks*, October 1983.
17. CERF, V., and KAHN, R., A Protocol for Packet Network Interconnection, *IEEE Transactions of Communications*, Com-22(5), May 1974.
18. CERF, V., A History of the ARPANET, *ConneXions, The Interoperability Report*, 480 San Antonio Rd, Suite 100, Mountain View, California, October 1989.
19. CHERITON, D. R., Local Networking and Internetworking in the V-System, *Proceedings of the Eighth Data Communications Symposium*, 1983.
20. CHERITON, D., VMTP: A Transport Protocol for the Next Generation of Communication Systems, *Proceedings of ACM SIGCOMM '86*, 406–415, August 1986.
21. CHESSON, G., Protocol Engine Design, *Proceedings of the 1987 Summer USENIX Conference*, Phoenix, AZ, June 1987.

22. CHESWICK, W., and BELLOVIN, S., *Firewalls And Internet Security: Repelling the Wiley Hacker*, 2nd edition, Addison-Wesley, Reading, Massachusetts, 1998.
23. CLARK, D., The Structure of Systems Using Upcalls, *Proceeding of the Tenth ACM Symposium on Operation Systems Principles*, 171–180, 1985.
24. CLARK, D., and FANG, W., Explicit Allocation Of Best-Effort Packet Delivery Service, *IEEE/ACM Transactions On Networking*, 6(4), August 1998.
25. CLARK, D., LAMBERT, M., and ZHANG, L., NETBLT: A High Throughput Transport Protocol, *Proceedings of ACM SIGCOMM '87*, August 1987.
26. CLARK, D., JACOBSON, V., ROMKEY, J., and SALWEN H., An Analysis of TCP Processing Overhead, *IEEE Communications*, 23–29, June 1989.
27. COHEN, D., On Holy Wars and a Plea for Peace, *IEEE Computer*, 48–54., 1981.
28. COMER, D. E., *Computer Networks And Internets*, 2nd edition, Prentice-Hall, Upper Saddle River, New Jersey, 1999.
29. COMER, D. E., and KORB, J. T., CSNET Protocol Software: The IP-to-X25 Interface, *Computer Communications Review*, 13(2), 1983.
30. COMER, D. E., NARTEN, T., and YAVATKAR, R., The Cypress Network: A Low-Cost Internet Connection Technology, *Technical Report TR-653*, Purdue University, West Lafayette, IN, April 1987.
31. COMER, D. E., NARTEN, T., and YAVATKAR, R., The Cypress Coaxial Packet Switch, *Computer Networks and ISDN Systems*, vol. 14:2-5, 383–388, 1987.
32. COMER, D. E., and STEVENS, D. L., *Internetworking With TCP/IP, Volume II: Design, Implementation, and Internals*, 3rd edition, Prentice-Hall, Upper Saddle River, New Jersey, 1999.
33. COMER, D. E., and STEVENS, D. L., *Internetworking With TCP/IP, Volume III: Client-Server Programming And Applications, BSD socket version*, 2nd edition, Prentice-Hall, Upper Saddle River, New Jersey, 1996.
34. COMER, D. E., and STEVENS, D. L., *Internetworking With TCP/IP, Volume III: Client-Server Programming And Applications, AT&T TLI version*, Prentice-Hall, Upper Saddle River, New Jersey, 1994.
35. COMER, D. E., and STEVENS, D. L., *Internetworking With TCP/IP, Volume III: Client-Server Programming And Applications, Windows Sockets version*, Prentice-Hall, Upper Saddle River, New Jersey, 1997.
36. COTTON, I., Technologies for Local Area Computer Networks, *Proceedings of the Local Area Communications Network Symposium*, 1979.
37. DALAL, Y. K., and PRINTIS, R. S., 48-Bit Absolute Internet and Ethernet Host Numbers, *Proceedings of the Seventh Data Communications Symposium*, 1981.
38. DEERING S. E., and CHERITON, D. R., Multicast Routing in Datagram Internetworks and Extended LANs, *ACM Transactions on Computer Systems*, 8(2), 85–110, May 1990.
39. DEERING, S., ESTRIN, D., FARINACCI, D., JACOBSON, V., LIU, C-G, and WEI, L., An Architecture for Wide-Area Multicasting Routing, *Proceedings of ACM SIGCOMM '94*, 126–135, August 1994.
40. DENNING P. J., *The Science of Computing: Worldnet*, in American Scientist, 432–434, September-October 1989.
41. DENNING P. J., *The Science of Computing: The ARPANET After Twenty Years*, in American Scientist, 530–534, November-December 1989.
42. DE PRYCKER, M., *Asynchronous Transfer Mode Solution for Broadband ISDN*, 3rd edition, Prentice-Hall, Upper Saddle River, New Jersey, 1995.

43. DIGITAL EQUIPMENT CORPORATION, INTEL CORPORATION, and XEROX CORPORATION, *The Ethernet: A Local Area Network Data Link Layer and Physical Layer Specification*, September 1980.
44. DION, J., The Cambridge File Server, *Operating Systems Review*, 14(4), 26–35, Oct. 1980.
45. DRIVER, H., HOPEWELL, H., and IAQUINTO, J., How the Gateway Regulates Information Control, *Data Communications*., September 1979.
46. EDGE, S. W., Comparison of the Hop-by-Hop and Endpoint Approaches to Network Interconnection, in *Flow Control in Computer Networks*, J-L. GRANGE and M. GIEN (EDS.), North-Holland, Amsterdam, 359–373, 1979.
47. EDGE, S., An Adaptive Timeout Algorithm for Retransmission Across a Packet Switching Network, *Proceedings of ACM SIGCOMM '83*, 1983.
48. ERIKSSON, H., MBONE: The Multicast Backbone, *Communications of the ACM*, 37(8), 54–60, August 1994.
49. FALK, G., The Structure and Function of Network Protocols, in *Computer Communications, Volume I: Principles*, CHOU, W. (ED.), Prentice-Hall, Upper Saddle River, New Jersey., 1983.
50. FARMER, W. D., and NEWHALL, E. E., An Experimental Distributed Switching System to Handle Bursty Computer Traffic, *Proceedings of the ACM Symposium on Probabilistic Optimization of Data Communication Systems*, 1–33, 1969.
51. FEDOR, M., GATED: A Multi-Routing Protocol Daemon for UNIX, *Proceedings of the 1988 Summer USENIX conference*, San Francisco, California, June 1988.
52. FEINLER, J., JACOBSEN, O. J., and STAHL M., *DDN Protocol Handbook Volume Two, DARPA Internet Protocols*, DDN Network Information Center, SRI International, 333 Ravenswood Avenue, Room EJ291, Menlo Park, California, December 1985.
53. FLOYD, S., and JACOBSON, V., Random Early Detection Gateways for Congestion Avoidance, *IEEE/ACM Transactions on Networking*, 1(4), August 1993.
54. FRANK, H., and CHOU, W., Routing in Computer Networks, *Networks*, 1(1), 99–112, 1971.
55. FULTZ, G. L., and KLEINROCK, L., Adaptive Routing Techniques for Store-and-Forward Computer Communication Networks, presented at *IEEE International Conference on Communications*, Montreal, Canada, June 14–16, 1971.
56. GERLA, M., and KLEINROCK, L., Flow Control: A Comparative Survey, *IEEE Transactions on Communications*, April 1980.
57. HINDEN, R., HAVERTY, J., and SHELTZER A., The DARPA Internet: Interconnecting Heterogeneous Computer Networks with Gateways, *Computer*, September 1983.
58. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, Information processing systems — Open Systems Interconnection — *Transport Service Definition*, International Standard number 8072, ISO, Switzerland, June 1986.
59. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, Information processing systems — Open Systems Interconnection — *Connection Oriented Transport Protocol Specification*, International Standard number 8073, ISO, Switzerland, July 1986.
60. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, Information processing systems — Open Systems Interconnection — *Specification of Basic*

- Specification of Abstract Syntax Notation One (ASN.1)*, International Standard number 8824, ISO, Switzerland, May 1987.
61. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, , Information processing systems — Open Systems Interconnection — *Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)*, International Standard number 8825, ISO, Switzerland, May 1987.
62. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, Information processing systems — Open Systems Interconnection — *Management Information Service Definition, Part 2: Common Management Information Service*, Draft International Standard number 9595-2, ISO, Switzerland, May 1988.
63. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, Information processing systems — Open Systems Interconnection — *Management Information Protocol Definition, Part 2: Common Management Information Protocol*, Draft International Standard number 9596-2, May 1988.
64. JACOBSON, V., Congestion Avoidance and Control, *Proceedings ACM SIGCOMM '88*, August 1988.
65. JAIN, R., On Caching Out-of-Order Packets in Window Flow Controlled Networks, *Technical Report*, DEC-TR-342, Digital Equipment Corporation, January 1985.
66. JAIN, R., Divergence of Timeout Algorithms for Packet Retransmissions, *Proceedings Fifth Annual International Phoenix Conference on Computers and Communications*, Scottsdale, AZ, March 1986.
67. JAIN, R., A Timeout-Based Congestion Control Scheme for Window Flow-Controlled Networks, *IEEE Journal on Selected Areas in Communications*, Vol. SAC-4, no. 7., October 1986.
68. JAIN, R., RAMAKRISHNAN, K., and CHIU, D-M., Congestion Avoidance in Computer Networks With a Connectionless Network Layer. *Technical Report*, DEC-TR-506, Digital Equipment Corporation, August 1987.
69. JAIN, R., *The Art of Computer Systems Performance Analysis*, John Wiley & Sons, New York, 1991.
70. JAIN, R., Myths About Congestion Management in High-speed Networks, *Internetworking: Research and Experience*, 3(3), 101–113, May 1992.
71. JAIN, R., *FDDI Handbook; High-Speed Networking Using Fiber and Other Media*, Addison Wesley, Reading, Massachusetts, 1994.
72. JENNINGS, D. M., LANDWEBER, L. H., and FUCHS, I. H., Computer Networking for Scientists and Engineers, *Science* vol. 231, 941–950, February 28, 1986.
73. JUBIN, J., and TORNOW, J., The DARPA Packet Radio Network Protocols, *IEEE Proceedings*, January 1987.
74. KAHN, R., Resource-Sharing Computer Communications Networks, *Proceedings of the IEEE*, 60(11), 1397–1407, November 1972.
75. KARN, P., PRICE, H., and DIERSING, R., Packet Radio in the Amateur Service, *IEEE Journal on Selected Areas in Communications*, May 1985.
76. KARN, P., and PARTRIDGE, C., Improving Round-Trip Time Estimates in Reliable Transport Protocols, *Proceedings of ACM SIGCOMM '87*, August 1987.
77. KAUFMAN, C., PERLMAN, R., and SPECINER, M., *Network Security: Private Communication in a Public World*, Prentice-Hall, Upper Saddle River, New Jersey, 1995.
78. KENT, C., and MOGUL, J., Fragmentation Considered Harmful, *Proceedings of ACM SIGCOMM '87*, August 1987.

79. КНУТ, Дональд, Эрвин, *Искусство программирования в 3-х томах*, Издательский дом “Вильямс”, 2000.
80. LAMPSON, B. W., PAUL, M., and SIEGERT, H. J., (EDS.), *Distributed Systems — Architecture and Implementation (An Advanced Course)*, Springer-Verlag, Berlin, 1981.
81. LAMPSON, B. W., SRINIVASAN, V., and VARGHESE, G., IP Lookups Using Multiway and Multicolumn Search, *IEEE/ACM Transactions on Networking*, vol. 7, 324–334, June 1999.
82. LANZILLO, A. L., and PARTRIDGE, C., Implementation of Dial-up IP for UNIX Systems, *Proceedings 1989 Winter USENIX Technical Conference*, San Diego, CA, January 1989.
83. LEFFLER, S., McKUSICK, M., KARELS, M., and QUARTERMAN, J., *The Design and Implementation of the 4.4BSD UNIX Operating System*, Addison Wesley, Reading, Massachusetts, 1996.
84. McNAMARA, J., *Technical Aspects of Data Communications*, 2nd edition. Digital Press, Digital Equipment Corporation, Bedford, Massachusetts, 1998.
85. McQUILLAN, J. M., RICHER, I., and ROSEN, E., The New Routing Algorithm for the ARPANET, *IEEE Transactions on Communications*, (COM-28), 711–719, May 1980.
86. METCALF, R. M., and BOGGS D. R., Ethernet: Distributed Packet Switching for Local Computer Networks, *Communications of the ACM*, 19(7), 395–404, July 1976.
87. MILLER, C. K., THOMPSON, D. M., Making a Case for Token Passing in Local Networks, *Data Communications*, March 1982.
88. MILLS, D., and BRAUN, H-W., The NSFNET Backbone Network, *Proceedings of ACM SIGCOMM '87*, August 1987.
89. MORRIS, R., Fixing Timeout Intervals for Lost Packet Detection in Computer Communication Networks, *Proceedings AFIPS National Computer Conference*, AFIPS Press, Montvale, New Jersey, 1979.
90. NAGLE, J., On Packet Switches With Infinite Storage, *IEEE Transactions on Communications*, Vol. COM-35:4, April 1987.
91. NARTEN, T., Internet Routing, *Proceedings ACM SIGCOMM '89*, September 1989.
92. NEEDHAM, R. M., System Aspects of the Cambridge Ring, *Proceedings of the ACM Seventh Symposium on Operating System Principles*, 82–85, 1979.
93. NELSON, J., 802: A Progress Report, *Datamation*, September 1983
94. NEWMAN, P., MINSHALL, G., and LYON, T. L., IP Switching — ATM Under IP, *IEEE Transactions on Networking*, Vol. 6:2, 117–129, April 1998.
95. OPPEN, D., and DALAL, Y., The Clearinghouse: A Decentralized Agent for Locating Named Objects, Office Products Division, XEROX Corporation, October 1981.
96. PARTRIDGE, C., Mail Routing Using Domain Names: An Informal Tour, *Proceedings of the 1986 Summer USENIX Conference*, Atlanta, GA, June 1986.
97. PARTRIDGE, C., Implementing the Reliable Data Protocol (RDP), *Proceedings of the 1987 Summer USENIX Conference*, Phoenix, Arizona, June 1987.
98. PARTRIDGE, C., *Gigabit Networking*, Addison-Wesley, Reading, Massachusetts, 1994.
99. PELTON, J., *Wireless and Satellite Telecommunications*, Prentice-Hall, Upper Saddle River, New Jersey, 1995.
100. PERLMAN, R., *Interconnections: Bridges and Routers*, 2nd edition, Addison-Wesley, Reading, Massachusetts, 2000.

101. PETERSON, L., *Defining and Naming the Fundamental Objects in a Distributed Message System*, Ph.D. Dissertation, Purdue University, West Lafayette, Indiana, 1985.
102. PETERSON, L., and VIE, B. DA, *Computer Networks: A Systems Approach*, 2nd edition, Morgan Kaufmann, San Francisco, CA, 1999.
103. PIERCE, J. R., Networks for Block Switching of Data, *Bell System Technical Journal*, 51, 1972.
104. POSTEL, J. B., Internetwork Protocol Approaches, *IEEE Transactions on Communications*, COM-28, 604–611, April 1980.
105. POSTEL, J. B., SUNSHINE, C. A., and CHEN D., The ARPA Internet Protocol, *Computer Networks*, 1981.
106. QUARTERMAN, J. S., and HOSKINS, J. C., Notable Computer Networks, *Communications of the ACM*, 29(10), October 1986.
107. RAMAKRISHNAN, K. and JAIN, R., A Binary Feedback Scheme For Congestion Avoidance In Computer Networks, *ACM Transactions on Computer Systems*, 8(2), 158–181, May 1990.
108. REYNOLDS, J., POSTEL, J., KATZ, A. R., FINN, G. G., and DESCHON, A. L., The DARPA Experimental Multimedia Mail System, *IEEE Computer*, October 1985.
109. RITCHIE, D. M., A Stream Input-Output System, *AT&T Bell Laboratories Technical Journal*, 63(8), 1987–1910, October 1984.
110. RITCHIE, D. M., and THOMPSON, K., The UNIX Time-Sharing System, *Communications of the ACM*, 17(7), 365–375, July 1974; revised and reprinted in *Bell System Technical Journal*, 57(6), 1905–1929, July-August 1978.
111. ROSE, M., *The Internet Message: Closing The Book with Electronic Mail*, Prentice-Hall, Upper Saddle River, New Jersey, 1993.
112. ROSENTHAL, R. (ED.), *The Selection of Local Area Computer Networks*, National Bureau of Standards Special Publication 500-96, November 1982.
113. SALTZER, J., Naming and Binding of Objects, *Operating Systems, An Advanced Course*, Springer-Verlag, 99–208, 1978.
114. SALTZER, J., Naming and Binding of Network Destinations, *International Symposium on Local Computer Networks*, IFIP/T.C.6, 311–317, April 1982.
115. SALTZER, J., REED, D., and CLARK, D., End-to-End Arguments in System Design, *ACM Transactions on Computer Systems*, 2(4), 277–288, November 1984.
116. SCHWARTZ, M., and T. STERN, IEEE Transactions on Communications, COM-28(4), 539–552., April 1980.
117. SHOCH, J. F., Internetwork Naming, Addressing, and Routing, *Proceedings of COMPCON*, 1978.
118. SHOCH, J. F., DALAL, Y., and REDELL, D., Evolution of the Ethernet Local Computer Network, *Computer*, August 1982.
119. SOLOMON, J., *Mobile IP: The Internet Unplugged*, Prentice-Hall, Upper Saddle River, New Jersey, 1997.
120. SOLOMON, M., LANDWEBER, L., and NEUHEGEN, D., The CSNET Name Server, *Computer Networks* (6), 161–172, 1982.
121. SRINIVASAN, V., and VARGHESE, G., Fast Address Lookups Using Controlled Prefix Expansion, *ACM Transactions on Computer Systems*, vol. 17, 1–40, February 1999.
122. STALLINGS, W., *Data and Computer Communications*, Macmillan Publishing Company, New York, 1985.
123. STALLINGS, W., *Local and Metropolitan Area Networks*, Prentice-Hall, Upper Saddle River, New Jersey, 1997.

124. STALLINGS, W., *High-Speed Networks: TCP/IP and ATM Design Principles*, Prentice-Hall, Upper Saddle River, New Jersey, 1998.
125. STEVENS, W. R., *UNIX Network Programming*, 2nd edition, Prentice-Hall, Upper Saddle River, New Jersey, 1998.
126. SWINEHART, D., McDANIEL, G., and BOGGS, D. R., WFS: A Simple Shared File System for a Distributed Environment, *Proceedings of the Seventh Symposium on Operating System Principles*, 9–17, December 1979.
127. TANENBAUM, A., *Computer Networks: Toward Distributed Processing Systems*, Prentice-Hall, Englewood Cliffs, New Jersey, 1981
128. TICHY, W., and RUAN, Z., Towards a Distributed File System, *Proceedings of Summer 84 USENIX Conference*, Salt Lake City, Utah, 87–97, June 1984.
129. TOMLINSON, R. S., Selecting Sequence Numbers, *Proceedings ACM SIGOPS/SIGCOMM Interprocess Communication Workshop*, 11–23, 1975.
130. WATSON, R., Timer-Based Mechanisms in Reliable Transport Protocol Connection Management, *Computer Networks*, North-Holland Publishing Company, 1981.
131. WEINBERGER, P. J., The UNIX Eighth Edition Network File System, *Proceedings 1985 ACM Computer Science Conference*, 299–301, 1985.
132. WELCH, B., and OSTERHAUT, J., Prefix Tables: A Simple Mechanism for Locating Files in a Distributed System, *Proceedings IEEE Sixth International Conference on Distributed Computing Systems*, 1845–189, May 1986.
133. WILKES, M. V., and WHEELER, D. J., The Cambridge Digital Communication Ring, *Proceedings Local Area Computer Network Symposium*, May 1979.
134. XEROX, Internet Transport Protocols, *Report XSIS 028112*, Xerox Corporation, Office Products Division, Network Systems Administration Office, 3333 Coyote Hill Road, Palo Alto, California, 1981.
135. ZHANG, L., Why TCP Timers Don't Work Well, *Proceedings of ACM SIGCOMM '86*, August 1986.