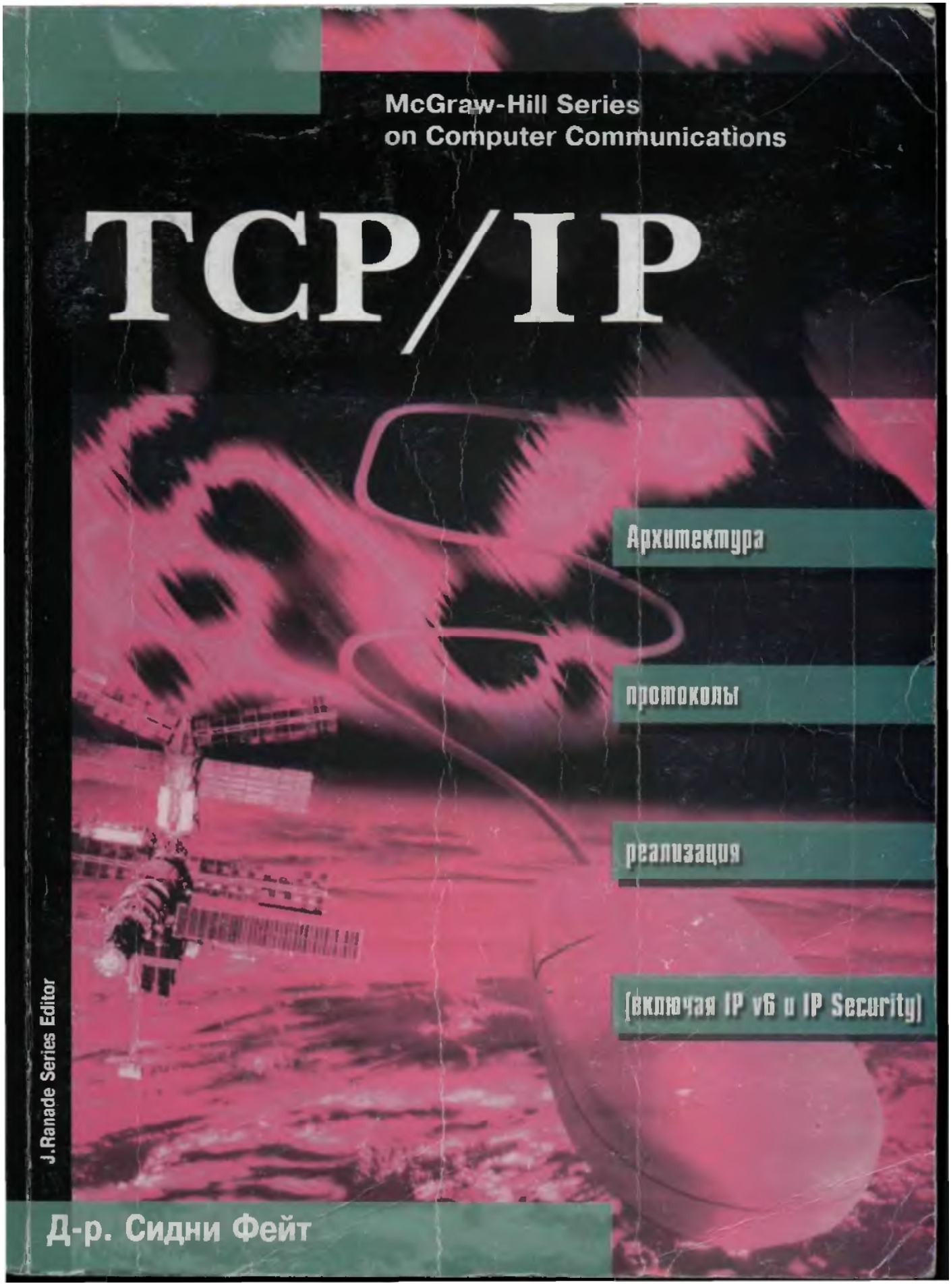
McGraw-Hill Series
on Computer Communications
TCP/IP
Архитектура
протоколы
реализация
(включая IP v6 и IP Security)
J.Ranade Series Editor
Д-р. Сидни Фейт

TCP/IP

ARCHITECTURE, PROTOCOLS, AND IMPLEMENTATION
WITH IPV6 AND IP SECURITY

Dr. Sidnie Feit

Second Edition

McGraw-Hill

TCP/IP

АРХИТЕКТУРА, ПРОТОКОЛЫ, РЕАЛИЗАЦИЯ
(ВКЛЮЧАЯ IP ВЕРСИИ 6 И IP SECURITY)

Д-р. Сидни Фейт

Второе издание



TCP/IP: Architecture, Protocols, and Implementation with IPv6 and IP Security – 2nd ed.
Dr. Sidnie Feit
Copyright 1997, 1993
All rights reserved

TCP/IP: Архитектура, протоколы, реализация (включая IP версии 6 и IP Security) – 2-е изд.
Д-р. Сидни Фейт

Переводчик М. Кузьмин
Научный редактор А. Головко
Корректоры Н. Веселкина, И. Красенкова, М. Ромашова
Верстка Т. Кирпичевой

Copyright © 1997, 1993 by The McGraw-Hill Companies, Inc.
ISBN 0-07-021389-5

McGraw-Hill

A Division of The McGraw-Hill Companies

© Издательство "Лори", 2000

Изд. № : OAI (03)
ЛР № : 070612 30.09.97 г.
ISBN 5-85582-072-6

Подписано в печать 24.05.2000 Формат 84 ×108/16
Бумага офсет № 1 Гарнитура Ньютон Печать офсетная
Печ. л. 28 Тираж 3200 Заказ № 350
Цена договорная

Издательство "Лори". Москва, ул. А. Живова, д. 10, стр. 1
Телефон для оптовых покупателей: (095)256-02-83
Размещение рекламы: (095)259-01-62

WWW.LORY–PRESS.RU

Отпечатано в типографии ИПО Профиздат.
109044, Москва, ул. Крутицкий вал, д. 18

Посвящается моему мужу и другу Вальтеру.

Предисловие

Эта книга служит введением в протоколы TCP/IP и содержит все необходимые сведения как для начинающих, так и для опытных пользователей. Она является практическим руководством по использованию TCP/IP и включает подробное описание применения этих протоколов в реальных сетях: как связать между собой локальные и глобальные сети, как выбрать имена систем, где получить сетевые адреса и как использовать существующие программные продукты для TCP/IP.

Книга поможет изучить работу протоколов TCP/IP разработчикам, системным и сетевым администраторам, программистам, обслуживающему персоналу или конечным пользователям, которые хотят познакомиться с работой своего сетевого окружения.

В книге описана терминология, концепции и механизмы TCP/IP. Рассматриваются стандарты, определяющие работу этих протоколов, а также методы работы с приложениями TCP/IP: каким образом выполняются фоновые процессы и как диагностировать состояние сетевых ресурсов. Для тех, кому это необходимо, приведено детальное (на уровне бит и байт) описание структур сетевых сообщений и информационных потоков при обмене этими сообщениями. Обсуждается работа программного интерфейса socket и приводятся примеры клиентских и серверных программ.

В данном издании существенно расширен объем материала. Целая глава посвящена наиболее распространенным коммуникационным протоколам. Отдельные главы описывают систему именования доменов, Word Wide Web, сетевые новости и приложения Gopher. Приведены сведения о следующем поколении протокола IP (версия 6). Средства безопасности рассматриваются на протяжении всей книги, но есть и отдельная глава по этой теме.

Сидни М. Фейт (Sidnie M. Feit)

Благодарности

Хочется выразить признательность факультету математики Йельского университета за предоставленную возможность посещения этого учебного заведения во время создания первой редакции книги. Работа на различных компьютерах университетской сети, а также ознакомление с предоставленной технической документацией оказали существенную помощь в написании книги. Руководитель компьютерного центра университета Г. Морроу Лонг познакомил меня с проблемами безопасности и способами их решения, с новым интересным программным обеспечением и важными тенденциями в развитии Интернета. Морроу был бесценным источником фактов о реальных реализациях сетевых систем.

Грэхем Ярброу прочитал рукопись книги от начала и до конца, предоставив ценные суждения о применении компьютеров и сетей на практике. Майкл МакФарланд прокомментировал несколько глав и сделал очень важные замечания.

Редактор этой серии книг, Джей Ренейд, всегда оказывал необходимую помощь в работе над книгой.

Компания Netmanage, Inc. предоставила последнюю версию Chameleon NFS (включающую *Personal Web Server*) и программное обеспечение сетевого мониторинга NewtWatch. Сценарии сетевого управления выполнялись с помощью *HP Open View for Windows Workgroup Node Manager*. Компания Network General предоставила монитор *Sniffer*, подробную документацию, советы и много мегабайт отчетов отслеживания работы протоколов. Компания Ashmount Research Ltd. предоставила программу для Windows под названием *NSLookup*.

Многие другие производители предложили свои продукты и техническую информацию. Компания FTP Software, Inc. поделилась своими программными продуктами для Windows и технической документацией к ним. Производители, и среди прочих Cisco Systems и Bay Systems, тщательно и подробно отвечали на все возникшие при работе над книгой вопросы о производимых ими продуктах.

Предисловие к русскому изданию

По тематике протоколов TCP/IP существует достаточно обширная литература, в том числе на русском языке (например, издательство "Лори" выпустило книгу Тодда Леммла, Моники Леммл и Джеймса Челлиса "TCP/IP. Учебное руководство для специалистов MCSE"). Однако эти книги в основном имеют инженерную направленность. Две-три первые главы в них посвящены теоретическим вопросам TCP/IP, а далее идет описание конкретных реализаций на уровне устройств и программных продуктов. Прогресс в области компьютерной техники ведет к тому, что приводимые в этих изданиях сведения быстро устаревают.

Предлагаемая русскому читателю книга имеет более теоретическую направленность, подробно и досконально описывая все связанные с TCP/IP спецификации и стандарты. Если в основном устройства или программные продукты устаревают примерно в течение года, то некоторые спецификации TCP/IP используются без изменений уже в течение десятка лет. Именно знакомство с теоретическими основами поможет глубже понять работу и функции различных протоколов, в том числе и реализованных в конкретных устройствах или программах.

Данную книгу можно с полным правом назвать энциклопедией, поскольку она содержит подробное обсуждение не только вопросов, связанных с TCP/IP, но и рассматривает смежную с ними тематику. Приведены сведения о протоколах, используемых в настоящее время, а также данные о протоколах и спецификациях, находящихся на стадии разработки и внедрения. Книга разделена на компактные части с пронумерованными заголовками. Для удобства читателей в конце тома приведены расшифровки всех аббревиатур и глоссарий. Надеемся, что данное издание послужит прекрасным справочником для многих специалистов по компьютерным сетям.

Хотя читателю более известны понятия из теории операционных систем, связанные с MS DOS или Windows, в данной книге в основном рассматриваются примеры из Unix. Мы не стали комментировать специфику стандартных средств этой операционной системы. Заметим только, что системные команды часто именуются программами, поскольку каждая из них реализуется отдельной программой. За более подробными сведениями можно обратиться к различным книгам издательства "Лори" по тематике Unix, в частности к прекрасной книге Кеннета Розена, Ричарда Розински, Джеймса Фарбра и Дугласа Хорста "Unix System V Release 4".

М. Кузьмин
Издательство "Лори"
Москва

Содержание

Предисловие . . . . . *vi*
Благодарности . . . . . *vii*
Предисловие к русскому изданию . . . . . *viii*

Введение **1**
I.1 Основы 1
I.2 Приложения TCP/IP 1
I.3 Терминология 2
I.3.1 Протоколы, элементы, стеки и наборы 2
I.3.2 Хосты 2
I.3.3 Маршрутизаторы 3
I.3.4 Интернет 3
I.3.5 Сетевой узел, система и элемент сети 3
I.3.6 ЛС, региональные сети и связи 3
I.3.7 Люди 3
I.3.8 Байты и октеты 3
I.3.9 Стиль "тупоконечников" и "остроконечников" 3
I.4 Реализации с использованием оборудования различных производителей. 4
I.5 Диалоги 4
I.6 Дополнительная литература 4

Глава 1 TCP/IP: что это такое и откуда взялось **5**
1.1 Введение 5
1.1.1 Зарождение TCP/IP 5
1.2 Принятие новых протоколов 6
1.3 Характеристики TCP/IP 6
1.3.1 Доступность TCP/IP 6
1.4 Интернет 7
1.5 INTERNIC 7
1.6 IAB, IETF и IESG 7
1.6.1 Рабочие группы и разработка протоколов 8
1.6.2 Другие источники протоколов Интернета 8
1.7 Requests For Comments 8
1.7.1 Состояние и статус стандартов 9
1.7.2 Присвоенные номера 9
1.7.3 RFC и стимулирование сетевого взаимодействия продуктов различных производителей 9
1.7.4 Связанные документы 10
1.8 Другие информационные ресурсы 10
1.9 Open System Interconnection 10

Глава 2 Обзор служб набора протоколов TCP/IP 11
2.1 Введение . . . 11
2.2 Коммуникации между приложениями . . . 11
2.2.1 Коммуникации с созданием соединений (TCP) . . . 11
2.2.2 Коммуникации без создания соединений (UDP) . . . 11
2.2.3 Интерфейс программирования socket . . . 11
2.2.4 Программный интерфейс RPC . . . 12
2.3 Основные службы . . . 12
2.3.1 Пересылка файлов . . . 12
2.3.2 Доступ с терминала . . . 12
2.3.3 Электронная почта . . . 12
2.3.4 Служба WWW . . . 13
2.4 Дополнительные службы . . . 13
2.4.1 Доступ к файлам . . . 13
2.4.2 Новости . . . 13
2.4.3 Служба имен DNS . . . 13
2.4.4 Коммерческое программное обеспечение . . . 14
2.4.5 Управление сетью . . . 14
2.4.6 Диалоги . . . 14
2.4.7 Диалог доступа с терминала . . . 14
2.4.8 Просмотр имен в базе данных DNS . . . 14
2.4.9 Диалог при пересылке файла . . . 15
2.4.10 WWW . . . 16
2.4.11 Новости . . . 17
2.4.12 Диалог для доступа к файлу . . . 17

Глава 3 Архитектура TCP/IP 18
3.1 Введение . . . 18
3.2 Деление на уровни . . . 18
3.2.1 Физический уровень . . . 19
3.2.2 Уровень связи данных . . . 19
3.2.3 Сетевой уровень . . . 19
3.2.4 Транспортный уровень (TCP) . . . 20
3.2.5 Транспортный уровень (UDP) . . . 20
3.2.6 Службы для приложений . . . 20
3.2.7 Пакетирование данных . . . 21
3.3 Обзор протоколов . . . 21
3.4 Маршрутизаторы и топология сети . . . 21
3.5 Маршрутизация в IP . . . 22
3.5.1 Протоколы маршрутизации . . . 23
3.6 Архитектура TCP . . . 23
3.7 Архитектура UDP . . . 24
3.8 Концепции безопасности . . . 24
3.8.1 Аутентификация . . . 24
3.8.2 Технология формирования резюме сообщения . . . 24
3.8.3 Целостность сообщения . . . 25
3.8.4 Конфиденциальность с помощью симметричного шифрования . . . 26
3.8.5 Асимметричный общедоступный ключ шифрования . . . 26
3.8.6 Комбинированное шифрование . . . 27

Глава 4 Технологии физического уровня и уровня связи данных 28
4.1 Введение . . . 28
4.2 Функции физического уровня, управление доступом к физическому носителю и уровень связи данных . . . 28
4.3 Сетевые технологии . . . 29
4.4 Извлечение данных из пакетов . . . 30
4.5 Протоколы связей "точка-точка" . . . 30

4.6 HDLC . . . 31
4.6.1 Формат кадра HDLC . . . 31
4.6.2 Недостатки HDLC . . . 32
4.7 Протокол PPP . . . 33
4.7.1 Сжатие в PPP . . . 33
4.8 Дополнительные возможности PPP . . . 34
4.8.1 Аутентификация . . . 34
4.8.2 Автоматическое отслеживание качества связи . . . 34
4.9 Протокол SLIP . . . 35
4.10 Локальные сети . . . 36
4.11 DIX Ethernet . . . 36
4.11.1 Носители для DIX Ethernet . . . 36
4.11.2 Протокол MAC для DIX Ethernet . . . 36
4.11.3 MAC-кадры DIX Ethernet . . . 37
4.12 Сети по спецификации 802 . . . 37
4.13 Заголовок LLC для 802.2 . . . 37
4.13.1 Спецификации 802.3 и 802.2 . . . 38
4.14 Уровни в сетях 802 . . . 39
4.15 Другие технологии локальных сетей . . . 39
4.15.1 Конфигурация и носители для Token-Ring . . . 39
4.15.2 MAC для 802.5 . . . 39
4.15.3 802.4 Token Bus . . . 39
4.15.4 FDDI . . . 39
4.16 Использование концентраторов . . . 40
4.17 Коммутация . . . 40
4.18 Широковещательные и многоадресные рассылки . . . 40
4.19 Сети с коммутацией пакетов . . . 41
4.20 Сети X.25 . . . 41
4.20.1 Уровни в X.25 . . . 42
4.20.2 X.25 и IP . . . 42
4.20.3 Многопротокольный режим поверх X.25 . . . 42
4.20.4 IP в отдельной виртуальной цепи X.25 . . . 43
4.20.5 Другие протоколы в отдельной виртуальной цепи X.25 . . . 43
4.20.6 Многопротокольный режим в виртуальной цепи . . . 43
4.20.7 Пакеты или PDU? . . . 43
4.21 Frame Relay . . . 43
4.22 SMDS . . . 44
4.22.1 IP поверх SMDS . . . 45
4.23 ATM . . . 45
4.24. Максимальное число пересылаемых элементов . . . 46
4.25 Создание туннелей . . . 47
4.26 Совместное использование сетевого интерфейса . . . 47
4.27 Замечания об уровне связи данных . . . 48
4.28 Завершающая часть кадра . . . 48
4.29 Рекомендуемая литература . . . 48

Глава 5 Именование и адресация 49

5.1 Введение . . . 49
5.2 Примеры имен Интернета . . . 49
5.3 Иерархическая структура имен . . . 50
5.4 Администрирование имен . . . 50
5.5 Формальная структура имен . . . 51
5.6 Всемирное дерево имен . . . 51
5.7 Конфигурирование имен систем . . . 52
5.8 Адреса . . . 53
5.9 Форматы адресов . . . 53
5.10 Классы адресов . . . 53

5.11 Адреса не подключенных к Интернету систем . . . 54
5.12 Примеры адресации . . . 55
5.12.1 Присваивание сети адресов класса A . . . 55
5.12.2 Присваивание сети адресов класса B . . . 55
5.12.3 Присваивание сетям адресов класса C . . . 55
5.13 Трансляция имен в адреса . . . 55
5.14 Псевдонимы имен . . . 56
5.15 Неэффективность классов адресов . . . 57
5.16 Сети и подсети TCP/IP . . . 57
5.17 Маска подсети . . . 58
5.18 Специальные зарезервированные адреса . . . 59
5.18.1 Идентификация сети и подсети . . . 59
5.18.2 Широковещательная рассылка в локальной подсети . . . 60
5.18.3 Широковещательные рассылки к подсети . . . 60
5.18.4 Широковещательные рассылки в сети . . . 60
5.18.5 Ограничения на IP-адрес . . . 60
5.18.6 Кольцевой адрес . . . 60
5.18.7 Заключение о зарезервированных специальных адресах . . . 62
5.19 Суперсети и CIDR . . . 62
5.20 Необходимость следующего поколения протокола IP . . . 63
5.21 IP-адреса, интерфейсы и множественное пребывание . . . 64
5.22 Конфигурирование адресов и масок подсети . . . 65
5.23 Взаимосвязь имен и адресов . . . 65
5.24 Протокол ARP . . . 65
5.24.1 Содержимое сообщения ARP . . . 66
5.24.2 Таблица ARP . . . 67
5.24.3 Обратные запросы ARP . . . 68
5.25 Множество адресов для одного интерфейса . . . 68
5.26 Прокси ARP . . . 69
5.27 Многоадресные рассылки . . . 69
5.27.1 Группы многоадресной рассылки . . . 70
5.27.2 Трансляция многоадресных рассылок в адреса Ethernet и FDDI . . . 70
5.27.3 Трансляция адресов многоадресных рассылок в адреса Token-Ring . . . 71
5.28 Дополнительная литература . . . 72

Глава 6 Протокол интернета 73

6.1 Введение . . . 73
6.2 Датаграммы IP . . . 73
6.3 Основные функции IP . . . 74
6.4 Использование маски подсети . . . 74
6.5 Хост в таблице маршрутизации IP . . . 75
6.6 Маршрутизация по следующему попаданию . . . 75
6.7 Еще один пример таблицы маршрутизации хоста . . . 76
6.8 Правило просмотра таблицы маршрутизации . . . 77
6.9 Таблицы маршрутизатора . . . 77
6.10 Таблица маршрутизации филиала компании . . . 78
6.11 Операции глобальной маршрутизации . . . 78
6.12 Возможности IP . . . 79
6.12.1 Адаптивная маршрутизация . . . 79
6.12.2 MTU, фрагментация и сборка . . . 80
6.13 Механизмы протокола IP . . . 80
6.13.1 Заголовок датаграммы . . . 80
6.13.2 Поля назначения, поле источника и поле протокола . . . 81
6.13.3 Версия, длина заголовка и длина датаграммы . . . 82
6.13.4 Приоритет и тип обслуживания . . . 82
6.13.5 Поле времени жизни . . . 84
6.13.6 Заголовок контрольной суммы . . . 84

6.14 Фрагментация . . . 84
6.14.1 Поле идентификации . . . 84
6.14.2 Поле флагов . . . 85
6.14.3 Поле смещения фрагмента . . . 85
6.14.4 Сборка фрагментированной датаграммы . . . 85
6.14.5 Тайм-аут сборки датаграммы . . . 86
6.14.6 Фрагментировать или не фрагментировать . . . 86
6.15 Просмотр статистики IP . . . 86
6.16 Варианты . . . 86
6.16.1 Маршрутизация от источника . . . 87
6.16.2 Обратный маршрут . . . 87
6.16.3 Описание маршрута . . . 88
6.16.4 Маршрутизация от источника и безопасность . . . 88
6.16.5 Запись пути . . . 88
6.16.6 Временная метка . . . 88
6.16.7 Базовая и улучшенная безопасность для Министерства обороны . . . 89
6.16.8 Конец списка вариантов и отсутствие операций . . . 89
6.16.9 Кодирование вариантов . . . 89
6.16.10 Кодирование Strict Source Route . . . 90
6.16.11 Кодирование Loose Source Route . . . 90
6.16.12 Кодирование Record Route . . . 90
6.16.13 Кодирование Timestamp . . . 90
6.16.14 Кодирование Basic и Extended Security Options . . . 91
6.17 Пример заголовка IP . . . 91
6.18 Сценарий обработки датаграммы . . . 92
6.18.1 Обработка в маршрутизаторе . . . 93
6.18.2 Обработка в хосте назначения . . . 93
6.19 Средства защиты и безопасность . . . 93
6.20 Замечания о производительности IP . . . 94
6.20.1 Полоса пропускания . . . 94
6.20.2 Использование буфера . . . 95
6.20.3 Ресурсы центрального процессора . . . 95
6.21 Дополнительные сведения о многоадресных рассылках . . . 95
6.22 Рекомендуемая литература . . . 96

Глава 7 Протокол ICMP 97

7.1 Введение . . . 97
7.2 Сообщения об ошибках ICMP . . . 97
7.2.1 Типы сообщений об ошибках . . . 98
7.2.2 Обязанность по отправке сообщения ICMP . . . 98
7.2.3 Входящие сообщения ICMP . . . 99
7.3 Когда не нужно посылать сообщение ICMP . . . 99
7.4 Формат сообщения ICMP . . . 99
7.4.1 Сообщение Destination Unreachable . . . 100
7.4.2 Сообщение Time Exceeded . . . 100
7.4.3 Сообщение Parameter Problem . . . 101
7.4.4 Проблемы перегрузок . . . 101
7.4.5 Сообщение Source Quench . . . 102
7.4.6 Сообщения Redirect . . . 102
7.4.7 Управление поступающими сообщениями ICMP . . . 103
7.5 Исследование MTU по пути . . . 103
7.6 Сообщения запросов ICMP . . . 104
7.6.1 Эхо-запросы и эхо-ответы . . . 105
7.6.2 Маска адреса . . . 106
7.6.3 Временная метка и ответ на Timestamp . . . 106
7.7 Просмотр действий в ICMP . . . 107
7.8 Обнаружение маршрутов . . . 108
7.8.1 Неисправные маршрутизаторы . . . 109
7.9 Дополнительная литература . . . 109

Глава 8 Маршрутизация в IP 110

8.1 Введение . . . 110
8.2 Автономные системы . . . 110
8.3 Маршрутизация в IP . . . 111
8.4 Метрики маршрутизации . . . 112
8.4.1 Протоколы вектора расстояния . . . 112
8.4.2 Протоколы по состоянию связи . . . 112
8.5 Таблицы маршрутизации . . . 112
8.6 Таблица маршрутизации по протоколу RIP . . . 113
8.6.1 Использование маски маршрута . . . 114
8.6.2 Маршрут по умолчанию . . . 114
8.6.3 Использование подсети 0 . . . 114
8.6.4 Прямые и косвенные назначения . . . 114
8.6.5 Метрики маршрутизации . . . 114
8.6.6 Возраст маршрута . . . 114
8.7 Таблица маршрутизации IGRP/BGP . . . 114
8.7.1 Использование маски маршрута . . . 116
8.7.2 Маршрут по умолчанию . . . 116
8.7.3 Прямые и косвенные точки назначения . . . 116
8.7.4 Малые подсети . . . 116
8.7.5 Строки для протокола Border Gateway Protocol . . . 116
8.7.6 Метрики маршрутизации . . . 116
8.7.7 Возраст маршрута . . . 116
8.8 Протоколы обслуживания таблиц маршрутизации . . . 116
8.9 Протокол RIP . . . 117
8.9.1 Инициализация RIP . . . 119
8.9.2 Обновление таблиц RIP . . . 119
8.9.3 Механизм RIP версии 1 . . . 119
8.9.4 Сообщения об изменениях в RIP версии 1 . . . 120
8.9.5 Поля сообщения об изменениях в RIP версии 1 . . . 120
8.9.6 Настройка RIP . . . 121
8.9.7 Триггерные изменения и хранение . . . 121
8.9.8 Деление горизонта и опасный реверс . . . 121
8.9.9 Нет маски подсети . . . 122
8.9.10 Широковещательные рассылки в локальной сети . . . 123
8.9.11 Отсутствие аутентификации . . . 123
8.9.12 Отсутствие распознавания медленных и быстрых связей . . . 123
8.9.13 Избыточный трафик . . . 123
8.10 Протокол RIP версии 2 . . . 124
8.10.1 Аутентификация в RIP версии 2 . . . 125
8.11 Переход на более интеллектуальные протоколы . . . 125
8.12 Протоколы IGRP и EIGRP . . . 125
8.12.1 Маршрутизация в IGRP . . . 125
8.12.2 Другие конфигурируемые значения IGRP . . . 126
8.12.3 Механизм протокола IGRP . . . 127
8.12.4 Внешняя маршрутизация . . . 127
8.12.5 Возможности EIGRP . . . 127
8.12.6 DUAL в EIGRP . . . 127
8.12.7 Таблицы топологии в DUAL . . . 128
8.12.8 Пригодный преемник в DUAL . . . 128
8.13 Протокол OSPF . . . 129
8.13.1 Автономные системы, области и сети . . . 129
8.13.2 Маршрутизация в области OSPF . . . 130
8.13.3 Кратчайшие пути для области OSPF . . . 130
8.13.4 Магистрали, грани и границы OSPF . . . 131
8.13.5 Маршрутизация через грань области OSPF . . . 132
8.13.6 Использование итоговой информации внутри области OSPF . . . 132
8.13.7 Точка назначения вне автономной области OSPF . . . 133
8.13.8 Протокол OSPF . . . 133
8.13.9 Сообщения Hello . . . 133

8.13.10 Назначенный маршрутизатор . . . 134
8.13.11 Смежность маршрутизаторов . . . 134
8.13.12 Инициализация базы данных маршрутизации . . . 134
8.13.13 Типы сообщений в OSPF . . . 135
8.13.14 Сообщения OSPF . . . 135
8.13.15 Содержание сообщения Link State Update протокола OSPF . . . 136
8.13.16 Улучшения в OSPF . . . 136
8.14 Маршрутизация в OSI . . . 136
8.15 Протоколы EGP . . . 137
8.16 EGP . . . 137
8.16.1 Модель EGP . . . 137
8.17 Протокол BGP . . . 139
8.17.1 Объединение маршрутов в BGP . . . 140
8.17.2 Механизмы BGP . . . 140
8.17.3 Содержание сообщения об изменениях в BGP . . . 140
8.17.4 Проблема выбора варианта . . . 141
8.17.5 Применение объединения маршрутов . . . 142
8.17.6 Изолированные маршруты BGP . . . 142
8.18 Дополнительная литература . . . 142

Глава 9 Протокол UDP 144

9.1 Введение . . . 144
9.1.1 Широковещательные и многоадресные рассылки . . . 144
9.2 Порты приложений . . . 145
9.3 Адреса socket . . . 147
9.4 Механизмы протокола UDP . . . 147
9.4.1 Заголовок UDP . . . 147
9.4.2 Контрольная сумма . . . 148
9.4.3 Другие функции UDP . . . 148
9.4.4 Пример сообщения UDP . . . 149
9.5 Нагрузки в UDP . . . 149
9.6 Дополнительная литература . . . 149

Глава 10 Протокол TCP 150

10.1 Введение . . . 150
10.1.1 Основные службы TCP . . . 150
10.1.2 TCP и модель клиент/сервер . . . 150
10.2 Концепции TCP . . . 151
10.2.1 Входной и выходной потоки данных . . . 151
10.2.2 Сегменты . . . 151
10.2.3 Выталкивание . . . 152
10.2.4 Срочные данные . . . 152
10.2.5 Порты приложения . . . 152
10.2.6 Адреса socket . . . 153
10.3 Механизм обеспечения надежности TCP . . . 153
10.3.1 Нумерация и подтверждение . . . 153
10.3.2 Поля портов, последовательности и ACK в заголовке TCP . . . 154
10.4 Установка соединения . . . 154
10.4.1 Сценарий соединения . . . 155
10.4.2 Установка значений параметров IP . . . 156
10.5 Пересылка данных . . . 156
10.6 Закрытие соединения . . . 157
10.6.1 Внезапное завершение . . . 158
10.7 Управление потоком . . . 159
10.7.1 Приемное окно . . . 159
10.7.2 Окно отправки . . . 159
10.8 Заголовок TCP . . . 161
10.8.1 Вариант максимального размера сегмента . . . 161
10.8.2 Использование полей заголовка в запросе на соединение . . . 161

10.8.3 Использование полей заголовка в ответе на запрос соединения . . . 161
10.8.4 Выбор начального порядкового номера . . . 162
10.8.5 Общепринятое использование полей . . . 162
10.8.6 Контрольная сумма . . . 163
10.9 Пример сегмента TCP . . . 163
10.10 Поддержка работы сеанса . . . 164
10.10.1 Зондирование окна . . . 164
10.11 Завершение сеанса . . . 164
10.11.1 Тайм-аут . . . 164
10.11.2 Поддержание соединения . . . 164
10.12 Производительность . . . 164
10.13 Алгоритмы повышения производительности . . . 166
10.13.1 Медленный старт . . . 166
10.13.2 Синдром "бестолкового окна" . . . 166
10.13.3 Алгоритм Нейгла . . . 167
10.13.4 Задержанный ACK . . . 167
10.13.5 Тайм-аут повторной пересылки . . . 168
10.13.6 Пример статистики . . . 170
10.13.7 Вычисления после повторной отправки . . . 170
10.13.8 Действия после повторной пересылки . . . 170
10.13.9 Экспоненциальное торможение . . . 170
10.13.10 Снижение перегрузок за счет уменьшения пересылаемых по сети данных . . . 171
10.13.11 Дублированные ACK . . . 172
10.13.12 Что делается после подавления источника? . . . 172
10.13.13 Статистика TCP . . . 172
10.14 Соответствие требованиям разработчика . . . 173
10.15 Барьеры для производительности . . . 174
10.16 Функции TCP . . . 175
10.17 Состояния TCP . . . 175
10.17.1 Анализ состояний соединения TCP . . . 177
10.18 Замечания о реализациях . . . 178
10.19 Дополнительная литература . . . 178

Глава 11 Конфигурация с помощью BOOTP и DHCP 179

11.1 Введение . . . 179
11.2 Требования протокола BOOTP . . . 179
11.3 Возможности BOOTP . . . 179
11.4 Необходимость DHCP . . . 180
11.5 Первая версия BOOTP . . . 180
11.6 Эволюция BOOTP . . . 181
11.7 Протокол BOOTP . . . 184
11.7.1 Формат сообщения BOOTP . . . 184
11.7.2 Доставка запроса от клиента на сервер . . . 185
11.7.3 Использование промежуточного агента . . . 186
11.7.4 Присваивание IP-адресов . . . 186
11.7.5 Загрузка клиента, знающего собственный IP-адрес . . . 187
11.7.6 Конфигурирование загрузки программного обеспечения . . . 187
11.7.7 Область для разработчиков . . . 187
11.7.8 Ответ безадресному клиенту . . . 187
11.7.9 Счетчик секунд . . . 188
11.8 Возможности DHCP . . . 188
11.8.1 Администрирование и автоматизация конфигурирования . . . 188
11.8.2 Перемещения и изменения . . . 188
11.9 Механизмы DHCP . . . 189
11.9.1 Присваивание IP-адресов . . . 189
11.9.2 Аренда адресов . . . 189
11.9.3 Связывание . . . 189

11.10 Совместимость и различия . . . 189
11.10.1 Типы сообщений . . . 190
11.10.2 Типичный начальный обмен сообщениями между клиентом и сервером . . . 190
11.10.3 Возобновление . . . 190
11.11 Параметры загрузки . . . 191
11.12 Другие методы автоматизации конфигурирования . . . 192
11.13 Дополнительная литература . . . 192

Глава 12 DNS 193

12.1 Введение . . . 193
12.2 Назначение DNS . . . 193
12.3 Программное обеспечение BIND . . . 193
12.4 Определители . . . 193
12.5 Просмотр адресов хостов . . . 194
12.6 Авторитетные ответы и ответы из кеша . . . 195
12.7 Трансляция адресов в имена . . . 196
12.8 Локальные и глобальные серверы имен доменов . . . 198
12.9 Делегирование . . . 199
12.10 Соединение серверов имен с Интернетом . . . 200
12.11 Разработка базы данных сервера имен . . . 200
12.11.1 Зоны . . . 201
12.11.2 Размещение серверов DNS . . . 201
12.11.3 Перенос зон . . . 201
12.12 Данные DNS . . . 201
12.13 Элементы описаний в DNS . . . 201
12.13.1 Записи о ресурсах . . . 202
12.14 Пример файла трансляции имен в адреса . . . 202
12.14.1 Запись SOA . . . 202
12.14.2 Время жизни (Time-To-Live) . . . 204
12.14.3 Дополнение имени . . . 204
12.14.4 Запись о сервере имен . . . 204
12.14.5 Записи об адресе . . . 204
12.14.6 Записи CNAME . . . 204
12.14.7 Записи для почтового обмена . . . 204
12.14.8 Записи TXT и HINFO . . . 205
12.15 Трансляция адресов в имена . . . 205
12.16 Формат сообщений DNS . . . 205
12.16.1 Секция заголовка . . . 206
12.16.2 Секция запроса . . . 206
12.16.3 Секции ответа . . . 206
12.17 Используемый транспорт . . . 207
12.18 Примеры . . . 207
12.19 Дополнительные типы записей . . . 209
12.20 Недостатки DNS . . . 210
12.21 Дополнительная литература . . . 210

Глава 13 Telnet 211

13.1 Введение . . . 211
13.2 Использование telnet для удаленной регистрации . . . 211
13.3 Обращение по telnet к заданному порту . . . 212
13.4 Модель эмуляции терминала в Telnet . . . 213
13.4.1 Сетевой виртуальный терминал . . . 214
13.5 Наиболее распространенные типы терминалов . . . 214
13.5.1 Терминалы ASCII . . . 214
13.5.2 Конфигурирование раскладки клавиатуры . . . 214
13.5.3 Терминалы IBM 3270 и 5250 . . . 215

13.6 Варианты . . . 215
13.6.1 Типы терминалов . . . 216
13.6.2 Согласование типа терминала VT100 . . . 216
13.6.3 Согласование характеристик терминала 3270 . . . 217
13.7 Управление текстовым клиентом telnet . . . 217
13.7.1 Важные управляющие последовательности . . . 218
13.8 Возможности NVT . . . 219
13.8.1 Набор символов N1VT . . . 219
13.8.2 Принтер NVT . . . 219
13.8.3 Взаимодействие клиент/сервер telnet в режиме NVT . . . 219
13.9 Команды telnet . . . 220
13.9.1 Сигнал синхронизации . . . 221
13.9.2 Декодирование наиболее общих команд . . . 221
13.9.3 Кодирование запросов выбора вариантов . . . 221
13.9.4 Дополнительные сведения о вариантах . . . 223
13.10 Применение telnet . . . 223
13.11 Замечания о безопасности . . . 223
13.11.1 Аутентификация в telnet . . . 223
13.12 Замечания о производительности . . . 224
13.13 X Windows . . . 224
13.14 Дополнительная литература . . . 225

Глава 14 Протокол FTP 226

14.1 Введение . . . 226
14.2 Общедоступный и личный доступ FTP . . . 226
14.2.1 Вводный диалог . . . 227
14.3 Модель FTP . . . 228
14.4 Команды FTP . . . 230
14.4.1 Использование команд в текстовом диалоге . . . 230
14.5 Типы данных, структуры файлов и методы пересылки . . . 233
14.5.1 Типы данных . . . 233
14.5.2 Пересылка текста ASCII . . . 233
14.5.3 Пересылка текста EBCDIC . . . 234
14.5.4 Пересылка двоичных данных . . . 234
14.5.5 Структуры файлов . . . 234
14.5.6 Режимы пересылки . . . 234
14.6 Протокол FTP . . . 235
14.6.1 Команды управления доступом . . . 235
14.6.2 Команды управления файлами . . . 236
14.6.3 Команды установки формата данных . . . 236
14.6.4 Команды пересылки файлов . . . 236
14.6.5 Дополнительные команды . . . 237
14.6.6 Команды сайта . . . 237
14.6.7 Восстановления после ошибок и перезапуск . . . 237
14.6.8 Коды ответов . . . 238
14.7 Безопасность . . . 238
14.7.1 Проверка имен хоста клиента . . . 238
14.7.2 PASV или PORT? . . . 238
14.7.3 Промежуточные прокси . . . 239
14.8 Замечания о производительности . . . 239
14.9 Trivial File Transfer Protocol . . . 240
14.9.1 Протокол TFTP . . . 240
14.9.2 Элементы данных протокола TFTP . . . 240
14.9.3 Варианты TFTP . . . 240
14.9.4 Сценарий TFTP . . . 241
14.10 Дополнительная литература . . . 241

Глава 15 RPC и NFS 242

15.1 Введение . . . 242
15.1.1 Назначение NFS . . . 242
15.1.2 Соотношения между NFS, RPC и XDR . . . 243
15.1.3 RPC как стандарт Интернета . . . 243
15.1.4 Реализации NFS и RPC . . . 243
15.2 Модель RPC . . . 243
15.3 Программы и процедуры RPC . . . 244
15.4 Типичная программа RPC . . . 245
15.5 Работа с дубликатами запросов RPC . . . 245
15.6 Portmapper в RPC . . . 246
15.6.1 Назначение Portmapper . . . 246
15.6.2 Процедуры portmapper . . . 247
15.6.3 Просмотр служб RPC через portmapper . . . 247
15.7 Программа rpcbind . . . 248
15.7.1 Назначение rpcbind . . . 248
15.7.2 Процедуры rpcbind . . . 249
15.8 Сообщения RPC . . . 249
15.9 Аутентификация в RPC . . . 250
15.9.1 Нулевая аутентификация . . . 250
15.9.2 Аутентификация систем . . . 251
15.9.3 Аутентификация DES . . . 251
15.9.4 Аутентификация в Kerberos . . . 251
15.10 Пример сообщений RPC версии 2 . . . 251
15.11 XDR . . . 254
15.11.1 Язык описания данных XDR . . . 254
15.11.2 Кодирование в XDR . . . 255
15.12 Программные интерфейсы RPC и XDR . . . 255
15.13 Введение в NFS . . . 255
15.14 Модель файлов NFS . . . 256
15.14.1 Источник формирования модели NFS . . . 256
15.15 Протокол монтирования . . . 257
15.15.1 Процедуры монтирования . . . 258
15.16 Особенности NFS . . . 258
15.17 Протокол NFS . . . 258
15.17.1 Описатели файлов . . . 258
15.17.2 Процедуры NFS . . . 259
15.17.3 Специальные утилиты . . . 260
15.17.4 Блокировка файлов . . . 260
15.17.5 Заметки о реализациях NFS . . . 261
15.17.6 Мониторинг NFS . . . 261
15.18 Дополнительная литература . . . 262

Глава 16 Электронная почта 263

16.1 Введение . . . 263
16.2 Почтовые протоколы Интернета . . . 264
16.3 Модель пересылки почтового сообщения . . . 265
16.4 Пересылка почтового сообщения . . . 266
16.4.1 Сценарий доставки почтового сообщения . . . 266
16.5 Идентификация получателя и обмен сообщениями . . . 267
16.6 Протокол SMTP . . . 269
16.6.1 Диалог при обмене почтой . . . 269
16.7 Временная метка и идентификатор сообщения . . . 271
16.8 Отброшенная почта . . . 271
16.9 Команды SMTP . . . 272
16.10 Коды ответов . . . 273
16.11 Формат сообщений Интернета . . . 273

16.12 Почтовые расширения файлов и MIME . . . 274
16.12.1 Улучшенный агент пересылки почты . . . 275
16.12.2 Диалог в улучшенной версии SMTP . . . 275
16.13 Формат сообщений MIME . . . 275
16.13.1 Заголовки описания типа содержания в MIME . . . 276
16.13.2 Пример сообщения MIME . . . 276
16.13.3 Типы содержания MIME . . . 277
16.13.4 Кодирование содержания . . . 279
16.13.5 Метод кодирования указанными печатными символами . . . 279
16.13.6 Метод кодирования Base64 . . . 279
16.14 Протокол POP . . . 280
16.15 Другие почтовые приложения . . . 281
16.16 Производительность . . . 281
16.17 Безопасность . . . 281
16.17.1 Проблемы с программой sendmail . . . 281
16.17.2 Безопасность электронной почты . . . 282
16.17.3 Secure MIME (S/MIME) . . . 282
16.18 Обмен сообщениями через X.400 . . . 282
16.18.1 Пример сообщения X.400 . . . 283
16.18.2 Именование получателей в X.400 . . . 283
16.18.3 Взаимодействие между X.400 и почтой Интернета . . . 284
16.19 Каталоги ISO/ITU-T . . . 284
16.19.1 Модель каталога . . . 284
16.20 Дополнительная литература . . . 284

Глава 17 Сетевые новости 286

17.1 Введение . . . 286
17.2 Иерархия групп новостей Интернета . . . 286
17.3 Агенты новостей . . . 287
17.4 Модель новостей . . . 287
17.5 Сценарий NNTP . . . 287
17.6 Применение агентов новостей для настольных систем . . . 290
17.7 Протокол NNTP . . . 292
17.7.1 Команды NNTP . . . 292
17.7.2 Коды состояния NNTP . . . 293
17.8 Различия между новостями и рассылочным списком . . . 294
17.9 Дополнительная литература . . . 294

Глава 18 Службы Gopher и WAIS 295

18.1 Введение . . . 295
18.2 Применение Gopher . . . 295
18.3 Типы информации в gopher . . . 296
18.4 Иерархия меню Gopher . . . 298
18.5 Архитектура gopher . . . 298
18.6 Отличия gopher от FTP . . . 298
18.7 Протокол gopher . . . 298
18.8 Файл .names . . . 299
18.9 Служба WAIS . . . 301
18.10 Дополнительная литература . . . 301

Глава 19 WWW 302

19.1 Введение . . . 302
19.1.1 Гипертекст . . . 302
19.1.2 Гипермедиа . . . 302
19.1.3 Гипермедиа и WWW . . . 302
19.2 История WWW . . . 303

19.3 Браузеры WWW . . . 303
19.4 URL . . . 304
19.4.1 URL для гипертекста . . . 304
19.4.2 URL для gopher . . . 304
19.4.3 URL для FTP . . . 304
19.4.4 URL для telnet . . . 305
19.4.5 URL для сетевых новостей . . . 305
19.4.6 URL электронной почты . . . 305
19.4.7 URL для WAIS . . . 305
19.5 Обобщенный формат URL . . . 305
19.5.1 Специальные символы . . . 305
19.6 Введение в HTML . . . 306
19.6.1 Создание документа на HTML . . . 306
19.6.2 Теги HTML . . . 307
19.6.3 Общий формат HTML-документа . . . 307
19.6.4 Заголовки HTML . . . 307
19.6.5 Абзацы и разрывы . . . 307
19.6.6 Неупорядоченные списки . . . 308
19.6.7 Упорядоченные списки . . . 308
19.6.8 Список определений . . . 308
19.6.9 Дополнительные теги . . . 309
19.6.10 Выделение в тексте . . . 309
19.6.11 Ссылки . . . 310
19.6.12 Ссылки на локальные документы . . . 310
19.6.13 Изображения . . . 310
19.6.14 Просмотр исходного кода HTML . . . 311
19.7 Архитектура HTTP . . . 311
19.7.1 Прокси-сервер . . . 311
19.8 Протокол HTTP . . . 312
19.8.1 Пример типичного диалога HTTP . . . 312
19.8.2 Заголовки сообщения . . . 313
19.8.3 Коды состояния . . . 314
19.9 Продолжение совершенствования . . . 314
19.10 Дополнительная литература . . . 314

Глава 20 SNMP 315

20.1 Введение . . . 315
20.1.1 Результат одобрения SNMP в IAB . . . 315
20.2 Модель SNMP . . . 316
20.2.1 Логическая база данных . . . 316
20.2.2 Агенты . . . 316
20.2.3 Диспетчеры . . . 316
20.2.4 Управляющая информационная база . . . 316
20.3 Назначение диспетчера и агента . . . 317
20.3.1 Прокси-агенты . . . 317
20.4 Сущность управляющей информации . . . 317
20.5 Структура управляющей информации . . . 318
20.5.1 Дерево SMI . . . 318
20.6 Имена идентификаторов объектов . . . 319
20.6.1 Идентификация значений в базе данных MIB . . . 320
20.6.2 Лексикографический порядок . . . 320
20.7 Наиболее важные модули MIB . . . 321
20.7.1 MIB-II . . . 321
20.7.2 Модули пересылки . . . 321
20.7.3 RMON MIB . . . 321
20.7.4 Реализация MIB от разработчиков оборудования . . . 322
20.8 Протокол сообщений SNMP . . . 323
20.8.1 Типы сообщений SNMP версии 1 . . . 323
20.8.2 Транспортные протоколы . . . 324

20.9 Форматы сообщений SNMP . . . 324
20.9.1 Формат сообщений gets, sets и responses в версии 1 . . . 324
20.9.2 Запрос get и ответ на него . . . 325
20.9.3 Запрос get-next и ответ на него . . . 326
20.9.4 Запрос set . . . 326
20.9.5 Сообщения trap . . . 326
20.9.6 Проблемы версии 1, исправленные в версии 2 . . . 327
20.9.7 Сообщение get-bulk версии 2 . . . 328
20.9.8 Сообщение trap в версии 2 . . . 328
20.9.9 Сообщение inform версии 2 . . . 328
20.9.10 Другие усовершенствования в версии 2 . . . 328
20.10 Документы MIB . . . 329
20.10.1 Управляемые объекты . . . 329
20.10.2 Первая абстрактная синтаксическая нотация (ASN.1) . . . 329
20.10.3 Типы данных MIB . . . 330
20.10.4 Целые числа . . . 330
20.10.5 Счетчики . . . 331
20.10.6 Масштаб . . . 331
20.10.7 TimeTicks . . . 332
20.10.8 Строки октетов . . . 332
20.10.9 Текстовые соглашения . . . 332
20.10.10 Кодирование типов данных в BER . . . 333
20.11 Что дальше? . . . 333
20.12 Дополнительная литература . . . 333

Глава 21 Программный интерфейс socket 334

21.1 Введение . . . 334
21.1.1 Программный интерфейс Berkeley . . . 334
21.1.2 Ориентация на Unix . . . 334
21.2 Службы socket . . . 335
21.3 Блокированные и неблокированные вызовы . . . 335
21.4 Вызовы socket . . . 335
21.5 Программирование работы TCP socket . . . 336
21.5.1 Модель сервера TCP . . . 336
21.5.2 Пассивное открытие сервера TCP . . . 336
21.5.3 Активное открытие клиента TCP . . . 337
21.5.4 Другие запросы . . . 337
21.6 Серверная программа TCP . . . 338
21.6.1 Вызовы в серверной программе TCP . . . 341
21.7 Клиентская программа TCP . . . 342
21.7.1 Вызовы в клиентской программе TCP . . . 344
21.8 Более простой сервер . . . 344
21.9 Интерфейс программирования socket для UDP . . . 346
21.10 Программа сервера UDP . . . 347
21.10.1 Вызовы в серверной программе UDP . . . 349
21.11 Клиентская программа UDP . . . 349
21.11.1 Запросы в клиентской программе UDP . . . 351
21.12 Дополнительная литература . . . 351

Глава 22 IP версии 6 352

22.1 Введение . . . 352
22.2 Обзор IPv6 . . . 352
22.3 Терминология . . . 353
22.4 Адреса IPv6 . . . 353
22.4.1 Выделение адресов . . . 353
22.4.2 Общие принципы выделения адресов . . . 354
22.4.3 Префикс формата адреса . . . 355
22.4.4 Адресация провайдеров . . . 355
22.4.5 Адреса для независимых сайтов . . . 355

22.4.6 Адреса локальных связей . . . 355
22.4.7 Адреса локальных сайтов . . . 355
22.4.8 Формат многоадресной рассылки . . . 356
22.4.9 Несколькоадресные рассылки . . . 356
22.5 Специальные адреса . . . 357
22.5.1 Неспецифицированные адреса . . . 357
22.5.2 Кольцевые адреса версии 6 . . . 357
22.5.3 Адреса версии 4 . . . 357
22.5.4 Взаимодействие адресов версии 6 с сетями версии 4 . . . 357
22.6 Формат заголовка IPv6 . . . 358
22.6.1 Приоритет . . . 358
22.6.2 Использование меток потока . . . 359
22.7 Дополнительные заголовки IPv6 . . . 359
22.7.1 Использование заголовка Routing . . . 360
22.7.2 Операции с заголовком Routing . . . 360
22.7.3 Дополнительный заголовок Hop-by-Hop . . . 361
22.7.4 Фрагментация . . . 361
22.7.5 Варианты Destination . . . 362
22.8 Автоконфигурация в версии 6 . . . 362
22.8.1 Назначение маршрутизаторов . . . 362
22.8.2 Список адресных префиксов . . . 362
22.8.3 Адреса интерфейсов IPv6 . . . 363
22.8.4 Изменение адресов . . . 363
22.8.5 Тестирование уникальности адреса . . . 363
22.9 Конфигурирование через DHCPv6 . . . 363
22.10 Переход на IPv6 . . . 363
22.10.1 Необходимость изменений . . . 364
22.10.2 Путь перехода на новую версию . . . 364
22.10.3 Изменения в DNS . . . 364
22.10.4 Туннели через сети версии 4 . . . 365
22.11 Резюме . . . 365
22.12 Дополнительная литература . . . 365

Глава 23 ICMPv6 и исследование соседей 366

23.1 Введение . . . 366
23.2 Базовые сообщения ICMPv6 . . . 366
23.2.1 Destination Unreachable . . . 367
23.2.2 Packet Too Big . . . 367
23.2.3 Time Exceeded . . . 367
23.2.4 Parameter Problem . . . 368
23.2.5 Echo Request и Echo Reply . . . 368
23.2.6 Group Membership . . . 368
23.3 Исследование соседей . . . 368
23.3.1 Автоконфигурация через маршрутизаторы . . . 369
23.3.2 Сообщения Neighbor Solicitation и Advertisement . . . 369
23.3.3 Address Resolution . . . 370
23.3.4 Обнаружение дублирования IP-адресов . . . 370
23.3.5 Обнаружение недостижимости соседа . . . 370
23.3.6 Сообщение Redirect . . . 370
23.4 Дополнительная литература . . . 370

Глава 24 Безопасность в IP 371

24.1 Введение . . . 371
24.2 Безопасность . . . 371
24.3 Стратегия безопасности . . . 371
24.4 Сценарии обеспечения безопасности . . . 372
24.4.1 Сценарий 1 . . . 372
24.4.2 Конфигурирование аутентификационной информации для сценария 1 . . . 373
24.4.3 Односторонняя безопасность . . . 373

24.4.4 Количество ключей аутентификации . . . 374
24.4.5 Сценарий 2 . . . 375
24.4.6 Сценарий 3 . . . 376
24.4.7 Обобщение . . . 376
24.5 Элементы протокола безопасности . . . 376
24.5.1 Ассоциации безопасности . . . 377
24.5.2 Authentication Header . . . 377
24.5.3 Режимы транспорта и туннеля . . . 378
24.5.4 Инкапсуляция защищенной полезной нагрузки . . . 378
24.5.5 Аутентификация в режиме туннеля . . . 379
24.5.6 Обслуживание ключей . . . 379
24.6 Дополнительная литература . . . 379

Приложение A Сокращения и аббревиатуры 380

Приложение B RFC и другие документы по TCP/IP 387
B.1 Возможность получения документов RFC . . . 387
B.2 Assigned Numbers . . . 387
B.3 Регистрационные формы . . . 387
B.4 Система именования доменов . . . 387
B.5 Стандарты RFC . . . 387

Приложение C Центры сетевой информации и другие службы 402
C.1 Регистрация . . . 402
C.1.1 Основная служба регистрации NIC . . . 402
C.1.2 Европейская служба NIC . . . 402
C.1.3 Азиатско-Тихоокеанский NIC . . . 403
C.2 Поиск других NIC . . . 403
C.3 Поиск администраторов через WHOIS . . . 403
C.4 Идентификаторы регистрации IPv6 . . . 404
C.5 Функции безопасности CERT . . . 404

Приложение D Маски подсети переменной длины 405
D.1 Введение . . . 405
D.1.1 Маска подсети из семи бит . . . 405
D.1.2 Маска подсети из шести бит . . . 406
D.1.3 Подсети из 9-ти бит . . . 406
D.1.4 10-битовые подсети . . . 407
D.2 Маски подсетей с переменной длиной . . . 407
D.2.1 Присваивание маски линии "точка-точка" . . . 408
D.2.2 Локальная сеть малого офиса . . . 408
D.2.3 Большая локальная сеть . . . 408
D.2.4 Резюме . . . 408

Библиография 409

Глоссарий 412

Введение

1.1 Основы

С самых первых дней использования компьютеров хосты обменивались информацией с непосредственно подключенными к ним устройствами, такими, как устройство чтения перфокарт или устройство печати. Интерактивное использование компьютеров потребовало сначала локального, а затем удаленного подключения терминалов конечных пользователей. Далее последовало объединение нескольких компьютеров в рамках одной организации, что было вызвано необходимостью обеспечения обмена данными между компьютерами или потребностью пользователя одного из компьютеров получить доступ к другому компьютеру.

Разработчики компьютерных систем откликнулись на эти потребности созданием соответствующего аппаратного и программного обеспечения. Однако средства того времени обладали следующими недостатками:

- Программные средства были лицензионными (мы будем использовать для proprietary перевод "лицензионные", в смысле коммерческие и не доступные для бесплатного использования.— *Прим. пер.*) и работали только с оборудованием того же производителя.
- Поддерживался только ограниченный набор локальных и региональных сетей.
- Часто программные средства были очень сложными и требовали различных диалектов для работы с разными устройствами.
- Отсутствие гибкости не позволяло объединить уже существующие сети недорогими и простыми в работе средствами.

Эта ситуация изменилась с появлением протокола управления передачей/протокола Интернета (Transmission Control Protocol/Internet Protocol — TCP/IP) и порождаемыми им технологиями маршрутизации.

Сегодня компьютерные сети организаций стали взаимосвязанными системами. В организациях локальными сетями (Local Area Network — LAN) объединяются настольные рабочие станции (workstation), серверы (server) и хосты (host). Локальные сети (ЛС) соединяются с другими ЛС и региональными сетями (Wide Area Network — WAN).

Обычным для сегодняшнего дня требованием стала возможность взаимодействия между системами независимо от их территориального размещения в сети.

1.2 Приложения TCP/IP

С самого начала в TCP/IP было заложено несколько важных свойств для служб работы с приложениями:

- Терминальный доступ к любому хосту
- Возможность копирования файлов с одного хоста на другой
- Обмен сообщениями электронной почты между любыми двумя пользователями

С течением времени в наборе протоколов TCP/IP появились и другие возможности, очень важные для приложений:

- Печать на удаленном принтере (Remote Printing)

- Работа с сетевой файловой системой (Network File System — NFS)
- Сетевые новости (Network News)
- Gopher
- Word Wide Web (WWW.— Иногда эту службу Интернета в русскоязычной литературе называют "Всемирной паутиной", однако мы будем придерживаться более распространенного среди специалистов названия WWW. — *Прим. пер.*)

Кроме того, расширился набор утилит администрирования и обслуживания сети. Среди новых средств можно назвать:

- Службу каталогов для отображения содержательных сетевых имен хостов на их физические сетевые адреса
- Протокол динамического конфигурирования хоста (Dynamic Host Configuration Protocol — DHCP)
- Сетевое управление хостами, маршрутизаторами (router) и другими сетевыми устройствами

Семейство TCP/IP продолжает жить, расширяться и использоваться. Количество применяющих эти протоколы пользователей увеличивается экспоненциально, разрабатываются новые службы и продукты для модульной интеграции в TCP/IP.

Приложения для WWW привели к революционным переменам в вычислениях клиент/сервер и полностью преобразили методы выполнения прикладных расчетов.

Благодаря программным продуктам для TCP/IP множество организаций смогли подключиться к Интернету, что добавило новые силы этому семейству протоколов, первоначально разработанных для применения в военных и правительственных учреждениях. Именно с последними раньше были связаны основные проблемы сетевой адресации, которые решались путем разработки эффективного механизма адресации и маршрутизации, а также создания надежной защиты сетей от внешнего вторжения.

1.3 Терминология

Обмен данными, как это принято в технических дисциплинах, имеет собственный язык. Все специалисты в этой области используют одни и те же термины. Единственная проблема заключается в том, что разные группы людей одной специальности применяют одни и те же слова для различных понятий, равно как и разные слова для выражения одного и того же понятия.

Мы попытаемся ограничиться очень простым набором терминов, который и будем использовать во всей книге. В этом разделе даны используемые нами определения основных понятий.

1.3.1 Протоколы, элементы, стеки и наборы

Протоколом (protocol) называется набор правил для одной из коммуникационных функций. Например, протокол IP представляет собой набор правил для маршрутизации данных, а TCP определяет правила для надежной и последовательной доставки данных.

Элемент данных протокола (protocol data unit — PDU), или *пакет* (packet) — это форматированный элемент данных, который передается по сети. Пересылаемая в таком пакете информация часто называется полезной нагрузкой (payload).

Стек протоколов (protocol stack) представляет собой набор организованных по уровням протоколов, которые, работая совместно, позволяют приложениям обмениваться данными. Например, стеком является набор протоколов TCP, IP и Ethernet.

Набор протоколов (protocol suite) — это семейство протоколов, работающих совместно и связанных между собой. Набор протоколов TCP/IP обеспечивает множество различных возможностей, начиная от динамического определения адреса сетевого адаптера и заканчивая службой каталогов, определяющей способ доставки сообщения электронной почты.

1.3.2 Хосты

Хостом называется компьютер, который выполняет приложения и имеет одного или нескольких пользователей. Поддерживающий TCP/IP хост работает как конечная точка сетевой коммуникации. Отметим, что персональные компьютеры (ПК), рабочие станции, мини-компьютеры или большие ЭВМ подпадают под определения хоста и каждый из этих компьютеров может реализовать стек TCP/IP.

В книге будут также использоваться термины: "*станция*" (station), "*компьютер*" (computer) и "*компьютерная система*" (computer system) как синонимы термина "*хост*".

1.3.3 Маршрутизаторы

Маршрутизатор (router) управляет пересылкой данных по сети. Первоначально в стандартах TCP/IP использовался термин *шлюз* (gateway), однако в области производства большее распространение получил термин "маршрутизатор". В области коммуникаций термин "*шлюз*" определяет систему, выполняющую некоторое преобразование протокола.

В книге будет применяться термин "*маршрутизатор*", однако при обращении к документации по стандартам TCP/IP нужно помнить, что в них используется термин "*шлюз*".

1.3.4 Интернет

Термин "*интернет*" (со строчной буквы) обозначает сетевую среду (локальную или региональную), объединенную с помощью маршрутизаторов. *Интернет* (с прописной буквы) определяет сообщество из сетей интернета, объединяющее тысячи компьютеров.

1.3.5 Сетевой узел, система и элемент сети

Термины "*сетевой узел*" (network node), "*система*" (system) и "*элемент сети*" (network element) служат для обозначения такого коммуникационного объекта сети, для которого не указаны специализированные свойства (т.е. не задано, что это хост, маршрутизатор или иное устройство, например мост). Пример: "*Целью обслуживания сети является управление и мониторинг всех ее узлов*".

1.3.6 ЛС, региональные сети и связи

Локальные сети (ЛС) предназначены для обслуживания относительно малых географических областей, в основном не превышающих нескольких квадратных километров. Региональные сети охватывают большие географические области и обычно организованы на основе последовательных телефонных линий и устройств для совместного использования коммутации пакетов.

Более общий термин "*связь*" (link) определяет любую среду передачи данных для локальных или региональных сетей, через которую могут взаимодействовать любые узлы (с помощью протоколов уровня связи данных).

1.3.7 Люди

Термин "*хакер*" (hacker) часто используется в положительном смысле — человек, имеющий высокий уровень компьютерных знаний. С другой стороны, хакером называют и человека, пытающегося взломать личные компьютерные сети. В книге мы будем использовать второе значение слова "хакер".

1.3.8 Байты и октеты

Наиболее часто под *байтом* (byte) понимается группа из восьми бит. Однако слово "байт" означает и наименьшую адресуемую часть памяти компьютера. Время от времени некоторые производители компьютеров создают машины с иным размером байтов информации.

Общепринятым для технической документации является термин *октет* (octet), который всегда определяет ровно 8 бит данных. В книге мы будем использовать слова "байт" и "октет" как синонимы, а для обозначения байтов, не равных 8 бит, применять термин "*логический байт*" (logical byte).

1.3.9 Стиль "тупоконечников" и "остроконечников"

Некоторые компьютеры хранят данные начиная от наиболее значимого бита. Такой стиль называется стилем "*тупоконечников*" (Big Endian). Однако другие компьютеры первым размещают менее значимый бит.— стиль "*остроконечников*" (Little Endian; — Оба выражения заимствованы из романа Дж. Свифта "Путешествие Гулливера".— *Прим. пер.*)

Аналогичные названия имеют и стили для стандартов коммуникационного обмена данными, которые зависят от очередности передачи битов.

Стандарты для протоколов Интернета предполагают стиль "тупоконечников". Однако некоторые организации, например Институт инженеров по электротехнике и электронике (IEEE), предлагают при передаче начинать пересылку с битов или байтов наименьшей значимости (для обычных арабских чисел наименьшую значимость имеет крайняя правая цифра, а наибольшую — крайняя левая.— *Прим. пер.*).

1.4 Реализации с использованием оборудования различных производителей

В отличие от использовавшихся ранее лицензионных сетевых протоколов TCP/IP реализованы на компьютерах различных производителей и могут использовать программное обеспечение независимых компаний.

Реализация базировалась на принятых стандартах и бесплатном программном обеспечении, разработанном добровольцами. Строгие ограничения содержатся в дополнительных документах *Host Requirements* (требования к хостам) и *Router Requirements* (требования к маршрутизаторам).

Достигается определенный уровень взаимодействия систем, и конечные пользователи убеждаются, что нужные приложения прекрасно работают друг с другом.

Однако, если заглянуть за кулисы происходящих событий, можно обнаружить, что иногда разработчики ограничивают некоторые возможности с целью достичь большей производительности или улучшить процедуру исправления ошибок. Часто разработчики программного обеспечения просто не до конца понимают некоторые детали стандартов и в результате не реализуют в своих продуктах имеющиеся возможности.

Следовательно, любой из описанных в этой книге механизмов необязательно будет реально реализован в любом из программных продуктов. При покупке программного обеспечения TCP/IP для хоста нужно всегда проверять его на соответствие документу *Host Requirements*. Разработчик должен ответить на все вопросы о реализации в своем продукте любой из определенных в этом документе возможностей (эти возможности определяются как "должен и обязан" — must and should).

Разработчикам программного обеспечения можно рекомендовать внимательно изучить по этой книге все, *что* должны поддерживать протоколы, равно как и то, *каким образом* это должно быть реализовано. Однако одной этой книги будет недостаточно. Дело в том, что стандарты TCP бесплатно доступны в интерактивном виде (см. приложение B). Эти документы постоянно находятся в состоянии доработки — в них включаются новые возможности и более совершенные механизмы реализации и исключаются устаревшие разделы.

1.5 Диалоги

В книге приведено много примеров интерактивных диалогов (листингов работы пользователя). Текстовые диалоги были получены на компьютерах компании Sun Microsystems. Во многих примерах работа проводилась с хостом *tigger.jvnc.net*, который находится в Принстоне (Нью-Джерси). Это большой *сервер*, обслуживаемый провайдером *Global Enterprise Systems* (GES), который ранее назывался *JVNC* (именно поэтому имя сервера заканчивается на *jvnc.net*). Статистические данные о работе этого *сервера* заслуживают внимания, поскольку он взаимодействует через Интернет со многими *хостами* по всему миру. Несколько примеров были получены на компьютерах Йельского университета.

TCP/IP реализованы с помощью похожих пользовательских интерфейсов и наборов команд на различных типах компьютеров. Поэтому приведенные примеры диалогов будут очень похожи или полностью идентичны для совершенно разных систем. В текстовых диалогах вводимые пользователем фрагменты выделены полужирным шрифтом, а приглашения и ответы компьютера набраны обычным шрифтом.

Несколько примеров представляют работу с графическим пользовательским интерфейсом (Graphical User Interface — GUI) для приложений TCP/IP, выполняющихся на компьютерах Windows и Macintosh. Некоторые из этих примеров показывают работу приложения *Chameleon* компании Netmanage и браузера *Netscape Navigator* компании Netscape, Inc. Отдельные экраны получены в *HP Open View for Windows Workgroup Node Manager* компании Hewlett-Packard и *NSLookup for Windows* компании Ashmount Research Ltd. Работа электронной почты демонстрируется на примере *Eudora* компании Qualicomm для компьютеров Macintosh.

1.6 Дополнительная литература

Как и в любой другой работе по цифровым коммуникациям, в данной книге множество аббревиатур. Все они представлены в приложении A. Глоссарий содержит определение всех встречающихся в книге терминов.

В приложении B приведен список документов, которые определяют TCP/IP и связанные с этими протоколами возможности. Приложение C посвящено службам сетевых информационных центров (Network Information Center — NIC). Там же рассмотрены способы обращения в эти центры. Приложение D содержит примеры более эффективного использования IP-адресов (маски подсети переменной длины).

Глава

1

TCP/IP: что это такое и откуда взялось

1.1 Введение

В конце 60-х гг. Агентство перспективных исследовательских проектов (Advanced Research Project Agency — ARPA) Министерства обороны США (позднее переименованное в DARPA) начало сотрудничать с университетами и другими исследовательскими организациями в области новых технологий обмена данными.

Все эти организации совместно разработали Сеть агентства перспективных исследовательских проектов (Advanced Research Project Agency Network — ARPANET), первую сеть с коммутацией пакетов. Экспериментальный вариант этой сети из четырех узлов был запущен в эксплуатацию в 1969 г. Реализация прошла успешно, а ее возможности были протестированы на сети, протянувшейся через всю территорию США. В 1975 г. Агентство оборонных коммуникаций взяло на себя ответственность за эксплуатацию созданной сети, которая все еще рассматривалась как экспериментальный вариант.

1.1.1 Зарождение TCP/IP

Первые протоколы ARPANET работали медленно и часто приводили к краху сетевых коммуникаций. В статье Винтона Г. Серфа и Роберта Е. Кана A Protocol for Packet Network Interconnections (журнал *IEEE Transactions of Communications*, май 1974 г.) был предложен новый набор основных протоколов. В этой работе были заложены основы для последующей разработки *протокола Интернета* (IP) и *протокола управления передачей* (TCP). Начиная с 1980 г., потребовалось около трех лет для преобразования хостов ARPANET на новые протоколы (число хостов к тому времени приближалось к 100).

Возможности новых протоколов были продемонстрированы в 1978 г., когда терминал, расположенный на движущемся по Калифорнии автофургоне, переслал данные, сформированные в пакеты, на узел SRI International через всю территорию США и далее по спутниковой связи на хост в Лондоне (см. рис. 1.1).

В начале 80-х гг. ARPANET была полностью переделана на новые протоколы. К 1983 г. эта сеть содержала уже более 300 узлов и предоставляла своим пользователям обширные ресурсы. В 1984 г. исходная сеть ARPANET была разделена на две отдельные сети: первая, сохранившая исходное название, предназначалась для исследований и новых разработок, вторая — названная MILNET — стала неклассифицированной сетью для военных целей.



Рис. 1.1. Демонстрация работы TCP/IP с использованием нескольких различных сетевых технологий

В начале 80-х гг. ARPANET была полностью переделана на новые протоколы. К 1983 г. эта сеть содержала уже более 300 узлов и предоставляла своим пользователям обширные ресурсы. В 1984 г. исходная сеть ARPANET была разделена на две отдельные сети: первая, сохранившая исходное название, предназначалась для исследований и новых разработок, вторая — названная MILNET — стала неклассифицированной сетью для военных целей.

1.2 Принятие новых протоколов

В 1982 г. Министерство обороны США (Department of Defence — DOD) заимствовало набор коммуникационных протоколов ARPANET в качестве источника для формирования собственных распределенных вычислительных сетей.

В 1983 г. аналогичное заимствование набора протоколов TCP/IP было выполнено для военного стандарта. Принятие TCP/IP позволило расширить область использования этих протоколов на другие правительственные учреждения, что создало обширный рынок для данных технологий.

1.3 Характеристики TCP/IP

TCP/IP обладает уникальными характеристиками, которые обеспечивают долговечность этих протоколов. Архитектура TCP/IP позволяет объединять сетевые кластеры, формируя то, что называется "*интернетом*". Для пользователя интернет выглядит как одна большая сеть, сформированная из всех хостов, подключенных к отдельным сетевым кластерам.

Протоколы TCP/IP были разработаны с учетом независимости от аппаратного обеспечения хостов и их операционных систем, а также от используемой среды передачи данных и технологий связи. Эти протоколы обеспечивают высокую надежность, сохраняя работоспособность даже при высоком уровне сетевых ошибок, и, кроме того, поддерживают прозрачную перестраиваемую маршрутизацию при потере сетевых узлов или строк данных.

1.3.1 Доступность TCP/IP

Когда наличие TCP/IP стало обязательным требованием для всех компьютеров, закупаемых Министерством обороны США, разработчикам пришлось заняться реализацией TCP/IP для обеспечения условий правительственных поставок. На рис. 1.2 показано, как с помощью TCP/IP могут взаимодействовать различные системы, локальные и региональные сети.

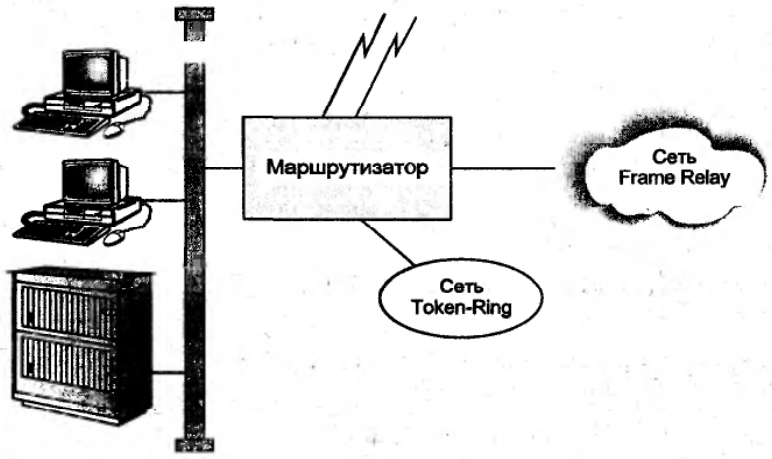

Рис. 1.2. TCP/IP — окружение для оборудования от различных производителей и различных сетей

Министерство обороны США обеспечило доступность TCP/IP, реализовав эти протоколы в Unix (по именам разработчиков — BBN — Bolt, Beranek, Newman). Далее Калифорнийский университет в Беркли использовал код BBN в операционной системе Berkeley Software Distribution (BSD) версии 4.2 (одна из разновидностей UNIX). Эта операционная система и ее более поздние версии были реализованы на многих аппаратных базах. Позднее TCP/IP был добавлен и в System V Unix компании AT&T.

В 90-х гг. TCP/IP начали использоваться в коммерческих программных продуктах и стали наиболее универсальными протоколами из доступных на рынке. Процесс интеграции протоколов TCP/IP в серверы и настольные системы был поистине стремительным.

Кроме того, поддержка TCP/IP реализована практически во всех сетевых технологиях (см. главу 4).

1.4 Интернет

Кроме простоты при объединении нескольких сетей, протоколы TCP/IP открыли дорогу для объединения с ARPANET сетей университетов и исследовательских организаций, что в конечном итоге привело к созданию суперсети *Интернет*. На протяжении 80-х годов магистральные соединения Интернета обеспечивались средствами ARPANET.

Характеристики протоколов TCP/IP обеспечили стабильное и единообразное расширение сети Интернет, которая стала самой большой всемирной сетью, объединяющей правительственные и военные сети, а также сети университетов и коммерческих организаций (каждая из которых может сама состоять из сотен подсетей). В 1985 г. была организована новая магистральная сеть National Science Foundation Net (NSFNET), которая обеспечила скоростные соединения для исследовательских центров и суперкомпьютеров.

Благодаря правительственной поддержке была сформирована инфраструктура *региональных служб провайдеров* (regional Service Provider), охватившая всю территорию США. Университеты и исследовательские лаборатории подключались к ближайшему региональному провайдеру, который обеспечивал магистральные соединения внутри общей сети.

Стремительное расширение Интернета привело к формированию провайдеров в десятках стран по всему миру.

В 1994 г. в Интернете уже были объединены миллионы компьютеров, и эта область стала реальным сектором рынка компьютерных технологий. Поддержка сети осуществлялась National Science Foundation (NSF), а соединенные между собой провайдеры США образовали огромный коммутирующий центр, распределенный по всей территории страны.

Интернет оставался инкубатором для новых технологий. Реализованные в этой сети службы электронной почты, сетевых новостей и досок объявлений предоставляли пользователям возможность общения и обмена идеями. Исследователи, системные программисты и администраторы обменивались между собой методами исправления ошибок в программных средствах, решениями проблем сетевого взаимодействия или рекомендациями по повышению производительности. Разработчики программного обеспечения публиковали в Интернете бесплатные копии бета-версий своих продуктов, что позволяло обычным пользователям загружать эти программы, испытывать их и комментировать их работу.

1.5 INTERNIC

Многие годы координирующую роль в Интернете осуществляло Министерство обороны США. Принадлежащий ему комитет (DDN Network Information Center — DDN NIC) обслуживал пользователей, системных администраторов, координаторов сайтов и администраторов сетей.

Весной 1993 г. обслуживание гражданских пользователей Интернета было передано в National Science Foundation, которая в настоящее время состоит из двух агентств:

- InterNIC *Registration Services* (служба регистрации InterNIC), принадлежащая Network Solutions, Inc. (Герндон, Вирджиния)
- InterNIC *Directory and Database Services* (служба каталогов и баз данных InterNIC), принадлежащая AT&T

Дополнительные регистрационные центры были созданы и в других странах мира. Такие центры координируют именование и адресацию компьютеров в Интернете.

InterNIC Directory and Database Services служит депозитарием стандартов Интернета и других информационных документов. Все документы доступны бесплатно.

В приложении C приведены дополнительные сведения об InterNIC и других информационных центрах Интернета.

1.6 IAB, IETF и IESG

Разработка новых протоколов TCP/IP и обслуживание старых координируется Советом по архитектуре Интернета (Internet Architecture Board — IAB, который ранее назывался Internet Activities Board). IAB идентифицировал техническую специализацию новых средств. Например, в прошлом IAB

направлял исследовательские работы по созданию новых протоколов сетевого управления, более функциональных протоколов маршрутизации и следующих версий IP.

В 1992 г. была сформирована *Ассоциация Интернета* (Internet Society), в которую и вошла IAB. Целью Internet Society является помощь в расширении и обеспечении успешной работы Интернета.

IAB контролировала несколько важных групп: *рабочую группу технологии Интернета* (Internet Engineering Task Force — IETF), которая разрабатывает и реализует новые протоколы, *управляющую группу технологии Интернета* (Internet Engineering Steering Group — IESG), которая осуществляет руководство и контроль за деятельностью IETF.

1.6.1 Рабочие группы и разработка протоколов

Членство в IETF является добровольным. Для решения определенной проблемы формируется рабочая группа из технических экспертов. Члены такой группы разрабатывают методологии, объединяющие теоретические исследования с последующей реализацией.

Реально правильность и полнота спецификации протокола проверяются при создании двух независимых версий одного протокола. Далее следует процесс *разработка-реализация-эксперимент-пересмотр*, позволяющий улучшить и расширить спецификацию протокола, равно как и повысить производительность ее реализации.

На практике при разработке протокола происходит устранение многих недостатков и пересмотр многих первоначальных положений еще до того, как спецификация протокола будет одобрена. В архитектуру протоколов стараются не закладывать слишком большие требования к системным ресурсам или решения, снижающие общую производительность систем.

1.6.2 Другие источники протоколов Интернета

Хотя большинство протоколов TCP/IP разрабатывается и реализуется рабочими группами IETF, существенное участие в этом процессе принимают исследовательские группы из университетов и коммерческих организаций. Чтобы независимые проекты получили одобрение, они должны быть и пригодны, и полезны.

Исходные коды новых протоколов часто помещаются в общедоступные базы данных Интернета. Разработчики могут использовать эти коды как отправную точку в своих собственных реализациях. Такой метод имеет множество преимуществ. Прежде всего, снижается стоимость разработки и время ее проведения. Одинаковые исходные коды позволяют также достичь согласования в работе продуктов от различных разработчиков.

Пользователи могут копировать и устанавливать исходные коды протоколов на свои системы. Разумеется, применение бесплатных программных продуктов не предусматривает технической поддержки и обслуживания, предоставляемых разработчиками для коммерческих реализаций.

1.7 Requests For Comments

Спецификации новых протоколов распространяются в документах, называемых *запросами комментариев* (Requests For Comments — RFC). Все документы RFC имеют последовательные номера. На сегодняшний день существует уже более тысячи таких документов.

Пользователи могут получить RFC в службе каталогов и баз данных InterNIC. Кроме того, эти документы существуют на множестве общедоступных сайтов по всему миру, поскольку разрешено их свободное копирование между любыми сайтами Интернета посредством пересылки файлов.

Иногда спецификации протоколов подвергаются изменениям, например вследствие исправления обнаруженных ошибок, а также для повышения производительности или добавления новых возможностей. Измененные протоколы публикуются в RFC с новыми номерами.

InterNIC обслуживает индексы для RFC, а для устаревших документов предоставляется номер заменяющего RFC. Например, индекс для RFC 1098 (стандарт SNMP) содержит ссылки на устаревший документ RFC (1067) и пополняющий RFC 1098 документ с номером 1157:

```
1098 Case, J.D.; Fedor, M.; Schoffstall, M.L.; Davin, C.
Simple Network Management Protocol (SNMP). 1989 April; 34p.
(Format: TXT=71563 bytes) (Obsoletes RFC 1067; Updated by RFC 1157)
```

Не все RFC описывают протоколы. Некоторые из них служат для систематизации и описания сведений, используемых в Интернете. Существует RFC с рекомендациями по выбору имен для компьютеров. Другой RFC содержит руководство по администрированию сетей TCP/IP и реализации в них средств безопасности. Имеются RFC, описывающие стратегии повышения производительности,

экспериментальные алгоритмы или обсуждающие этические вопросы Интернета. После пересмотра некоторые RFC получают статус документов *Best Current Practices* — BCP (описание лучшего текущего способа применения).

1.7.1 Состояние и статус стандартов

IAB периодически публикует информацию о работе над протоколами. Стадии разработки определяют текущее *состояние* протокола:

- Experimental (экспериментальный)
- Proposed (предлагаемый)
- Draft (черновик)
- Standard (стандарт)

Некоторые протоколы маркируются как *информационные* (informational), другие, не использующиеся в настоящее время,— как *исторические* (historical).

Протоколы классифицируются также по уровню требований. Некоторые протоколы являются стандартами, другие применяются только в специальных целях. Отдельные протоколы утратили свою полезность, и их применение отменено. Формальные требования отражают *статус* протокола:

- Required (требуется использовать)
- Recommended (рекомендован к применению)
- Elective (необязателен)
- Limited Use (ограниченное использование)
- Not recommended (не рекомендован к применению)

Текущий статус и состояние протоколов Интернета описываются в RFC, называемом *IAB Official Protocol Standards* (официальные стандарты протоколов IAB). Этот документ периодически изменяется, отражая изменение номеров RFC.

1.7.2 Присвоенные номера

Сетевые параметры, специальные сетевые адреса, имена служб и стандартные идентификаторы терминалов либо компьютерных систем перечислены в RFC с именем *Assigned Numbers* (присвоенные номера).

Присвоенные номера Интернета администрируются *организацией авторизации присвоенных номеров* (Internet Assigned Numbers Authority — IANA), расположенной в настоящее время в Институте информационных служб университета Южной Калифорнии.

Документ *Assigned Numbers* содержит почтовые адреса, номера телефонов и адреса электронной почты, которые разработчики протоколов могут использовать для регистрации информации об авторизации.

1.7.3 RFC и стимулирование сетевого взаимодействия продуктов различных производителей

То, что пользователи не должны придерживаться только одной компьютерной архитектуры, стало главной причиной одобрения TCP/IP в качестве коммуникационных стандартов правительственными учреждениями США. Эти организации желали покупать оборудование на рынке с реальной конкуренцией, когда предъявляемым требованиям удовлетворяет продукция различных разработчиков. Принятие и обслуживание стандартов должно было привести к снижению цен и лучшему качеству услуг.

Однако при использовании оборудования различных производителей возникает несколько проблем:

- Стандарты часто содержат необязательные возможности. При реализации различными производителями это может усложнить взаимодействие.
- Разработчики иногда не до конца понимают требования стандартов, вследствие чего их продукты работают не вполне корректно.
- Возможны ошибки в спецификациях стандартов.
- Некоторые реализации, даже при аккуратном соблюдении системным администратором всех требований, могут не обеспечивать должной гибкости и более точной настройки конфигурационных параметров для повышения производительности.

- Отдельная система с неудовлетворительными характеристиками пересылки или приема/передачи данных (за счет использования неэффективных алгоритмов) может снизить производительность всех систем сети.

Два документа RFC (октябрь 1989 г.) были посвящены как указанным проблемам, так и коррекции ошибок, разъяснению определений, спецификации необязательных возможностей, перечислению конфигурационных параметров и идентификации наиболее производительных алгоритмов. Наиболее важно то, что в этих документах специфицируются единые требования к реализации хостов. В прошлом сказывалось отсутствие этих документов. Корректность операций, взаимодействие и производительность существенно улучшились при строгом соблюдении требований следующих RFC:

RFC 1122, Requirements for Internet Hosts — Communication Layers (Требования к хостам Интернета — уровни взаимодействия). В этом документе описывается уровень связи данных, IP и TCP.

RFC 1123, Requirements for Internet Hosts — Application and Support (Требования к хостам Интернета — приложения и их поддержка). Этот документ определяет удаленную регистрацию, пересылку файлов, электронную почту и различные прикладные службы.

В 1995 г. был опубликован документ, относящийся к операциям маршрутизаторов:

RFC 1812 Requirements for IP Version 4 Routers (Требования к маршрутизаторам для протокола IP версии 4).

1.7.4 Связанные документы

Серия RFC не содержит спецификаций протоколов и была опубликована как отдельный набор документов *For Your Information* (FYI — К вашему сведению). Например: RFC 1325 *Answers to commonly asked "new Internet user" questions* (Ответы на наиболее распространенные вопросы новых пользователей Интернета).

Еще одна серия, *Internet Engineering Notes* (IEN — Заметки по технологии Интернета), содержит набор дискуссионных материалов, написанных еще в первые годы разработки протоколов Интернета.

1.8 Другие информационные ресурсы

В Интернете существует множество WWW-серверов и общедоступных файловых систем, которые размещаются в университетах, исследовательских центрах и коммерческих организациях. В этих системах предлагается различная информация о сетях, например: копии RFC, заметки об обсуждении новых алгоритмов, отчеты о тестировании производительности, исходные коды инструментов мониторинга работы сетевых протоколов, бесплатное программное обеспечение и сведения о программных продуктах. Каждый пользователь Интернета, способный передавать файлы или работать в браузере WWW, может скопировать любой из этих документов или исходных кодов.

Некоторые сети соединены с Интернетом посредством шлюзов электронной почты. Пользователи хостов таких сетей не имеют возможностей для пересылки файлов. К счастью, многие информационные системы обеспечивают распространение информации в виде сообщений электронной почты.

1.9 Open System Interconnection

Взаимодействие открытых систем (Open System Interconnection — OSI) стало результатом международных усилий по созданию компьютерных коммуникационных стандартов и базовых прикладных служб. Формально OSI разработана в рамках *Международной организации по стандартизации* (International Organization for Standardization — ISO), созданной для поддержки обмена и кооперации в сфере научных исследований и технологий. Стандарты ISO публикуются как документы этой организации.

Модель OSI (OSI model) стала обязательной частью любого курса обучения сетевым технологиям. Эта модель отражает базовые понятия, касающиеся идентификации места каждого протокола в общей схеме коммуникации.

Протоколы OSI используются только на небольшом числе европейских сайтов, но IETF опубликовала несколько RFC, относящихся к взаимодействию окружений TCP/IP и OSI.

Глава

2

Обзор служб набора протоколов TCP/IP

2.1 Введение

Почему семейство протоколов TCP/IP получило столь широкое распространение? Прежде всего, благодаря способности к взаимному объединению гетерогенных локальных и глобальных сетей. Не менее важной является способность создавать основы для коммуникаций "равный с равным" (peer-to-peer), а также базовых служб поверх такого взаимодействия. Кроме того, с самого начала протоколы TCP/IP ориентировались на поддержку взаимодействия типа клиент/сервер.

2.2 Коммуникации между приложениями

Существует два основных типа взаимодействия между приложениями. Первый тип — связи, *ориентированные на создание соединения* (connection-oriented), — применяется при работе приложения с потоком данных. Второй вариант — *связи без создания соединения* (connectionless) — предполагает обмен независимыми сообщениями и подходит для случайных взаимодействий между приложениями при небольшом объеме пересылаемых данных.

2.2.1 Коммуникации с созданием соединений (TCP)

Протокол TCP (Transmission Control Protocol) отвечает в TCP/IP за надежные коммуникации с созданием соединения по принципу "равный с равным". Сеанс регистрации с терминала и пересылка файлов выполняются с помощью TCP.

2.2.2 Коммуникации без создания соединений (UDP)

Некоторые операции обмена данными не требуют постоянного взаимодействия систем. Например, база данных на сетевом сервере может содержать таблицы имен сотрудников компании и их телефонные номера. Узнать номер телефона конкретного сотрудника можно при передаче на сервер запроса с указанием имени этого сотрудника. Сервер должен будет ответить сообщением, содержащим соответствующий телефонный номер. Такой тип взаимодействия поддерживается *протоколом пользовательских датаграмм* (User Datagram Protocol — UDP).

2.2.3 Интерфейс программирования socket

Реализации TCP/IP обычно предоставляют для разработчиков коммуникационный программный интерфейс. Многие из таких интерфейсов основаны на *программном интерфейсе socket* (дословно — "штепсельная розетка", "гнездо"), который первоначально был разработан для операционной системы Unix университета Беркли.

К программному интерфейсу socket относятся:

- Простые подпрограммы для создания, пересылки и приема независимых сообщений, используемых при коммуникациях без создания соединения по протоколу UDP
- Программы для создания соединения TCP, передачи и приема данных, а также для закрытия созданного соединения

2.2.4 Программный интерфейс RPC

Хотя и не так широко распространенный, как socket, программный интерфейс *вызова удаленных процедур* (Remote Procedure Call — RPC) для соединений типа клиент/сервер достаточно часто используется в различных системах. Первоначально он был реализован в компьютерах компании Sun Microsystems, а затем перемещен на многие другие платформы.

Клиент, использующий интерфейс RPC, способен вызывать подпрограммы, которые автоматически пересылают запрос на сервер. Сервер исполняет затребованную подпрограмму и возвращает ее выходные результаты клиенту. Именно с этим связано название данного интерфейса, поскольку локально запущенная программа может инициировать обработку на удаленной системе.

Например, описанное в п. 2.2.2 приложение для просмотра телефонных номеров может быть реализовано через программы RPC.

2.3 Основные службы

Реализация TCP/IP предполагает доступность, по крайней мере, трех прикладных служб: пересылки файлов, удаленной регистрации и электронной почты. Многие продукты имеют клиентские и серверные службы для WWW, а также функции для печати на удаленных принтерах.

2.3.1 Пересылка файлов

Пересылка файлов (file transfer) является старейшей службой TCP/IP. Протокол пересылки файлов (File Transfer Protocol — FTP) разрешает пользователю пофайловое копирование с одной системы на другую. FTP имеет дело с простыми типами файлов, такими как текстовые файлы в коде для обмена информацией Американского национального института стандартов (American National Standard Code for Information Interchange — ASCII) или неструктурированные двоичные файлы. FTP обеспечивает пользователю доступ к удаленной файловой системе для выполнения служебных операций: переименования и удаления файлов либо создания новых каталогов.

2.3.2 Доступ с терминала

В начале 70-х гг. многие производители компьютеров создавали модели терминалов, которые были совместимы только с их собственными компьютерными системами. Министерство обороны США закупало оборудование у различных производителей и, естественно, настаивало на обеспечении для каждого терминала единообразного доступа к любому компьютеру сети. Протокол терминального доступа *telnet* сделал возможными такие операции для любого типа терминала. С течением времени *telnet* расширил свои возможности по работе с самыми разнообразными моделями терминалов и операционными системами.

2.3.3 Электронная почта

Электронная почта (далее будем называть ее просто почтой, а когда речь пойдет об обычной почтовой службе, это будет оговорено дополнительно.— *Прим. пер.*) привела к распространению TCP/IP среди многих конечных пользователей. Стандартизованы два аспекта почтовой службы:

- Формат почтового сообщения, пересылаемого между пользователями. Определены форматы для неструктурированного текста, текста, состоящего из нескольких частей, и мультимедийных сообщений.

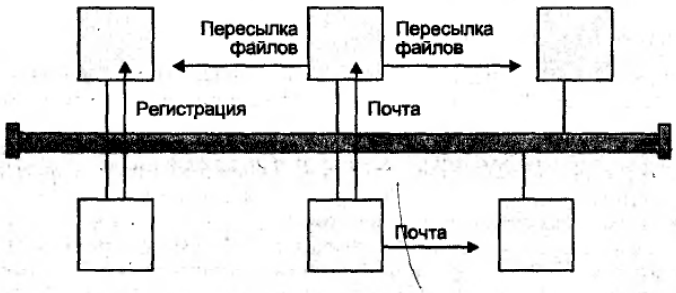

Рис. 2.1. Прикладные службы сети TCP/IP

- Механизмы для направления и пересылки методом сохранить-переслать дальше при обмене почтовыми сообщениями между хостами. С первых дней Интернета для этого применяется *простой протокол пересылки почты* (Simple Mail Transfer Protocol — SMTP). Более поздние расширения добавили этой службе новые функции.

Многие лицензионные почтовые системы были подключены к Интернету, что существенно расширило круг потенциальных пользователей почтовой службы.

На рис. 2.1 показано взаимодействие между хостами сети. Отметим, что TCP/IP полностью соответствует сетевой архитектуре "равный с равным" и любой из хостов может выступать как клиент или сервер, а также одновременно как клиент и сервер.

2.3.4 Служба WWW

Word Wide Web (WWW) — наиболее привлекательная система из всех прикладных служб клиент/сервер, реализованных в TCP/IP. Пользователь может получить доступ к прекрасно оформленным документам, содержащим графические изображения и звуковые файлы, легко перемещаться между сайтами сети одним щелчком мыши и проводить поиск в огромных информационных архивах.

2.4 Дополнительные службы

К набору протоколов TCP/IP были добавлены и другие службы. Ниже рассмотрены наиболее популярные и широко распространенные.

2.4.1 Доступ к файлам

Файловые серверы дают пользователю возможность работать с удаленными файлами так, как если бы они располагались на локальной системе. Первоначально файловые серверы получили распространение в локальных сетях персональных компьютеров как средство совместного использования дисковых ресурсов, централизации обслуживания и резервного копирования.

Многие продукты TCP/IP включают сетевую файловую систему (Network File System — NFS). Такие продукты поддерживают одну или обе роли NFS:

Доступ клиента к файлам. Позволяет компьютеру получить доступ к удаленным файлам как к локальным. Конечный пользователь и локальные программы могут не учитывать реальное размещение файла в сети.

Файловый сервер. Обслуживает каталоги, доступ к которым разрешен для других сетевых компьютеров.

2.4.2 Новости

Приложения для работы с электронными новостями появились для обслуживания локальных электронных досок объявлений (bulletin board) и распроетранения находящихся на них данных на другие сайты сети.

Многие организации используют бесплатное программное обеспечение для публикации внешней информации электронным способом. Другие организуют доступ к группам новостей Интернета, на которых обсуждаются самые разнообразные темы, начиная от спорта и кончая физикой плазмы. Такое программное обеспечение применяется и для доступа к коммерческим информационным службам новостей, например к Рейтер, AP или UPI.

2.4.3 Служба имен DNS

Для использования сетевых служб требуется способ идентификации удаленных компьютеров. Пользователи и программы могут указывать нужный компьютер по его имени, которое легко запомнить или ввести.

Для создания соединения с хостом имя хоста должно быть преобразовано в его цифровой адрес. Первоначально каждый хост TCP/IP хранил полный список всех имен и адресов для хостов сети. Однако при стремительном расширении Интернета это стало невозможным, поскольку число хостов стало измеряться тысячами и миллионами.

Система именования доменов (Domain Name System — DNS) предназначена для решения этой проблемы. DNS представляет собой базу данных для имен и адресов хостов, которая распределена по тысячам отдельных серверов. Протокол DNS разрешает пользователю послать запрос к базе данных локального сервера и получить ответ, который, возможно, придет от удаленного сервера.

Кроме трансляции имен и адресов хостов, серверы DNS предоставляют информацию для маршрутизации сообщений электронной почты в точку назначения.

2.4.4 Коммерческое программное обеспечение

Многие сторонние разработчики создают приложения, работающие поверх TCP/IP. Например, производители баз данных соединяют настольные компьютеры-клиенты с серверами средствами TCP/IP.

2.4.5 Управление сетью

По прошествии некоторого времени многие инструменты сетевого управления стали создаваться на основе набора протоколов TCP/IP. Например, существуют команды, позволяющие определить, находится ли конкретная система сети в рабочем состоянии, просмотреть ее текущую загрузку или получить список доступных в сети служб.

Такие команды очень полезны, но для централизации сетевого управления требуется единый и полный набор команд. В Интернете для этого был разработан *простой протокол сетевого управления* (Simple Network Management Protocol — SNMP), позволяющий обслуживать любой сетевой узел, начиная от простейшего устройства и заканчивая операционной системой хоста или прикладным программным обеспечением.

2.4.6 Диалоги

Лучшим способом понять работу служб TCP/IP является их применение на практике. Эта глава завершается несколькими краткими примерами интерактивных диалогов, иллюстрирующими работу служб сети. Во всех диалогах вводимые пользователем команды напечатаны полужирным шрифтом. Это соглашение будет применяться по всей книге. Основы работы со службами достаточно просты.

После начала работы каждая служба выводит некоторое сообщение. Конечный пользователь может по большей части просто игнорировать такие сообщения. В следующих главах мы еще вернемся к внутренним механизмам работы различных служб и подробно рассмотрим сведения, выводимые в этих информационных сообщениях.

2.4.7 Диалог доступа с терминала

Доступ с терминала является примером работы с простейшей службой. В приведенном ниже примере запрашивается соединение по *telnet* с хостом *bulldog.cs.yale.edu*. После установки соединения *telnet* информирует, что *комбинация клавиш* CONTROL-] служит для возврата к сеансу с локальной системой. Затем удаленный хост выводит приглашение регистрации *login:*, и с этого момента начинается обычная работа с удаленным хостом, как если бы это был локальный компьютер.

```
> telnet bulldog.cs.yale.edu
Trying 128.36.0.3 ...
Connected to bulldog.cs.yale.edu
Escape character is '^]' .

login:
```

Хотя это очень простой диалог с пользователем, за кулисами происходит достаточно большая работа. *Telnet* просматривает базу данных DNS и определяет, что адресом требуемой системы является 128.36.0.3. Именно этот адрес и будет использован *telnet* для соединения с удаленным хостом.

Схемы именования и нумерации TCP/IP будут рассмотрены в главе 5. Однако уже сейчас можно понять, что имя состоит из нескольких разделенных точками слов, а адрес — из четырех цифр, также разделенных точками.

2.4.8 Просмотр имен в базе данных DNS

Как и многие системы TCP/IP, используемый нами локальный хост имеет клиентское приложение *nslookup* (от network server lookup — просмотр сетевого сервера), которое разрешает пользователю интерактивно запросить базу данных DNS.

Ниже показан пример вывода имени и адреса локального сервера по умолчанию при вводе команды *nslookup*. Запрос к базе данных формируется при указании имени нужного хоста. В ответе повторяется введенное имя для проверки правильности его набора на клавиатуре.

```
> nslookup
Default Server: DEPT-GW.CS.YALE.EDU
Address: 128.36.0.36

> bulldog.cs.yale.edu
Server: DEPT-GW.CS.YALE.EDU
Address: 128.36.0.36

Name: bulldog.cs.yale.edu
Address: 128.36.0.3
```

2.4.9 Диалог при пересылке файла

Далее можно использовать FTP для копирования файла *chapter1* из каталога *book* хоста *plum.yale.edu* на локальный хост. Начинающиеся цифрами строки — это сообщения от сервера пересылки файлов. Команда *cd* (change directory — изменить текущий каталог) служит для перехода к каталогу *book* на удаленном хосте. Команда *get* (получить) используется для копирования файла.

```
> ftp plum.yale.edu
Connected to plum.yale.edu
220 plum FTP server (SunOS 4.1) ready.

Name : icarus
331: Password required for icarus

Password :
230 User icarus logged in.

ftp> cd book
250 CWD command successful.

ftp> get chapter1
200 PORT command successful.
150 ASCII data connection for chapter1 (130.132.23.16,3330) (32303 bytes).
226 ASCII Transfer compete.
32303 bytes recived in 0.95 seconds (33 Kbytes/s)

ftp> quit
221 Goodbye.
```

Пересылка файла упрощается в приложении для настольного компьютера с графическим пользовательским интерфейсом (Graphical User Interface — GUI). На рис. 2.2 показано копирование файла на персональный компьютер в приложении *Chameleon* компании Netmanage. Перечисленные справа файлы находятся на сервере Unix. Один из них (с именем Index-byname) выбран для копирования. После копирования ему будет присвоено имя index.txt, которое указано в левом окне. Операция копирования запускается щелчком мыши на командной кнопке со стрелкой или при перетаскивании значка файла.

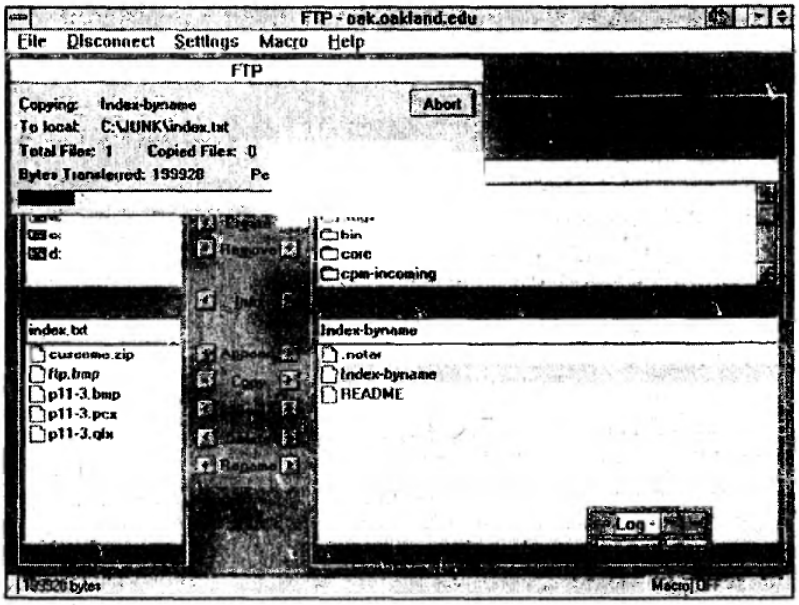

Рис. 2.2. Пересылка файла через приложение для настольного компьютера

2.4.10 WWW

Можно копировать файлы и с серверов WWW, не задумываясь об их реальном размещении. На рис. 2.3 показан экран *браузера Netscape Navigator.* Новые документы извлекаются щелчком мыши на одной из выделенных фраз. Далее их можно сохранить на локальном диске через меню File/Save.

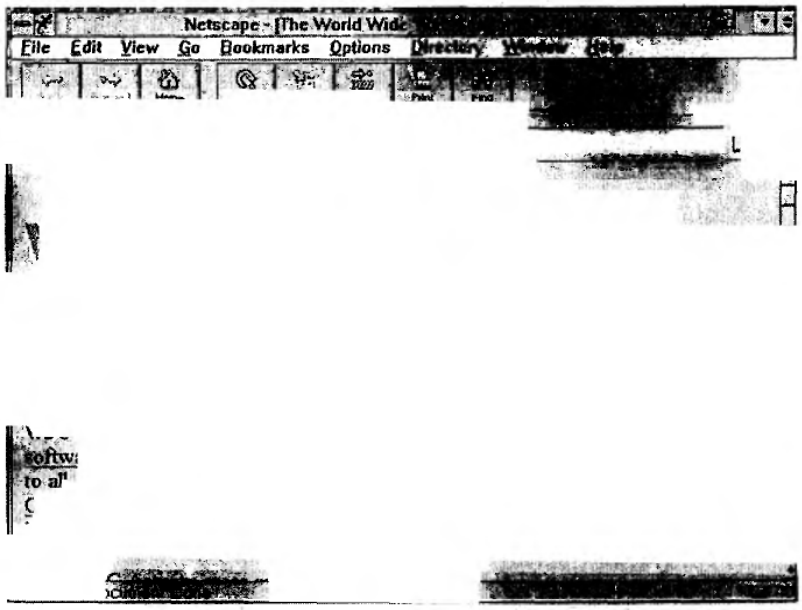

Рис. 2.3. Использование браузера *Netscape*

2.4.11 Новости

На рис. 2.4 представлен экран *Chameleon* для работы с новостями. На нем перечислены группы новостей по темам научных исследований.

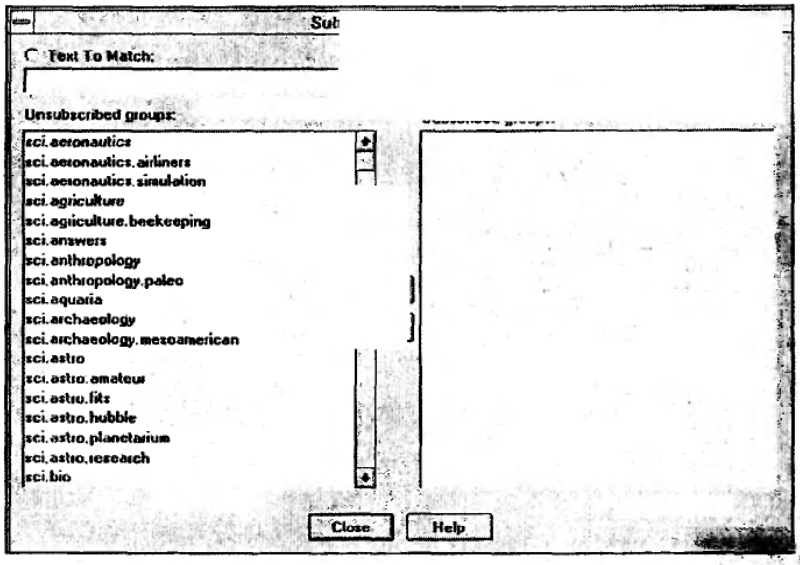

Рис. 2.4. Группы новостей по научной тематике

2.4.12 Диалог для доступа к файлу

Рассмотрим последний диалог с пользователем. В этом примере используется компьютер с дисковой операционной системой (Disk Operating System — DOS), подключенный к сети TCP/IP. Мы переключимся на устройство *d:* локального хоста и просмотрим содержимое корневого каталога:

```
d:
d:\> dir
  Volume in drive D is SERVER
  Directory of D:\

.          <DIR>              10-25-95      8:03a
..         <DIR>              10-25-95      8:03a
ALTBWPM                 711    2-18-95     12:53p
EGA512     FRS         3584    9-16-94      3:57p
WPRINT1               344392   11-05-95     13:28p
README     WPD         5492    9-16-95      3:57p
SPELL      EXE        40448    9-16-95      3:55p
WP         EXE       252416   11-15-95      4:51p
. . .
```

К этому примеру даже не требуется особых пояснений: файлы, которые отмечены как находящиеся на локальном диске *d:*, реально читаются с удаленного сервера NFS.

Глава
3

Архитектура TCP/IP

3.1 Введение

Протоколы TCP/IP разработаны для сетевого окружения, которое было мало распространено в 70-х гг., но сегодня стало нормой. Эти протоколы позволяют соединять оборудование различных производителей и способны работать через различные типы носителей или сред и связи данных. Они позволили объединить сети в единую сеть *интернет*, все пользователи которой имеют доступ к набору базовых служб.

Более того, спонсировавшие разработку TCP/IP научные, военные и правительственные организации хотели получить возможность подключения к интернету новых сетей без изменения служб уже существующих в интернете сетей.

Все эти требования нашли отражение в архитектуре TCP/IP. Требования независимости от носителей и расширения за счет подключения новых сетей привели к решению о пересылке данных в интернет с разделением их на части и маршрутизацией каждой из этих частей как независимого элемента.

Такие возможности гарантируют надежную пересылку данных от хоста источника к хосту назначения. Вследствие этого разработчики маршрутизаторов направили свои усилия на повышение производительности и внедрение новых коммуникационных технологий.

Все это привело к прекрасной масштабируемости протоколов TCP/IP и возможности их применения на различных системах — от больших ЭВМ (mainframe) до настольных компьютеров. На практике полезный набор функциональных свойств сетевого управления маршрутизацией реализуется неинтеллектуальными устройствами, подобными мостам, мультиплексорам или коммутаторам.

3.2 Деление на уровни

Для достижения надежности обмена данными между компьютерами необходимо обеспечить выполнение нескольких операций:

- Пакетирование данных
- Определение путей (маршрутов) пересылки данных
- Пересылку данных по физическому носителю
- Регулировку скорости пересылки данных в соответствии с доступной полосой пропускания и возможностью приемника получать посланные ему данные
- Сборку полученных данных, чтобы в формируемой последовательности не было потерянных частей
- Проверку поступающих данных на наличие дублированных фрагментов
- Информирование отправителя о том, сколько данных было передано успешно
- Пересылку данных в нужное приложение
- Обработку ошибок и непредвиденных событий

В результате программное обеспечение для коммуникации получается достаточно сложным. Следование модели с разделением на уровни позволяет упростить объединение сходных функций в группы и реализовать разработку коммуникационного программного обеспечения по модульному принципу.

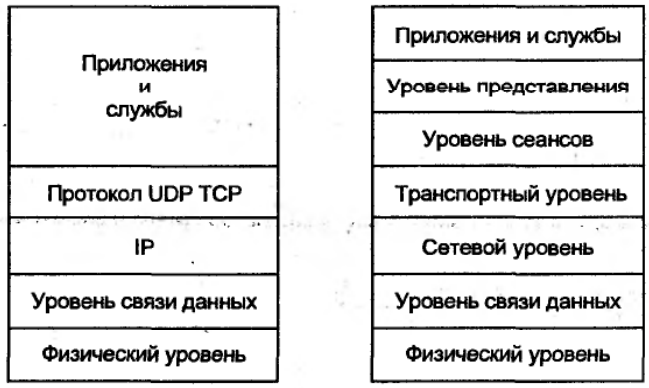

Рис. 3.1. Уровни TCP/IP и OSI

Специфика структуры протоколов TCP/IP определяется требованиями коммуникаций в научных и военных организациях. IP позволяет объединить различные типы сетей в интернет, а TCP несет ответственность за надежную пересылку данных.

Коммуникационная модель обмена данными OSI строго соответствует структуре TCP/IP. Уровни и терминология модели OSI стали стандартной частью коммуникационной структуры обмена данными.

На рис. 3.1 показаны уровни OSI и TCP/IP. Начнем их анализ с самого нижнего уровня (в TCP/IP формально не определены уровни сеанса и представления).

3.2.1 Физический уровень

Физический уровень (physical layer) имеет дело с физическими носителями, разъемами и сигналами для представления логических нулей и единиц. Например, адаптеры сетевого интерфейса Ethernet и Token-Ring и соединяющие их кабели реализуют функции физического уровня.

3.2.2 Уровень связи данных

Уровень связи данных (data link layer) организует данные в *кадры* (frame). Иногда его называют канальным уровнем. Как показано на рис. 3.2, каждый кадр имеет заголовок (header), содержащий адрес и управляющую информацию, а завершающая секция кадра (trailer) используется для исправления ошибок (иногда ее называют хвостом кадра. — *Прим. пер.*).

Заголовки кадров локальных сетей содержат физические адреса источника и назначения, которые идентифицируют передающую и принимающую интерфейсные карты локальной сети (сетевые адаптеры). Заголовки кадров, пересылаемых по региональной сети Frame Relay, содержат циклические идентификаторы в специальном адресном поле.

Вызов *соединения* (связи) в локальной сети, т.е. создание некоторой линии между конечными точками передачи данных, и аналогичные возможности в региональных сетях описываются протоколами уровня связи данных.

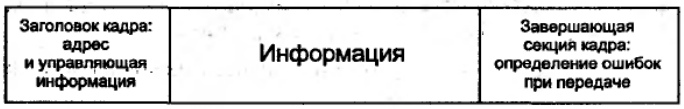

Рис. 3.2. Формат кадра

3.2.3 Сетевой уровень

Функции сетевого уровня (network layer) выполняет протокол IP, который осуществляет маршрутизацию данных между системами. Данные могут следовать по одному пути или использовать несколько различных путей при перемещении в интернете. Данные пересылаются в элементах, называемых *датаграммами* (datagram).

Как показано на рис. 3.3, датаграмма имеет заголовок IP, содержащий информацию об адресации для третьего уровня. Маршрутизатор проверяет адрес назначения для пересылки датаграммы в нужное место.



Рис. 3.3. Датаграмма IP

Уровень IP называется "*без создания соединения*", поскольку каждая датаграмма маршрутизируется независимо и протокол IP не гарантирует тот же порядок получения датаграмм, как при их отправке. IP маршрутизирует трафик без учета взаимодействий между приложениями, которым принадлежат конкретные датаграммы.

3.2.4 Транспортный уровень (TCP)

Протокол TCP выполняет функции транспортного уровня (transport layer) и обеспечивает надежную службу пересылки данных для приложений. В TCP/IP встроен специальный механизм, гарантирующий пересылку данных без ошибок и пропусков и в той последовательности, в которой они были отправлены.

Приложения, например пересылки файлов, передают данные в TCP, который добавляет к ним заголовок и формирует элемент, называемый *сегментом* (segment).

TCP отсылает сегменты в IP, в котором производится маршрутизация данных в заданное место. На другой стороне соединения TCP предполагает получение тех же сегментов данных от IP, определяет приложение, которому направлены эти данные, и передает их приложению в том порядке, в котором они были отправлены.

3.2.5 Транспортный уровень (UDP)

Приложение может послать другому приложению независимое сообщение с помощью протокола UDP, который добавляет к сообщению заголовок и формирует элемент, называемый *датаграммой UDP* или *сообщением UDP*.

UDP передает исходящие сообщения в IP и предполагает на другой стороне получение входящих сообщений от IP. Далее UDP определяет приложение, которому направлены данные.

UDP реализует коммуникационную службу без создания соединения, которая часто используется для просмотра содержимого простых баз данных.

3.2.6 Службы для приложений

Как уже отмечалось в главе 2, набор протоколов TCP/IP включает стандартные службы для приложений, такие как доступ с терминала, пересылка файлов, обращение к файловым серверам NFS, электронная почта, сетевые новости, WWW и просмотр адресов в DNS.

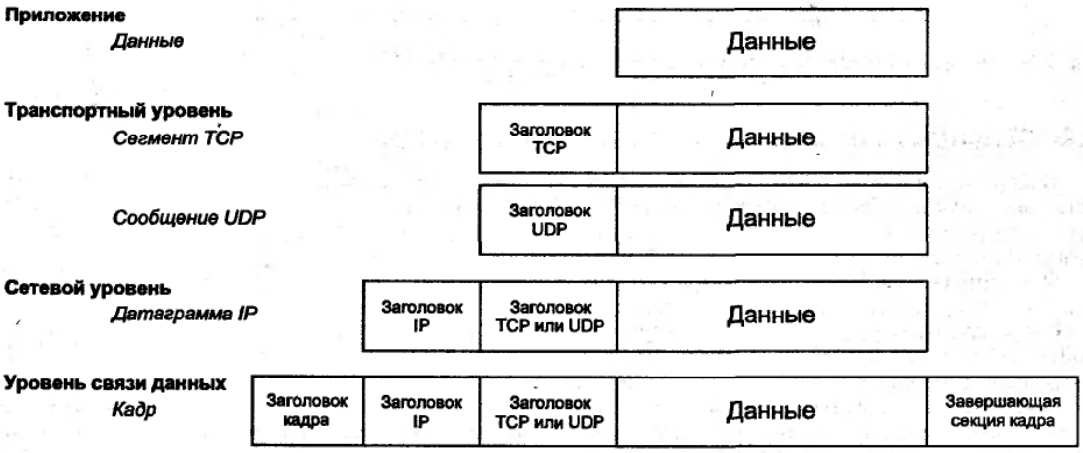

Рис. 3.4. Пакетирование данных перед пересылкой по сети

3.2.7 Пакетирование данных

На рис. 3.4 показано, как пакетируются прикладные данные перед пересылкой по сети. Основным термином для объединения информации с заголовком соответствующего сетевого уровня является *элемент данных протокола* (Protocol Data Unit — PDU). Например, сегмент TCP является PDU транспортного уровня, а датаграмма IP — PDU сетевого уровня.

3.3 Обзор протоколов

На рис. 3.5 представлено соотношение между отдельными компонентами набора протоколов TCP/IP.

Хотя текстовые пользовательские интерфейсы для пересылки файлов, доступа с терминала, работы с новостями или запросами к DNS для определения адреса по имени формально не стандартизованы, многие разработчики копируют интерфейс конечного пользователя из BSD Unix. Работающие в режиме текстовых команд пользователи находят, что пользовательский интерфейс не слишком отличается в разных системах.

Для настольных систем Windows и Macintosh существует множество графических пользовательских интерфейсов. Хотя они и отличаются в деталях, но в целом следуют стандартным соглашениям операционных систем и обычно могут использоваться без специального изучения.

Клиенты WWW, сетевых новостей, пересылки файлов (FTP), почты (SMTP) и терминального доступа (*telnet*) могут взаимодействовать со своими серверами через соединения TCP. Большинство клиентов NFS обмениваются со своими серверами сообщениями UDP, хотя некоторые реализации NFS предполагают использование как UDP, так и TCP.

Просмотр каталогов DNS основан на сообщениях UDP. Станции управления SNMP извлекают сведения из сетевых устройств с помощью сообщений UDP.

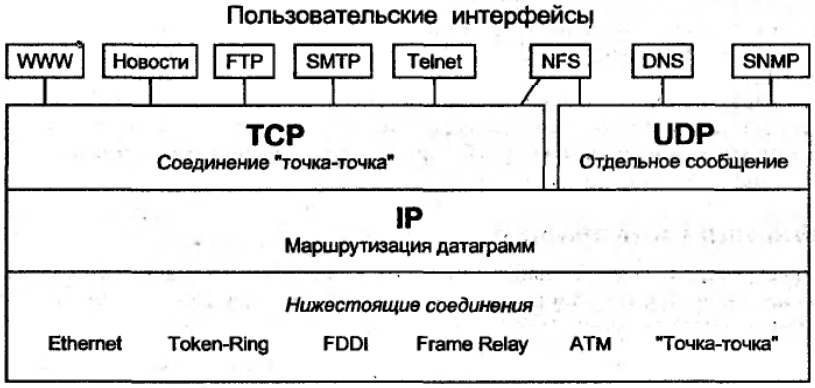

Рис. 3.5. Соотношение между компонентами набора протоколов TCP/IP

3.4 Маршрутизаторы и топология сети

Набор протоколов TCP/IP может использоваться как в независимых локальных или региональных сетях, так и для их объединения в общие сети интернета. Любой хост с TCP/IP может взаимодействовать с другим хостом через локальную сеть, соединение "точка-точка" или через региональную сеть с пакетированием информации (см. рис. 3.6).

Объединение сетей в интернет предполагает использование *маршрутизаторов IP*. На рис. 3.7 показана сеть интернет, созданная из независимых сетей, соединенных маршрутизаторами IP.

Современные маршрутизаторы обеспечивают работу нескольких аппаратных интерфейсов, которые можно комбинировать для применения с конкретной сетевой топологией: Ethernet, Token-Ring, FDDI, синхронные соединения "точка-точка", Frame Relay и т.д.

Сети интернет можно построить с помощью самых разнообразных топологий. Однако если интернет будет иметь логически связанную структуру, маршрутизаторы смогут выполнять свою работу более эффективно и быстрее реагировать на неисправности в отдельных сегментах сети, перенаправляя датаграммы по функционирующим путям. Простая для понимания логическая структура поможет сетевым администраторам в диагностике, локализации и ликвидации сетевых неисправностей.

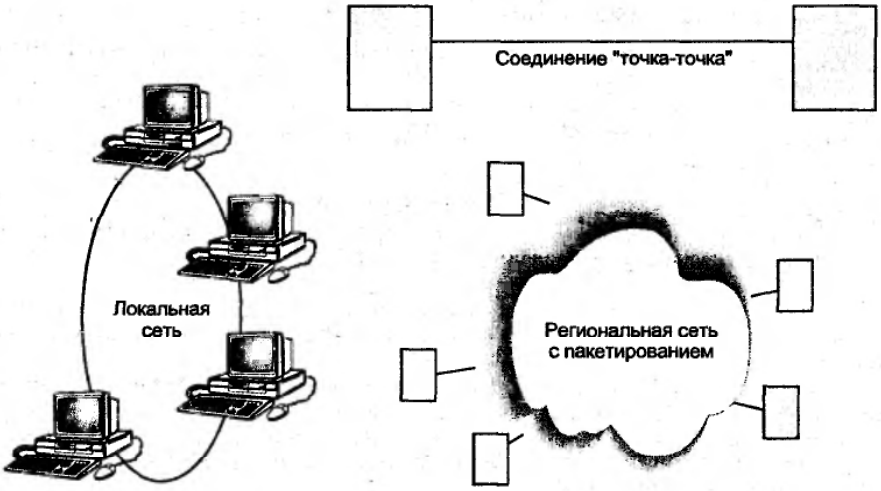

Рис. 3.6. Независимые друг от друга сети

Обширный и основанный на конкуренции рынок маршрутизаторов IP помог развитию архитектуры TCP/IP. Разработчики маршрутизаторов быстро реализовывали новые топологии локальных и региональных сетей, предоставляя своим клиентам возможность выбора среди аналогичных устройств. За последние несколько лет существенно снизилось соотношение цены маршрутизаторов к их производительности.

3.5 Маршрутизация в IP

Программное обеспечение IP работает на хостах и маршрутизаторах IP. Если пункт назначения датаграммы не находится в том же самом сетевом сегменте, что и ее источник, то протокол IP локального хоста направляет такую датаграмму на локальный маршрутизатор. Если последний не подключен непосредственно к узлу назначения датаграммы, то она будет передана другому маршрутизатору. Этот процесс продолжается до тех пор, пока датаграмма не достигнет заданного пункта назначения.

Маршрутизатор IP определяет местоположение *удаленного* узла по таблице маршрутизации (routing table), содержащей сведения о ближайших маршрутизаторах, которым должен быть направлен трафик датаграмм для достижения конечной точки в сети.

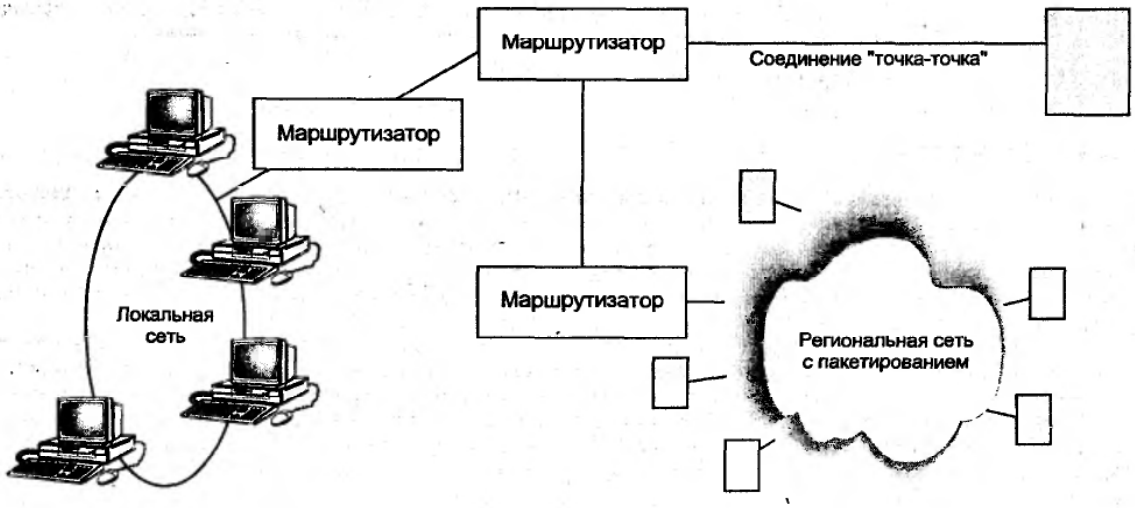

Рис. 3.7. Объединение независимых сетей маршрутизаторами

3.5.1 Протоколы маршрутизации

В небольшой статической сети интернет таблицы маршрутизации могут заполняться и обслуживаться вручную. В больших сетях интернет корректность таблиц маршрутизации поддерживается самими устройствами посредством обмена информацией между маршрутизаторами. Маршрутизаторы могут динамически определять следующие события:

- Добавление к интернету новой сети
- Разрушение пути к пункту назначения или невозможность его достижения за заданное время
- Добавление в интернет нового маршрутизатора, который может обеспечить более короткий путь к месту назначения

Не существует единого стандарта для обмена информацией между маршрутизаторами. Свобода выбора между несколькими согласованными протоколами позволяет добиться наилучшей производительности в каждом конкретном случае.

Сетевая возможность по управлению организацией сети соответствует понятию "*автономной системы*" (Autonomous System — AS). Организация может выбрать любой из протоколов обмена информацией о маршрутизации, который связан с ее собственной автономной системой. Протоколы обмена информацией о маршрутизации применяются внутри автономных систем в виде *протокола внутреннего шлюза* (Interior Gateway Protocol — IGP).

Протокол информации о маршрутизации (Routing Information Protocol — RIP) стал одним из популярных стандартов IGP. Широкое распространение этого протокола связано с его простотой, однако новый протокол "*Сначала открывать самый короткий путь*" (Open Shortest Path First — OSPF) имеет еще более обширный набор полезных возможностей.

Хотя все маршрутизаторы поддерживают один или несколько стандартных протоколов, некоторые разработчики реализуют собственные лицензионные протоколы для обмена информацией между маршрутизаторами. Многие продукты для маршрутизаторов могут одновременно обрабатывать несколько протоколов.

3.6 Архитектура TCP

TCP реализуется на хостах. Наличие TCP на каждом конце соединения обеспечивает для доставки данных локального приложения следующие возможности:

- Точность
- Сохранение последовательности
- Полноту
- Исключение дублирования

Базовый механизм для реализации этих возможностей начинает использоваться с самого начала обмена данными. Передающая система TCP:

- Нумерует каждый сегмент
- Устанавливает таймер
- Пересылает сегмент

Принимающая система TCP сообщает своему партнеру, сколько данных было передано правильно, посредством выдачи подтверждения (acknowledgment — ACK). Если подтверждение пересылки сегмента не будет получено за заданный интервал времени, TCP производит повторную пересылку этого сегмента. Такая стратегия называется *повторной трансляцией с положительным подтверждением* (retransmission with positive acknowledgment). Иногда повторная пересылка приводит к дублированию доставленных на принимающую систему сегментов.

Принимающая система TCP должна расположить приходящие сегменты в правильном порядке и исключить дублирование. TCP передает данные в приложение в правильном порядке, без пропусков.

Поскольку одна сторона отправляет данные, а другая их принимает, TCP можно назвать *полнодуплексным* (full-duplex) протоколом: обе стороны соединения могут одновременно посылать и принимать данные (т.е. присутствуют два потока данных). TCP одновременно выполняет роли передатчика и приемника.

3.7 Архитектура UDP

UDP реализуется на хостах. Протокол не обеспечивает целостности доставки данных, поскольку эта функция возлагается на обменивающиеся данными приложения. Именно они проверяют целостность доставляемых данных.

Приложение, которое хочет переслать данные с помощью UDP, передает блок данных в UDP, а протокол UDP просто добавляет к ним заголовок и производит их пересылку по сети.

Участвующие во взаимодействии по UDP приложения могут посылать сообщения с пользовательскими датаграммами в любое время. Клиент и сервер, которые надстроены над UDP, несут ответственность за все взаимоотношения при обмене пользовательскими датаграммами.

3.8 Концепции безопасности

TCP/IP успешно обслуживает открытые соединения между компьютерами локальных, региональных, а также глобальных сетей. Однако к соединениям стали предъявляться требования обеспечения безопасности.

Базовые концепции безопасности в сетевом окружении подобны аналогичным концепциям для центрального хоста:

- Аутентификация пользователей
- Целостность (гарантия отсутствия изменения данных)
- Конфиденциальность (защита от нежелательного раскрытия информации)

3.8.1 Аутентификация

Важным аспектом компьютерной безопасности является выяснение "кто есть кто". Ранее это определяли идентификатор и пароль пользователя. Аналогичным образом в поле "From:" сообщения электронной почты идентифицируется отправитель. Однако пароль может быть перехвачен любителем подслушивать в сети, и сообщение электронной почты может быть фальсифицировано.

Если речь идет о пересылке серьезных транзакций в сетях TCP/IP, то требуется способ для надежной идентификации отправителя. Процесс проверки на авторство называется *аутентификацией* (authentication, дословно: проверка подлинности.— *Прим. пер.*).

3.8.2 Технология формирования резюме сообщения

Простой, но эффективный способ технологии аутентификации основан на *резюме сообщения* (message digest). Как показано на рис. 3.8, такое резюме вычисляется по содержимому сообщения с помощью секретного ключа. В настоящее время наиболее распространен алгоритм Message Digest 5 (MD5), который был разработан Рональдом Ривестом (см. RFC 1321).

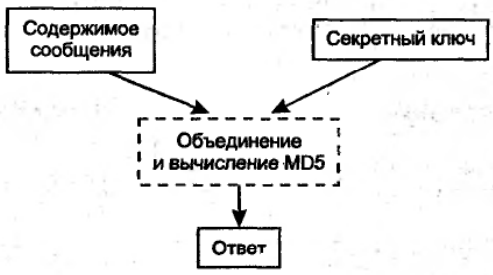

Рис. 3.8. Использование резюме сообщения.

Взаимное исследование (challenge handshake) иллюстрирует один из способов применения резюме сообщения. Как и при обычной аутентификации, пользователю присваивается пароль, регистрируемый на хосте. Однако этот пароль уже не пересылается по сети. Вместо этого настольная система выполняет вычисление по алгоритму MD5, используя пароль и секретный ключ (ключ шифрования.— *Прим. пер.*). Как показано на рис. 3.9:

1. Пользователь посылает на хост свой идентификатор.
2. Хост посылает пользователю сообщение со случайным содержимым.
3. Хост и настольная система пользователя выполняют вычисления по алгоритму MD5 для сообщения от хоста и секретного пароля пользователя.
4. Система пользователя отсылает ответ хосту.
5. Хост сравнивает ответ. Если ответ верен, пользователь аутентифицируется.

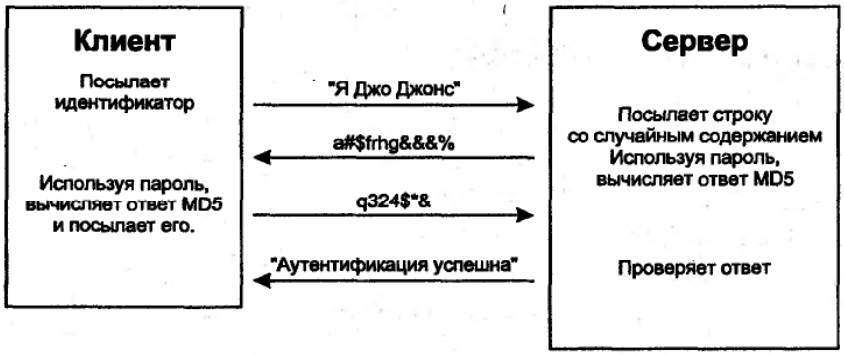

Рис. 3.9. Использование MD5 при взаимном исследовании

3.8.3 Целостность сообщения

MD5 и совместно используемые секретные ключи можно применять для определения изменений в данных при их пересылке по сети. Рассмотрим рис. 3.10:

1. Вычисление MD5 выполняется над данными с помощью секретного ключа.
2. Данные и полученное сообщение посылаются партнеру.
3. Партнер выполняет вычисление MD5 над полученными данными и известным секретным ключом.
4. Партнер сравнивает полученный результат с соответствующим резюме сообщения. При совпадении считается, что данные не изменились.

Отметим, что, не зная секретного ключа, подглядывающий за пересылаемыми данными злоумышленник не сможет фальсифицировать или изменить эти данные. Такой механизм применяется в системах защищенной электронной почты и безопасных от вторжения транзакциях клиент/сервер.

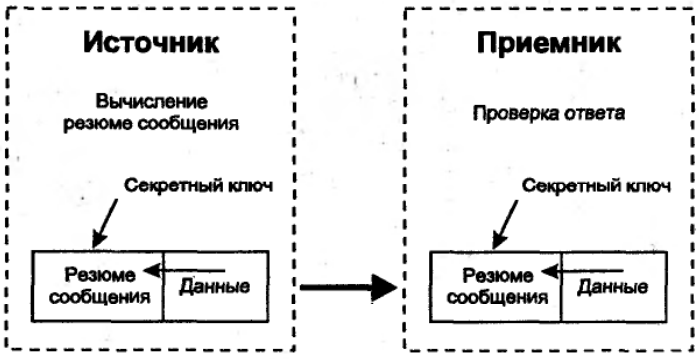

Рис. 3.10. Защита пересылаемых данных с помощью резюме сообщения, вычисленного по MD5

3.8.4 Конфиденциальность с помощью симметричного шифрования

Для предотвращения чтения и нежелательного использования пересылаемых данных злоумышленником (snooper) данные должны быть зашифрованы. Классическим способом является согласование секретных ключей между отправителем и получателем. Часто при пересылке добавляется резюме сообщения, и получатель может проверить, что данные получены в том виде, в котором они были отправлены. Как показано на рис. 3.11, после шифрования данные выглядят как бессмысленные строки.

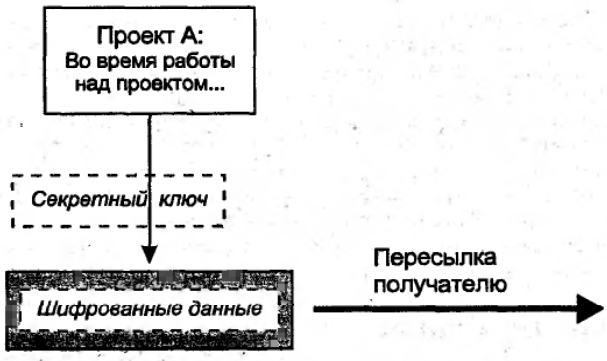

Рис. 3.11. Симметричное шифрование

Этот традиционный метод шифрования называется *симметричным*. Симметричное шифрование предполагает использование *одного и того же* ключа как для шифрования, так и для последующей расшифровки. Обе стороны знают ключ и должны сохранять его в тайне. Недостатки такого способа следующие:

- В целях большей безопасности каждой взаимодействующей паре приходится применять собственный секретный ключ.
- Изменение ключа связано с большими трудностями.

3.8.5 Асимметричный общедоступный ключ шифрования

Методы *асимметричного* шифрования известны достаточно давно (основные идеи были заложены в работах Диффи, Хеллмана и Меркля). При таком методе для шифрования и расшифровки используются *различные* ключи.

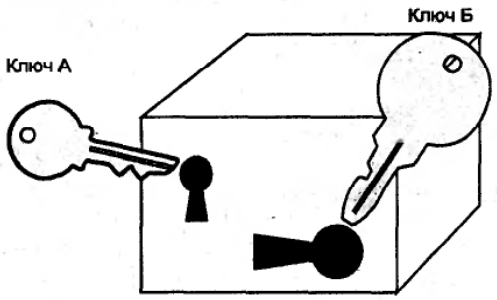

Рис. 3.12. Использование различных ключей для открытия и закрытия

Рассмотрим шкатулку с двумя различными ключами (А и Б), как показано на рис. 3.12:

- Если шкатулка закрывается ключом А, то открывается ключом Б.
- Если шкатулка закрывается ключом Б, то открывается ключом А.

Асимметричное шифрование называется также шифрованием по *общедоступным ключам* (public key), поскольку позволяет управлять ключами более согласованным способом. Ключ А может быть *общедоступным*. Его значение можно открыть для друзей или даже хранить в одном из доступных файлов.

- Все партнеры могут применять общедоступный ключ для шифрования пересылаемых данных.
- *Однако только вы будете знать личный ключ, и никто иной не сможет расшифровать посылаемые вам данные.*

Схема шифрования по общедоступным/личным ключам основана на том, что очень трудно подобрать два числа с большими значениями (количество проверок при этом выражается степенной функцией), чтобы получить значение ключей шифрования. Лучшим специалистам потребуется несколько месяцев, чтобы расшифровать данные с 129-разрядным ключом. Однако скорость работы компьютеров постоянно увеличивается, и вряд ли можно ожидать, что 1024-разрядные ключи останутся секретными по истечении еще нескольких лет.

Обслуживание общедоступных/личных ключей гораздо проще, чем симметричных. Однако нужна уверенность, что опубликованный общедоступный ключ "Jane Jone's Public Key" реально принадлежит нужной Джейн Джон, а не другому человеку с тем же именем.

К сожалению, известные сегодня методы асимметричного шифрования достаточно медленны, поэтому наиболее предпочтительна комбинация симметричных и асимметричных методов.

3.8.6 Комбинированное шифрование

Комбинированное шифрование реализуется следующим образом:

- Выбирается случайный симметричный ключ.
- По этому ключу шифруются данные.
- Случайный ключ шифруется с помощью общедоступного ключа шифрования получателя и включается в пересылаемое сообщение (это похоже на помещение нового случайного ключа в контейнер, который будет закрыт общедоступным ключом шифрования получателя).
- Получатель расшифровывает временный случайный ключ и далее использует его для расшифровки данных.

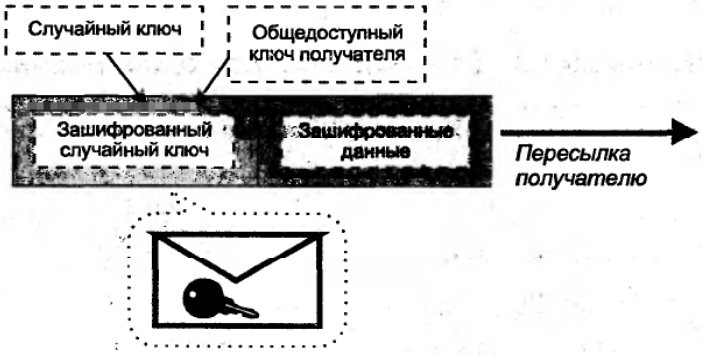

Рис. 3.13. Вложенный в зашифрованное сообщение ключ

Как показано на рис. 3.13, общедоступный ключ получателя обеспечивает защитную оболочку вокруг случайного ключа. Открыть эту оболочку сможет только получатель сообщения.

В следующих главах мы рассмотрим реализацию этих методов в приложениях и коммуникациях TCP/IP. Наиболее впечатляющий результат рассмотрен в главе 24, где описываются аутентификация и шифрование на уровне IP как для классической версии 4 протокола IP, так и для новой версии 6 — IP Next Generation (следующее поколение IP).

Глава

4

Технологии физического уровня и уровня связи данных

4.1 Введение

За последние несколько лет было предложено беспрецедентное количество новых технологий для локальных и региональных сетей, быстро утвердившихся на компьютерном рынке. Произошел огромный скачок от технологий носителей на витых парах и волоконной оптики — скачок, который никто не мог предвидеть. Сети ISDN, Frame Relay, T1, Fractional T1, T3, волоконно-оптические линии SONET, SMDS, новые кабельные соединители и технология ATM обеспечивают связь с обширными территориями, которая становится все быстрее и дешевле.

Организация IETF быстро реагирует на каждую новую технологию, создавая спецификацию для работы с IP в новом носителе и для других протоколов. Следом за ними разработчики маршрутизаторов создают аппаратные интерфейсы и драйверы, дающие пользователям возможность ощутить все преимущества новых технологий.

Работа IETF видна по большой серии RFC, в том числе:

The Point-to-Point Protocol (PPP) for the Transmission of Multiprotocol Datagrams over Point-to-Point Links (Протокол PPP для пересылки многопротокольных датаграмм по соединениям "точка-точка")

Standard for the transmission of IP datagrams over IEEE 802 networks (Стандарт для пересылки датаграмм IP по сетям IEEE 802)

Transmission of IP and ARP over FDDI Networks (Пересылка IP и ARP по сетям FDDI)

Classical IP and ARP over ATM (Классические пересылки IP и ARP по сетям ATM)

4.2 Функции физического уровня, управление доступом к физическому носителю и уровень связи данных

В этой главе мы рассмотрим работу IP поверх различных технологий нижнего уровня. Однако сначала обратимся к происходящим на этих уровнях событиям (см. рис. 4.1).

Физический уровень определяет кабели, соединители и электрические характеристики носителя. Правила представления логических единиц и нулей в носителе описываются на физическом уровне.

Пересылаемые данные для сохранения их смысла пакетируются в кадры (некоторые авторы называют такие элементы пакетами). Кадр переносит информацию по отдельной связи. Для достижения места (точки) назначения датаграмма IP может перемещаться по нескольких связям.

Описание формата кадра принадлежит уровню связи данных. Формат кадра различается в разных технологиях нижнего уровня, используемых для создания связи (например, линии T1, цепи Frame Relay или локальные сети Ethernet). Каждый кадр имеет заголовок, содержащий сведения, необходимые для его доставки по связи. Формат заголовка зависит от применяемой технологии.

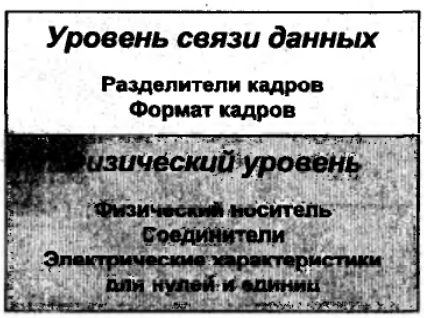

Рис. 4.1. Функции нижних уровней

4.3 Сетевые технологии

Все сетевые технологии можно разделить на четыре категории:

1. Связи "точка-точка" в региональных сетях
2. Локальные сети
3. Службы доставки пакетов региональных сетей
4. Службы коммутации ячеек

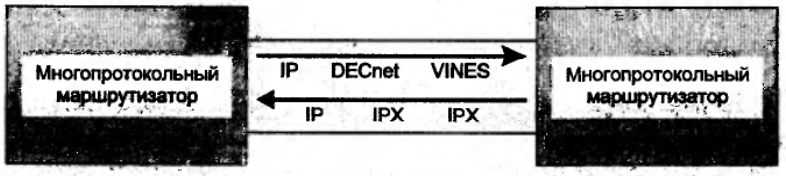

Рис. 4.2. Несколько протоколов совместно используют один носитель.

Для каждой технологии необходим механизм, который:

- идентифицирует место назначения, когда один интерфейс обслуживает несколько систем (например, интерфейс локальных сетей)
- выявляет ошибки при пересылке данных

На сегодняшний день *многопротокольным* окружением стали как локальные, так и региональные сети. Как показано на рис. 4.2, связи часто совместно используются несколькими протоколами (например, TCP/IP, Novell IPX/SPX, DECnet или Vines); эти же связи применяются при перенаправлении трафика. Многопротокольные хосты и маршрутизаторы должны иметь возможность сортировки различных типов трафика и, следовательно, иметь механизмы для:

- идентификации типа протокола для PDU, используемого в каждом кадре.

Определение типа протокола представляется не очень сложной работой. Нужно иметь стандартный список протоколов, присвоить каждому из них уникальный номер и поместить такой номер в одно из полей заголовка кадра.

Однако проблема в том, что для описания типа протокола существует несколько стандартов, каждый из которых определяет собственный подход к идентификации полей и присвоенных протоколам номеров. В этой главе мы познакомимся с различными форматами, используемыми в наиболее распространенных технологиях пересылки данных.

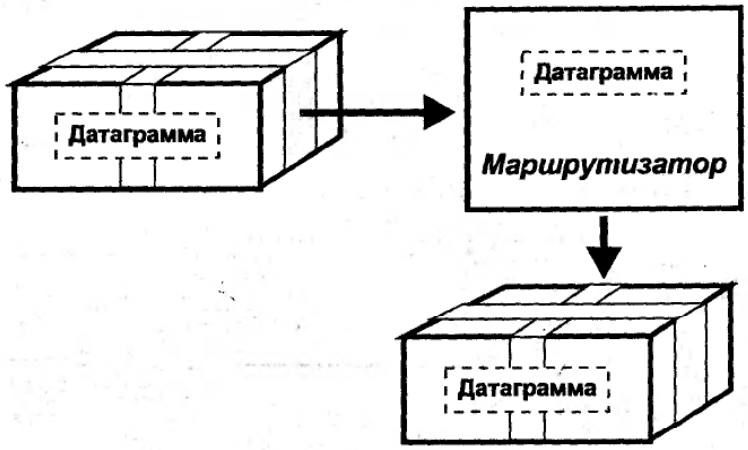

Рис. 4.3. Извлечение данных из пакета

4.4 Извлечение данных из пакетов

В соревнованиях по многоборью спортсмены сначала преодолевают один из участков вплавь, далее пересаживаются на велосипед и т.д. Протокол IP работает подобным же образом: датаграмма перемещается из одной среды передачи в другую (из одного носителя в другой), пока не достигнет пункта назначения.

Перед тем как датаграмма будет передана по сетевой связи, она заключается в соответствующий этой связи кадр. После получения кадра маршрутизатор (см. рис. 4.3):

- удаляет обрамление кадра и извлекает датаграмму
- анализирует IP-адрес назначения датаграммы и выбирает следующий носитель для дальнейшей пересылки
- заключает датаграмму в новый кадр (пакетирует ее) и передает по следующей связи, направляя ее далее по маршруту

Перейдем к более детальному описанию и обсудим способы пакетирования данных для различного типа сетевых технологий. Начнем со связей "точка-точка".

4.5 Протоколы связей "точка-точка"

Датаграммы IP могут передаваться по связям "точка-точка" между парой хостов, хостом и маршрутизатором или парой маршрутизаторов. Протокол IP передает датаграмму посредством множества различных взаимодействий TCP или UDP по одиночной связи "точка-точка".

IP не знает и не заботится об идентичности приложения-источника и приложения-приемника. Каждый раз, когда IP сталкивается с исходящей датаграммой, он передает ее так, как это специфицировано в данном протоколе. Как иллюстрирует рис. 4.4, совместно использовать одну связь могут трафики различных взаимодействий клиент/сервер — примерно так же, как различные автомобили используют одну автостраду.

В настоящее время трафик IP, пересылаемый по связям "точка-точка", пакетируется несколькими различными способами:

- с использованием общепринятой версии протокола "точка-точка" HDLC
- через стандартный протокол Интернета PPP
- с использованием протокола SLIP

Понемногу реализации перемещаются в сторону стандарта Интернета PPP, который имеет множество разнообразных возможностей.

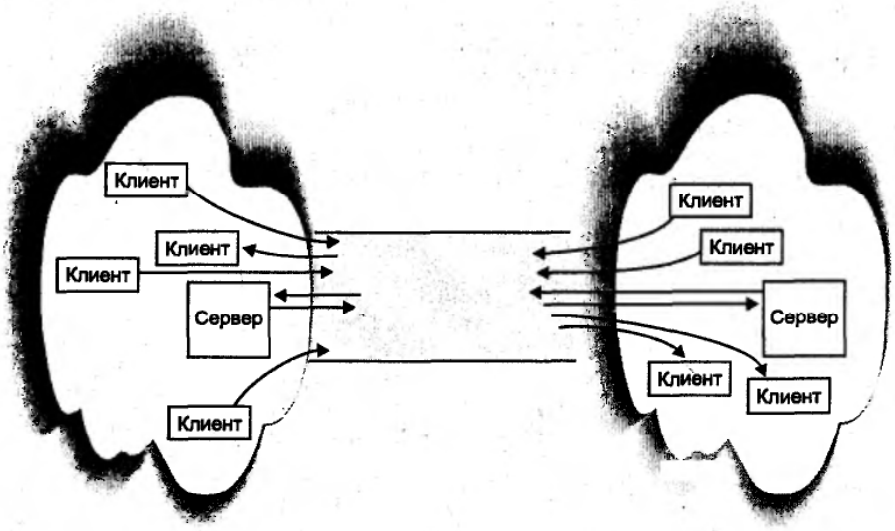

Рис. 4.4. Множество клиентов и серверов совместно используют одну сетевую связь.

4.6 HDLC

Протокол управления высокоуровневой связью данных (High-level Data Link Control — HDLC) является международным стандартом для связи "точка-точка" начиная с 60-х годов. HDLC пересылает серию данных как синхронизированный по времени поток бит, разделенный на кадры. Каждый кадр отделяется специальным шаблоном (*флажком*):

```
0 1 1 1 1 1 1 0
```

Для распознавания этого шаблона необходимо исключить его возникновение в пользовательских данных. Для этого после пересылки флажка открытия кадра передающая аппаратура вставляет нули после каждых пяти последовательных единиц в пользовательских данных. Такой способ называется *вставкой нулевого бита* (zero-bit insertion) или *набивкой битов* (bit-stuffing).

После выявления начала кадра приемник на другом конце связи выполняет удаление всех нулей после каждых пяти последовательных единиц внутри кадра (это делается на аппаратном уровне).

На рис. 4.5 показаны данные до и после вставки дополнительных битов.



Рис. 4.5. Вставка нулевого бита в HDLC

4.6.1 Формат кадра HDLC

Использование шаблона в протоколе HDLC влияет на всю структуру формата кадра. На рис. 4.6 показан информационный кадр HDLC, имеющий заголовок, данные и завершающую секцию, которая содержит *контрольную последовательность кадра* (Frame Check Sequence — FCS). Октет шаблона применяется как разделитель в начале и в конце кадра.

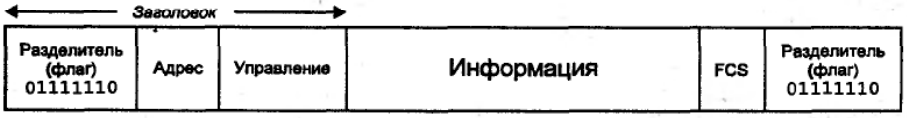

Рис. 4.6. Формат кадра HDLC с разделителями

FCS создается в результате математического вычисления на основе содержимого кадра. Полученный результат называется *циклической избыточной суммой* (Cyclic Redundancy Check — CRC), и некоторые авторы используют для именования завершающей секции кадра название CRC, а не FCS. Аналогичные вычисления выполняются в точке назначения связи. Если полученный при этом результат не будет равен значению поля FCS, значит, некоторые биты кадра изменились при пересылке и кадр должен игнорироваться как содержащий ошибку.

Использование контрольной последовательности кадра — это очень полезная идея. Поле FCS можно обнаружить практически во всех кадрах локальных и региональных сетей.

Заголовок кадра HDLC имеет поле *адреса назначения* (destination address). Такое поле необходимо для *многоточечной* (multipoint) версии протокола HDLC (например, в протоколе Synchronous Data Link Control (SDLC) компании IBM), которая позволяет нескольким системам совместно использовать одну линию. Каждой системе присваивается собственный адрес, а трафик этой системы перенаправляется в соответствии с адресом в заголовке кадра.

IP не использует технологию многоточечной линии связи, и передаваемые в кадрах HDLC датаграммы IP имеют своим адресом двоичное значение 11111111 (шестнадцатеричное X'FF), которое называется широковещательным адресом (broadcast), определяющим пересылку кадра на *все станции* сети. (Далее в книге для записи шестнадцатеричных чисел используется формат X'N, где X указывает на шестнадцатеричное число, N — представляет само число, а "'" — разделяет два поля такой записи.— *Прим. пер.*)

Заголовок кадра HDLC имеет поле *управления* (control). Некоторые протоколы связи помещают в это поле номера пересылаемых кадров или номера кадров для подтверждения их получения. Примерами могут служить протоколы SDLC и LAPB, использующие поле управления для нумерации, подтверждения приема и повторной трансляции кадров. Такие протоколы выполняют повторную пересылку тех кадров, для которых не получено подтверждение их получения приемником за заданный интервал времени.

Кадры, переносящие датаграммы IP, как и кадры для пересылки данных других протоколов, например IPX или DECnet, не требуют нумерации и подтверждения. Для IP и других похожих протоколов в управляющем поле записывается значение X'03, указывающее на *нечисловой информационный кадр* (Unnumbered Information frame) протокола HDLC.

Таким образом, датаграммы IP в кадрах HDLC имеют формат, представленный на рис. 4.7.



Рис. 4.7. Формат кадра HDLC, передающего датаграмму IP

Обобщив, можно отметить, что при пересылке датаграмм IP в кадрах HDLC:

- Используется широковещательный адрес X'FF.
- Управляющее поле имеет значение X'03 — нечисловой информационный кадр.

4.6.2 Недостатки HDLC

То, что HDLC является стандартом, еще *не означает* успешного взаимодействия друг с другом связей "точка-точка" между различными реализациями интерфейсов HDLC.

В HDLC определено множество дополнительных и необязательных возможностей, что приводит к различным "стандартным" реализациям HDLC. Еще более запутывает ситуацию предоставление многими разработчиками собственных версий HDLC для интерфейсов "точка-точка".

В результате долгое время не было единого стандарта для коммуникаций "точка-точка", что существенно затрудняло использование оборудования от различных производителей.

Разработка HDLC была выполнена до появления многопротокольных сетей. Однако сегодня многие линии "точка-точка" служат для пересылки трафика от различных протоколов, что приводит к дополнительным проблемам.

Решение этих вопросов поручено комитету IETF.

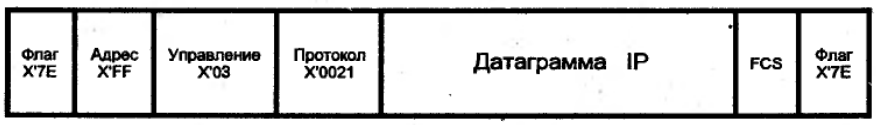

Рис. 4.8. Формат кадра PPP, переносящего датаграмму IP

4.7 Протокол PPP

Рабочая группа IETF предложила решение на основе *протокола "точка-точка"* (Point-to-Point Protocol — PPP). *PPP* может использоваться в любой полнодуплексной цепи — синхронной с пересылкой битов или асинхронной (старт/стоп) с пересылкой байтов. Этот протокол пригоден для медленных последовательных линий связи, быстрых выделенных линий, ISDN или волоконно-оптических каналов SONET. PPP был разработан для пересылки PDU различных протоколов — IP, IPX, DECnet, ISO и т.д. Кроме того, PPP обеспечивает пересылку данных через сетевые мосты.

PPP содержит несколько подпротоколов. Например:

- *Протокол управления связью* (Link Control Protocol) служит для установки, проверки, конфигурирования и закрытия сетевой связи.
- *Протокол управления сетью* (Network Control Protocol) предназначен для инициализации, конфигурирования и завершения использования отдельного сетевого протокола. Индивидуальный протокол Network Control Protocol определен для IP, IPX, DECnet, ISO и т.д.

Типичный сценарий PPP выполняется следующим образом:

- Начинающая соединение по PPP система посылает кадр *Link Control*. Ее партнер отвечает дополнительным кадром Link Control, устанавливая параметры связи.
- Проводится обмен кадрами *Network Control Protocol* для выбора и конфигурирования используемых протоколов сетевого уровня.
- Данные выбранного протокола пересылаются по связи в кадрах PPP. Каждый кадр имеет поле заголовка, идентифицирующее тип протокола для содержащихся в кадре данных.
- Для завершения связи применяется обмен кадрами Link Control и Network Control.

Заголовок кадра PPP похож на заголовок HDLC, но содержит одно дополнительное поле для идентификации протокола следующего уровня. На рис. 4.8 показан формат кадра PPP с датаграммой IP. Адресное поле имеет значение X'FF (широковещательная рассылка), а управляющее поле — X'03 (нечисловая информация). Дополнительное *поле протокола* (protocol field) имеет значение X'00-21, что соответствует пересылке датаграмм IP. Номера для протоколов определены в документе RFC *Assigned Numbers* (присвоенные номера) от IANA.

4.7.1 Сжатие в PPP

Может показаться не очень разумным включение одних и тех же октетов адреса и управления в каждый кадр. Партнеры на каждом конце связи PPP могут работать в режиме *сжатия* (compression) для исключения этих полей.

Значения в поле протокола указывают, является ли содержимое кадра сообщением Link Control или Network Control, либо полезной информацией (например, датаграммой IP). При установке связи по PPP поле протокола имеет длину 16 бит, но далее при пересылке полезной информации его длина может быть сокращена до 8 бит. Следовательно, датаграмма может быть пакетирована более эффективно (см. рис. 4.9).

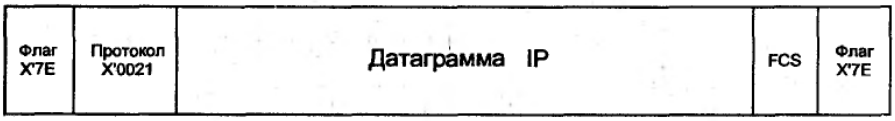

Рис. 4.9. Кадр PPP в сжатом формате

Еще одной возможностью в PPP является сжатие методом Вана Джекобсона, позволяющее исключить лишние байты, пересылаемые в сеансе TCP. Заголовки IP и TCP вместе имеют длину от 40 байт и более. Сжатие методом Вана Джекобсона уменьшает типичную 40-байтовую комбинацию до 3, 4 или 5 байт. Для этого оба партнера должны сохранять первоначальные заголовки. При изменениях во время сеанса TCP будут пересылаться только измененные значения в заголовках. Поскольку большая часть используемой в заголовках информации имеет статическую природу, объем пересылаемых изменений будет незначителен.

4.8 Дополнительные возможности PPP

Рабочая группа по PPP решила и несколько дополнительных проблем, которые могут возникнуть при использовании связей "точка-точка".

4.8.1 Аутентификация

Протокол PPP часто используется для удаленных коммуникаций и связи мобильного пользователя по коммутируемым соединениям. Коммутируемые соединения (dialup connection) иногда применяются для связи локальных сетей подразделений компании с сетью главного офиса.

Перед тем как разрешить внешней системе соединиться с сетью по коммутируемому соединению, следует провести аутентификацию такой системы. В настоящее время PPP поддерживает два способа аутентификации:

- Простой *протокол аутентификации по паролю* (Password Authentication Protocol — PAP). В этом случае используются идентификатор пользователя и его пароль, которые вкладываются в кадры, пересылаемые по связи в процессе ее создания.
- *Протокол аутентификации по взаимной проверке* (Challenge Handshake Authentication Protocol — CHAP).

Метод взаимной проверки был рассмотрен в главе 3 и не представляет особых трудностей при изучении. Как показано на рис. 4.10:

1. По связи открытым текстом пересылается имя пользователя.
2. Удаленный партнер отвечает случайным проверочным сообщением.
3. Локальная система вычисляет резюме сообщения (используя содержание сообщения и пароль пользователя как входную информацию) и отсылает обратно ответ.
4. Удаленный партнер на основе пароля выполняет те же самые вычисления и сравнивает полученные результаты.

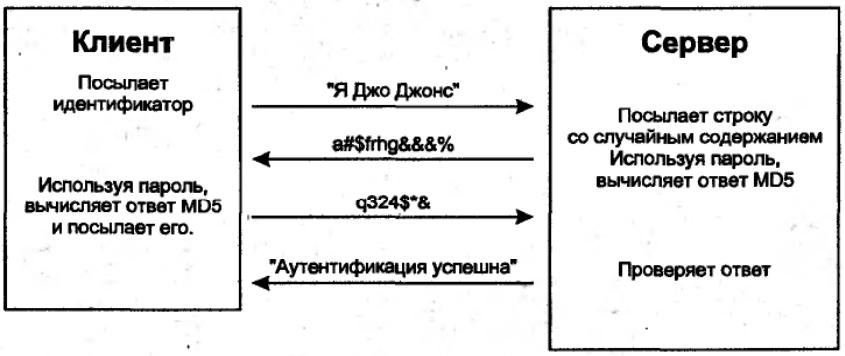

Рис. 4.10. Взаимная проверка в PPP

Подглядывающий за работой сети злоумышленник будет видеть при каждом установлении соединения различные бессмысленные байты. При использовании 16-байтового пароля практически невозможно определить его, подглядывая за сетевым соединением.

4.8.2 Автоматическое отслеживание качества связи

Часто PPP используется между двумя маршрутизаторами. Иногда со временем ухудшается качество соединения. Было бы неплохо заранее определять опасное состояние связи, для чего предусматривается автоматическое выполнение некоторых операций. Например, маршрутизатор может завершить коммутируемое соединение и провести повторный набор телефонного номера для воссоздания этого соединения. Если аналогичная проблема возникает в выделенной линии, то, возможно, придется временно переключить трафик на резервный канал связи.

В PPP реализован очень простой и эффективный способ проверки качества связи. При мониторинге состояния связи подсчитывается количество посланных и полученных кадров и октетов с учетом игнорируемых и ошибочных. Периодически полученный отчет передается на другой конец связи.

Такие сведения позволяют провести анализ произошедших в связи событий. Например, если за определенный интервал времени было послано 100 000 октетов, а партнер успешно получил только 50 000, то ясно, что со связью не все в порядке.

4.9 Протокол SLIP

Протокол интерфейса последовательной линии связи (Serial Line Interface Protocol — SLIP), созданный до PPP, обеспечивает уже устаревшие методы пересылки датаграмм IP по последовательной линии связи.

SLIP — наиболее примитивный из всех разработанных протоколов. Датаграмма IP просто пересылается по последовательной линии, байт за байтом. SLIP маркирует конец датаграммы специальным разделительным байтом: 11000000 (X'C0). Что же произойдет, когда такой байт появится в самой датаграмме? Передающая часть SLIP использует Ecs-последовательность, которую принимающая часть SLIP преобразует обратно в реальные данные:

C0 в данных —> DB DC

DB в данных —> DB DD

Обычно SLIP используется для соединения компьютеров PC, Macintosh и Unix с сетями IP по коммутируемым соединениям. Отметим, что SLIP не обеспечивает проверки FCS, передавая выявление ошибок на более высокие уровни. SLIP не обеспечивает пересылки никаких протоколов, кроме IP.

Протокол SLIP со сжатием (Compressed SLIP — CSLIP) является улучшенной версией протокола SLIP, производящей сжатие заголовков TCP/IP методом Вана Джекобсона и обеспечивающей лучшую производительность, чем SLIP.

SLIP можно использовать между хостами, хостом и маршрутизатором или между маршрутизаторами. На рис. 4.11 показан коммуникационный сервер, поддерживающий работу неинтеллектуальных терминалов ASCII и коммутируемые соединения с терминалами по SLIP. Для трафика SLIP данное устройство работает как маршрутизатор IP.

Наиболее привлекательным свойством SLIP является его широкое распространение. Наиболее неприятное свойство этого протокола состоит в том, что пользователь рабочей станции должен написать сценарий, который будет читать приглашение от коммуникационного сервера и посылать на сервер идентификатор пользователя, пароль и другую информацию в определенном месте выполнения диалога. PPP имеет большую функциональность и не требует использования сценариев, и вследствие этого понемногу вытесняет SLIP.

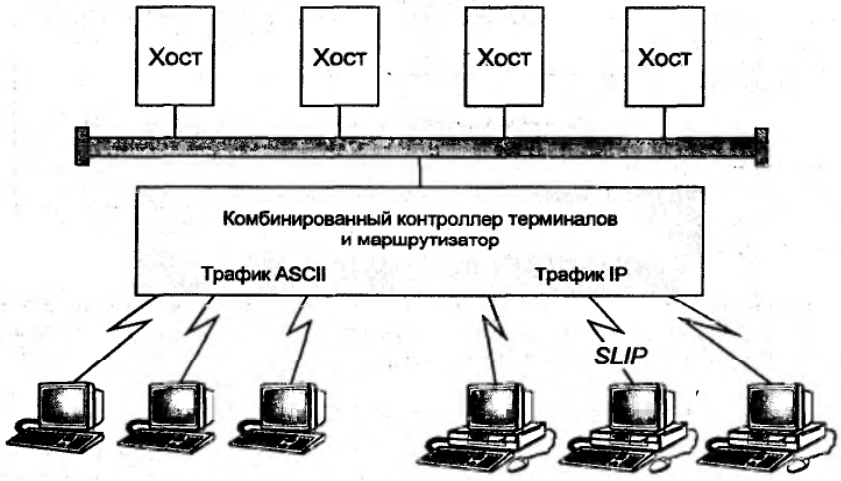

Рис. 4.11. Подключение терминала ASCII и соединения SLIP

4.10 Локальные сети

Рассмотрим, как IP и другие протоколы пакетируют кадры для пересылки по локальным сетям. Классическая локальная сеть предполагает следующие свойства:

- Станции совместно используют физический носитель.
- Существуют правила *управления доступом к носителю* (Media Access Control — MAC), определяющие моменты времени, когда станция может пересылать данные.
- Данные передаются в кадрах.

Начнем с сетей *Ethernet*, поскольку они предоставляют наиболее простой пример реализации локальных сетей.

4.11 DIX Ethernet

Локальные сети Ethernet первыми смогли передавать датаграммы IP. Компании Digital Equipment Corporation (DEC), Intel Corporation и Xerox Corporation совместно определили исходную спецификацию *DIX* Ethernet в 1980 г. Пересмотренная версия 2 этой спецификации появилась в 1982 г.

4.11.1 Носители для DIX Ethernet

Традиционным магистральным носителем для данной технологии является узкополосный коаксиальный кабель. Первоначально применялся жесткий полудюймовый кабель с сопротивлением 50 Ом. Позднее стал использоваться тонкий и более гибкий коаксиальный кабель в 1/4 дюйма, названый thinnet (тонкий сетевой) или *cheapernet* (дешевый сетевой), а еще позднее многие сети перешли на применение витых пар. Скорость передачи сигналов в 10 Мбит/с превалировала достаточно долгое время, однако сейчас доступна скорость пересылки в 100 Мбит/с. Сегодня DIX Ethernet может работать на широкополосных коаксиальных или волоконно-оптических кабелях.

Различия между вариантами Ethernet отражаются в следующей нотации:

```
[Скорость пересылки данных в Мбит/с][тип носителя][максимальная длина
кабельного сегмента в сотнях метров]
```

Таким образом, 10BASE5 означает узкополосный (baseband) коаксиальный кабель со скоростью передачи данных в 10 Мбит/с и максимальной длиной сегмента в 500 м. Спецификация для тонкого кабеля 10BASE2 означает узкополосный коаксиальный кабель со скоростью передачи данных в 10 Мбит/с и максимальной длиной сегмента в 200 м.

10BROAD36 должна означать широкополосный коаксиальный кабель со скоростью обмена в 10 Мбит/с и максимальной длиной сегмента в 3600 м. Обозначениями для витых пар и волоконной оптики являются 10BASET и 10BASEF соответственно, хотя это и не вполне согласуется с правилами нотации.

4.11.2 Протокол MAC для DIX Ethernet

DIX Ethernet использует простую процедуру MAC с очень длинным названием: *множественный доступ с контролем несущей и обнаружением конфликтов* (Carrier Sense Multiple Access with Collision Detection — CSMA/CD).

Интерфейс для работы с данными пакетирует информацию в кадры и прослушивает состояние носителя. Когда носитель свободен, интерфейс начинает пересылку данных (см. рис. 4.12).

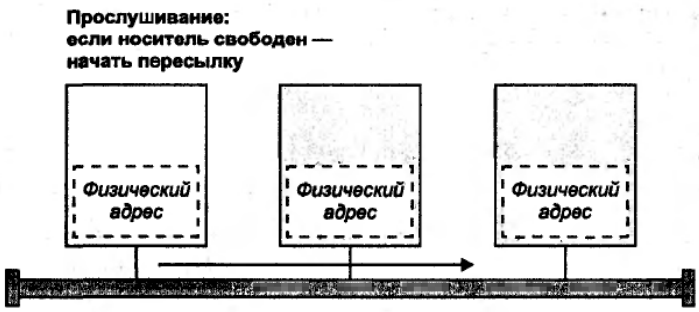

Рис. 4.12. Управление доступом к носителю в Ethernet

Заголовок кадра содержит физический адрес интерфейса назначения, часто называемый MAC-адресом. Система с указанным в кадре адресом принимает посланное сообщение и обрабатывает его. Если две или больше станций одновременно начинают пересылку данных, возникает конфликт. Пересылка отменяется на случайный для каждой станции промежуток времени и повторяется снова (каждая станция при этом будет начинать пересылку уже в разное время.— *Прим. пер.*).

4.11.3 MAC-кадры DIX Ethernet

Формат кадров DIX Ethernet с датаграммами IP показан на рис. 4.13.

Адреса источника и назначения имеют длину по 6 октетов (сами адреса администрируются IEEE). Значение *типа* в X'08-00 означает пересылку датаграмм IP.

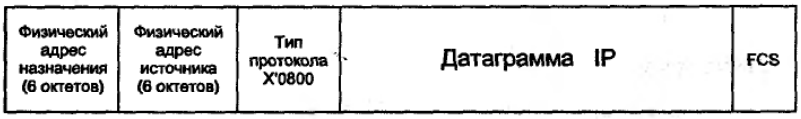

Рис. 4.13. Кадр DIX Ethernet с датаграммой IP

В Ethernet существуют значения типов и для других протоколов (см. документ IANA *Assigned Numbers*). Носитель может использоваться совместно несколькими протоколами, поскольку в каждом кадре Ethernet идентифицируется тип протокола, что позволяет станции назначения переслать полученную информацию нужной процедуре.

Для правильной работы протокола CSMA/CD требуются кадры длиной не менее 64 октетов. Следовательно, для очень коротких датаграмм нужно будет добавить незначащий заполнитель.

4.12 Сети по спецификации 802

После того как DIX Ethernet и другие технологии локальных сетей доказали на компьютерном рынке свою полезность, IEEE организовала *комитет 802* по разработке и публикации стандартов для технологий локальных сетей.

Стандарты из серии 802, объединяющие разработки разных производителей, были признаны ISO и повторно опубликованы как документы этой организации.

Стандарты 802 касаются физических носителей, управления доступа к носителю и форматов кадров для различных типов локальных сетей. Например:

- 802.3 описывает несколько измененную версию Ethernet.
- 802.4 специфицирует широковещательную локальную сеть с пересылкой маркера, разработанную для производственных помещений.
- 802.5 описывает технологию Token-Ring.
- 802.6 определяет подсети на основе двойной шины для распределенных очередей (Distributed Queue Dual Bus), использующихся в городских сетях (Metropolitan Area Network — MAN).

4.13 Заголовок LLC для 802.2

Отдельный стандарт 802.2 определяет заголовок *управления логической связью* (Logical Link Control — LLC), используемый во всех технологиях локальных сетей по спецификациям 802. Заголовок LLC выполняет две задачи:

- Для кадров OSI идентифицирует протоколы источника и назначения
- Содержит поля управления

Описание IEEE предполагает множество формальных правил, однако каждый из элементов достаточно прост.

Элементы для назначения и источника протоколов ISO каждого кадра описываются кодами *точки доступа к службе назначения* (Destination Service Access Point — DSAP) и кодами *точки доступа к службе источника* (Source Service Access Point — SSAP).

Значения DSAP/SSAP присваиваются протоколам ISO, но не протоколу IP или множеству других протоколов, используемых на практике. Для IP и других распространенных протоколов DSAP и SSAP

устанавливаются в значение X'AA, что означает наличие следующего далее другого заголовка, который и будет определять тип протокола для размещенных в кадре данных. Дополнительный заголовок называется подзаголовком *протокола доступа в подсети* (Subnetwork Access Protocol — *SNAP*).

Подзаголовок SNAP содержит вводную часть (introducer) со следующим далее старым знакомым — кодом типа Ethernet. Вводная часть имеет прекрасное название — *уникальный организационный идентификатор* (Organizationally Unique Identifier — OUI). Он определяет, кто несет ответственность за присвоение номеров протоколов.

Вводная часть (OUI) для кодов типа Ethernet (см. рис. 4.14) имеет значение X'00-00-00. Отдельный OUI со значением X'00-80-C2 служит для введения номеров различных протоколов мостов.

4.13.1 Спецификации 802.3 и 802.2

Стандарт 802.3 содержит описание носителя Ethernet, протокола доступа к носителю CSMA/CD и формата MAC-кадров. В соответствии со стандартами комитета 802 заголовок 802.2 должен включаться в MAC-кадр 802.3.

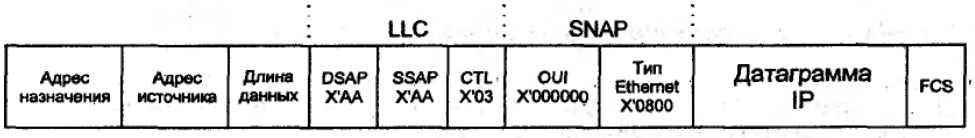

Рис. 4.14. Кадр 802.3 с заголовком LLC 802.2 и подзаголовком SNAP

На рис. 4.14 показан результат размещения датаграммы IP в кадре 802.3/802.2.

- Отметим, что в отличие от DIX Ethernet третье поле заголовка кадра 802.3 содержит сведения о *длине* информации, которая следует далее (за исключением заполнителя) вместо кода типа Ethernet. Длина определяется в 8 октетов полей LLC и SNAP. Далее мы увидим, что в заголовке IP размещается поле длины датаграммы, хотя для IP это значение является избыточным.
- DSAP и SSAP имеют значения X'AA, указывая на следующий далее подзаголовок SNAP.
- Поле управления содержит X'03, что означает нечисловую информацию — так же, как и в HDLC.
- Вводная часть поля SNAP содержит X'00-00-00, указывая на следующий далее тип Ethernet (который имеет значение X'08-00).

Другие протоколы (например, IPX или DECnet) имеют похожие кадры — нужно только подставить правильные значения для типов Ethernet.

Отметим, что 8 дополнительных октетов добавлены без каких-либо изменений в функциях IP. Именно поэтому многие реализации продолжают использовать старую спецификацию формата DIX Ethernet. Сетевые адаптеры Ethernet Network Interface Card (интерфейсные карты сети Ethernet) и их программные драйверы обычно поддерживают оба стандарта, а при конфигурировании пользователь может выбрать нужный вариант.

Часто термин *Ethernet* применяют как для старой DIX, так и для реализации IEEE 802.3/802.2. Иногда очень важно разделять эти понятия, поскольку системы, сконфигурированные для работы с DIX, не могут взаимодействовать с системами, сконфигурированными для 802.3/802.2.

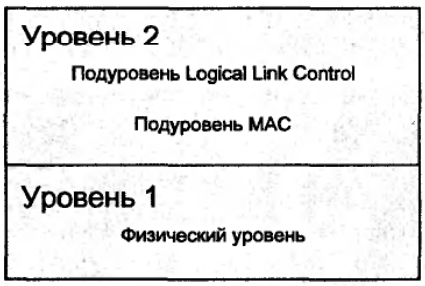

Рис. 4.15. Уровни для локальных сетей 802

4.14 Уровни в сетях 802

Ознакомимся со взглядом IEEE на сетевой мир. С появлением локальных сетей 802 IEEE разделил сетевой уровень 2 (уровень связи данных) на два подуровня (см. рис. 4.15).

Подуровень MAC обеспечивает правила доступа к носителю — слушать и пересылать для 802.3 или ждать маркера для 802.5. Этот же уровень определяет первую часть заголовка кадра, которая содержит физические адреса источника и назначения.

Подуровень Logical Link Control описывает формат заголовка LLC. Он же определяет достаточно сложные правила для коммуникаций в те моменты, когда производится нумерация, подтверждение пересылки, управление потоком и повторная пересылка кадров с использованием уровня связи данных. Связи, обеспечивающие такие возможности, называются связями *типа 2* (Type 2). Существует несколько протоколов, включая SDLC, LAPB или LAPD, которые выполняют коммуникации в локальных сетях с помощью связей типа 2.

Разумеется, датаграммы IP требуют только указания на подуровне Logical Link Control сведений о включении в кадр датаграммы IP. Обычно IP работает поверх протокола связи *типа 1*.

4.15 Другие технологии локальных сетей

По требованию IEEE локальные сети Token-Ring, token bus и FDDI *должны* включать заголовок LLC 802.2 и подзаголовок SNAP для пересылки протокола IP или иных протоколов с кодом типа Ethernet. Для них не существует укороченного формата.

Те же самые поля LLC и SNAP определяют вложенный протокол, как показано на рис. 4.14, а именно:

X'AA-AA-03-00-00-00 (*тип Ethernet*)

4.15.1 Конфигурация и носители для Token-Ring

Локальные сети Token-Ring были представлены компанией IBM, а позднее IEEE стандартизировал их как протокол 802.5. Станции в сети Token-Ring образуют физическое кольцо.

4.15.2 MAC для 802.5

Идея управления доступом к носителю (MAC) на основе маркера, или жетона, достаточно проста. Специальный кадр, называемый *маркером* (token), передается по кольцу от станции к станции. Когда станция получает такой маркер, она должна отправить данные дальше в течение ограниченного интервала времени. По завершении этого интервала удерживающая маркер станция обязана переслать его следующей станции.

Хотя основная идея не требует пояснений, для протокола пересылки маркера по кольцу нужны более сложные механизмы, чем для сети Ethernet. В частности, протокол для уровня MAC спецификации 802.5 включает процедуры связывания и разъединения кольца Token-Ring, идентификации ближайших соседей, выявления неисправной станции или потерянного маркера, предотвращения циклической пересылки данных и решения проблем с сигналами. Для различных функций 802.5 определяются разные заголовки MAC-уровня. Тип пересылающего данные протокола указывается через заголовки LLC и SNAP, размещенные за информационным полем маршрутизации (Routing Information Field) кадра Token-Ring.

4.15.3 802.4 Token Bus

Стандарт 802.4 описывает широкополосную шину локальной сети на основе коаксиального кабеля, в которой используется маркер для управления доступом к носителю. 802.4 является частью набора *протоколов автоматизации производства* (Manufacturing Automation Protocol), предлагаемых для использования в промышленных условиях. Сигналы в широкополосном коаксиальном кабеле не подвержены влиянию электромагнитных помех, связанных с промышленным производством. Использование протокола с пересылкой маркера позволяет достичь предсказуемого расписания доступа к локальной сети. Однако 802.4 никогда не имел широкого распространения.

4.15.4 FDDI

Волоконно-оптический интерфейс для распределенных данных (Fiber Distributed Data Interface — FDDI) со скоростью пересылки в 100 Мбит/с часто используется в локальных сетях для создания магистральных соединений, объединяющих низкоскоростные сегменты локальных сетей.

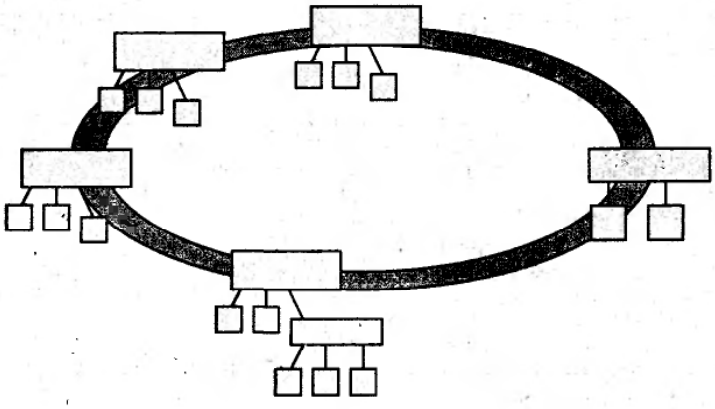

Рис. 4.16. Топология сетей FDDI

- FDDI в первую очередь предназначен для использования с волоконно-оптическим кабелем, хотя в отдельных частях сети могут применяться витые пары.
- Как показано на рис. 4.16, основу FDDI составляет одиночное (или двойное) кольцо, называемое *транком* (trunk). Станции могут соединяться непосредственно с транком или подключаться к нему через концентраторы. Допустимо подключение к транку древовидной структуры из концентраторов и станций.
- Когда в качестве транка применяется двойное кольцо, локальная сеть может быть сконфигурирована на восстановление работы при отказе одного из колец. Обычно трафик передается только по одному кольцу транка, но при его отказе начинает применяться второе кольцо, что позволяет обойти неисправность и продолжить работу сети.

Доступ к носителю в FDDI производится на основе пересылки маркера. Реально модель MAC-протокола очень похожа на 802.5 Token-Ring.

Кадр FDDI имеет MAC-заголовок и завершающую секцию, а когда кадр служит для пересылки IP, то используются уже знакомые нам заголовки 802.2 LLC и SNAP, отражающие перенос в кадре датаграммы IP.

4.16 Использование концентраторов

Локальные сети Ethernet, Token-Ring и FDDI вначале существенно различались по топологии кабельных сетей, однако со временем большинство организаций перешло на подключение систем через концентраторы (hub). Эти устройства упрощают администрирование локальных сетей и позволяют перейти на единую физическую топологию — звезду или цепочку звезд.

4.17 Коммутация

Все технологии рассмотренных локальных сетей имеют одно общее свойство: пересылаемый по сети кадр прослушивается всеми станциями сети. Хотя формально кадр предназначен только для одного физического адреса, любой владелец сетевой системы может настроить ее на смешанный режим (promiscuous), когда станция будет захватывать все пересылаемые в сетевом сегменте данные.

Требования к повышению производительности и усилению защиты привели к реализации *коммутации трафика*. Некоторые интеллектуальные концентраторы проводят переключение при каждой пересылке кадра между источником и приемником, делая его недоступным для остальных станций сети.

4.18 Широковещательные и многоадресные рассылки

Технологии локальных сетей со множественным доступом поддерживают широковещательные рассылки (broadcast). Для этого используется один специальный физический адрес назначения, указывающий, что обработать кадр должны все подключенные к локальной сети интерфейсы. Шестнадцатеричное представление широковещательного адреса можно записать как:

X'FF-FF-FF-FF-FF-FF

Сетевой интерфейс можно настроить и на прием кадров, посланных при одной или нескольких *многоадресных рассылках* (multicast). Многоадресность позволяет кадру попасть на указанный набор сетевых систем. Многоадресная рассылка всегда имеет единицу в младшем бите первого байта адреса:

X'01-00-00-00-00-00

Значения остальных бит устанавливаются в соответствии с конкретной службой многоадресной рассылки.

IANA зарезервировала список физических адресов многоадресных рассылок для нескольких служб. Например, многоадресная рассылка может применяться для передачи сообщения всем мостам сети. Отображение многоадресной рассылки уровня 3 в сетях IP в многоадресную рассылку уровня 2 будет рассмотрено в главе 5.

Термин *"одноадресная рассылка"* (unicast) применяется для указания уникального физического адреса, присвоенного одному из интерфейсов при выполнении широковещательной или многоадресной рассылки. Если заголовок кадра содержит сведения об одноадресной рассылке, то предполагается доставка такого кадра только одному, указанному сетевому интерфейсу.

Теперь рассмотрим технологии для региональных сетей.

4.19 Сети с коммутацией пакетов

Технология коммутации пакетов была введена в экспериментальном порядке еще в ARPANET. Затем она была улучшена и расширена многими дополнительными возможностями коммуникации данных. Сети с пакетами X.25 получили широкое распространение еще с 80-х годов. Однако большинство пользователей предпочли новую технологию коммутации пакетов Frame Relay, обеспечивающую широкий спектр разнообразных возможностей.

4.20 Сети X.25

Обычная телефонная сеть позволяет соединиться с любым другим абонентом в любой точке планеты. Существует специальная международная организация по стандартам, ответственная за правила объединения национальных телефонных сетей в общемировую систему. Долгое время эта организация называлась *Международным консультативным комитетом по телефонной и телеграфной связи* (International Telegraph and Telephone Consultative Committee — *CCITT*). Позднее она была переименована в *Сектор стандартизации в телекоммуникациях Международного телекоммуникационного союза* (Telecommunication Standardization Sector of the International Telecommunications Unit — *ITU-T*).

В течение 70-х гг. CCITT начал работу над рекомендациями для создания глобальной *цифровой* сети. Работа была завершена в 1980 г. Наиболее важной является рекомендация X.25, определяющая правила для подключения компьютеров к цифровой сети. Точнее, X.25 описывает интерфейс между компьютером, именуемым *оборудованием цифрового терминала* (data terminal equipment — *DTE*), и сетевым коммуникационным элементом — *оборудованием для терминирования цифровых цепей* (data circuit-terminating equipment — *DCE*) как части сети для личного и общедоступного использования, в последнем случае — для провайдера сетевых услуг.

X.25 устанавливает правила для надежных цепей между компьютерами. Эти *цепи* именуются *виртуальными*, поскольку в отличие от телефонной сети во время вызова пользователю не предоставляется фиксированный путь пересылки данных. Реальные связи используются совместно многими конкурирующими виртуальными цепями. Однако такое использование является прозрачным (невидимым) для пользователя цепи.

X.25 получил всемирное признание, и многие общедоступные цифровые сети X.25 соединяют компьютеры в глобальные сообщества.

Цифровые сети X.25 предоставляют цепи двух типов. *Коммутируемые виртуальные цепи* (switched virtual circuit) производят вызов данных так, как это делается в обычных телефонных сетях (в рекомендации X.121 от CCITT определен 14-значный номер для запросов X.25). Вызывающий абонент набирает 14-значный номер нужного ему компьютера, по которому производится вызов. В другом случае пользователь может применить *постоянную виртуальную цепь* (permanent virtual circuit), которая работает подобно выделенной телефонной линии.

Рекомендации CCITT не ограничивают *внутренней* структуры региональных цифровых сетей X.25, однако многие из них используют технологию внутренней коммутации пакетов.

4.20.1 Уровни в X.25

Протокол X.25 имеет три уровня. Уровень связи данных называется *балансированным протоколом доступа к связи* (Link Access Protocol Balanced — LAPB), а сетевой уровень — *уровнем пакетов X.25* (X.25 Packet Level). Владеющий оборудованием DTE пользователь устанавливает связь по X.25 с оборудованием DCE провайдера. Эта связь используется для пересылки данных *нескольких* виртуальных цепей уровня 3. Коммутируемая виртуальная цепь инициализируется посылкой пакета *Call Request* (запрос на вызов).

4.20.2 X.25 и IP

X.25 — одна из многих технологий региональных сетей, способных пересылать датаграммы IP. IP использует виртуальные цепи X.25 таким же способом, как телефонные линии,— "точка-точка", т.е. трафик IP передается между хостами и маршрутизаторами через виртуальные цепи X.25.

Протоколы для связи X.25 (уровня 2) и пакетов X.25 (уровня 3) снимают проблемы правильного порядка передачи данных и коррекции ошибок. Цепи X.25 специально предназначены для создания надежного соединения между конечными точками связи.

Может показаться странным, что ненадежные службы IP работают поверх надежных протоколов X.25. И еще более странным, что, как в X.25, так и в IP реализованы протоколы уровня 3. Однако, учитывая стоимость и общепринятые требования, можно не обращать внимания на нечеткость деления на уровни. Элементы протоколов уровня 3 для сетей VINES, DECnet и SNA могут передаваться по цепям X.25 не менее успешно. Кроме того, данные для работы мостов по уровню 2 также иногда передаются по цепям X.25.

На рис. 4.17 показан трафик IP от нескольких источников, который маршрутизируется по одной виртуальной цепи X.25 и пересылается в несколько различных точек назначения.



Рис. 4.17. Использование сети X.25 для маршрутизации датаграмм IP

4.20.3 Многопротокольный режим поверх X.25

Существуют два метода для пересылки многопротокольного трафика по сети X.25 (аналогичные методы и форматы применяются для пакетного режима ISDN):

1. Для каждого протокола устанавливается *отдельная* виртуальная цепь. Во время вызова партнеру указывается на пересылаемый протокол.
2. Устанавливается *одна* виртуальная цепь, совместно используемая несколькими протоколами. Во время вызова указывается на многопротокольный режим. Партнеру сообщается о применяемых протоколах, и соответствующие сведения добавляются в каждый из заголовков пакетов.

Выбор одного из методов определяется тем, насколько службы провайдера могут реализовывать дополнительные виртуальные цепи и как долго выполняются эти процессы.

В зависимости от экономической ситуации система может устанавливать коммутируемое соединение X.25 по умолчанию, когда несколько различных трафиков ожидают пересылки на удаленные сайты. Запрос закрывается по прошествии некоторого периода отсутствия активности. Обработка запроса обычно представляет собой очень медленный процесс, что делает многопротокольный режим более предпочтительным.

4.20.4 IP в отдельной виртуальной цепи X.25

Если трафик IP пересылается по отдельной коммутируемой виртуальной цепи, то это отражается в пакете *Call Request* протокола X.25, который инициализирует цепь. В этом пакете имеется необязательное поле *Call User Data* (вызываемые пользователем данные), которое для указания на трафик IP должно содержать значение X'CC.

Значение X'CC является *идентификатором протокола сетевого уровня* (Network Layer Protocol ID — NLPID), как это установлено для трафика IP организацией ISO.

4.20.5 Другие протоколы в отдельной виртуальной цепи X.25

Несколько других протоколов также имеют коды ISO для NLPID, но коммерческим лицензионным протоколам такие коды не присвоены. Однако, как можно предположить, многие коммерческие протоколы производят присваивание двухбайтового кода типа для общепринятого многопротокольного окружения — Ethernet. Например, трафик AppleTalk имеет код типа Ethernet со значением X'80-9B.

Для запуска в виртуальной цепи одного протокола с присвоением кода типа Ethernet код NLPID должен иметь значение X'80 со следующим далее подзаголовком SNAP, что указывается в поле Call User Data пакета Call Request протокола X.25. Например, для установки виртуальной цепи на работу с трафиком AppleTalk следует послать:

X'80-00-00-00-80-9B

4.20.6 Многопротокольный режим в виртуальной цепи

Если в виртуальной цепи организуется многопротокольный режим, поле Call User Data устанавливается в X'00 и в *каждый* кадр добавляется дополнительный заголовок, позволяющий идентифицировать тип протокола. Идентификация датаграммы IP очень эффективно выполняется посредством значения IP NLPID, равного X'CC,— это и будет дополнительным заголовком.

Для протоколов, идентификация которых выполняется кодом типа Ethernet, заголовок сообщения начинается NLPID со значением X'80, что указывает на следующий далее подзаголовок SNAP. Например, для цепи с многопротокольным режимом *каждому* PDU протокола AppleTalk будет предшествовать заголовок:

X'80-00-00-00-80-9B

4.20.7 Пакеты или PDU?

Существует незначительная сложность в способе пересылки информации по X.25. Некоторые сети X.25 передают пакеты очень маленького размера. Однако передать весь высокоуровневый PDU (например, датаграмму IP) можно через непрерывную *последовательность пакетов* (packet sequence) с объединением данных в единый PDU на другой стороне цепи (для этого служит флажок "more/nomore" — еще/больше нет). В этом случае идентификатор протокола требуется только в заголовке первого пакета X.25 из пересылаемой последовательности.

4.21 Frame Relay

Сети X.25 обеспечивают надежную и последовательную пересылку данных. Однако высоки непроизводительные расходы, связанные с качеством обслуживания в этих сетях. Когда трафик IP пересылается потоком по виртуальной цепи X.25, непроизводительные расходы приводят к существенным потерям.

Технология Frame Relay (это протокол уровня 2) более подходит для набора протоколов TCP/IP. В этом случае к датаграмме IP добавляются только заголовок уровня связи данных и завершающая часть для проверки ошибок.

X.25 хранит сообщение до подтверждения его приема и пересылает сообщение повторно, если не получает сигнала подтверждения (ACK). В отличие от X.25 Frame Relay *не сохраняет* сообщения, *не ждет* получения ACK и *не пересылает* данные повторно, что позволяет более эффективно использовать доступную полосу пропускания.

Начальный кадр Frame Relay стандартным способом определяет *только* службу *одной* виртуальной цепи. Пользователь должен связаться со службой провайдера и согласовать с ним требуемую скорость передачи при доступе к заданному сайту. Многие провайдеры обеспечивают скорости обмена вплоть до максимальной для линии T1 скорости в 1.544 Мбит/с. Вне территорий США и Японии доступны линии E1 со скоростью 2.048 Мбит/с. (T1 и E1 отличаются только организациями, принявшими данные стандарты — американским и европейским комитетами соответственно. С технической точки зрения данные стандарты, включая доступные скорости обмена, подобны, хотя и не совместимы

полностью между собой.— *Прим. пер.*) Обычно клиент платит фиксированную месячную арендную плату, величина которой зависит от согласованной скорости обмена.

Коммутируемые службы Frame Relay позволяют системам с присвоенными номерами динамически устанавливать коммуникационные цепи, как при обычном телефонном соединении. Поддержка коммутируемых служб обеспечивает больше возможностей, но заранее трудно предвидеть пользовательский трафик в каждый момент времени, а сети будут работать с перегрузками в случайные моменты.

Frame Relay обеспечивает лучшую производительность по сравнению с X.25 и поэтому чаще применяется на практике. На основе оборудования Frame Relay некоторые организации создают собственные лицензионные системы для внутренних сетей.

Как и для рассмотренных ранее протоколов, комитет IETF специфицировал формат для многопротокольной маршрутизации и перехода трафика через сетевые мосты для совместного использования цепей Frame Relay. Инкапсуляция датаграмм IP представлена на рис. 4.18.

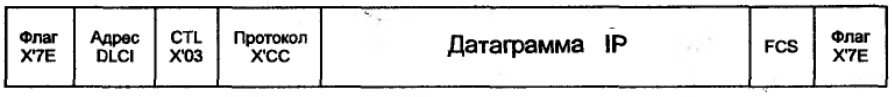

Рис. 4.18. Инкапсуляция датаграммы IP в Frame Relay

Адресное поле Frame Relay обычно имеет длину в 2 октета и содержит 10-битовое поле *идентификатора соединения по связи данных* (Data Link Connection Identifier — DLCI), определяющее отдельную цепь. Несколько бит в адресном поле используется для наполнения сигналов значениями, когда нужно указать, что кадр должен обрабатываться определенным образом, например для указания отмены кадра во время перегрузки. Если провайдер использует более длинные адреса, можно расширить адресное поле до 3 или 4 октетов.

Поле управления (CTL) имеет значение X'03 (т.е. нечисловая информация). Идентификатор протокола X'CC указывает, что кадр содержит датаграмму IP.

Кадры пересылаются по сети провайдера. Отбрасываются все кадры, проверочная последовательность (FCS) которых указывает на ошибку в них.



Рис. 4.19. Заголовок кадра Frame Relay с кодом типа Ethernet

Для протоколов, которые должны описываться кодом типа Ethernet (например, AppleTalk), заголовок сообщения имеет формат, показанный на рис. 4.19. Для улучшения выравнивания сообщения после поля управления вставлен добавочный октет-заполнитель X'00. Значение X'80 для NLPID говорит о следующем далее подзаголовке SNAP. В нашем примере он содержит код типа Ethernet для протокола AppleTalk.

За исключением байта-заполнителя (pad), данный заголовок идентичен заголовку для цепи X.25 в многопротокольном режиме.

4.22 SMDS

Служба коммутируемых многомегабитных данных (Switched Multimegabit Data Service — SMDS) является еще одной общедоступной службой с коммутацией пакетов. Она была разработана для региональных компаний Bell (в свое время корпорация Bell была разделена правительством США на несколько региональных компаний.— *Прим. пер.*). Данная служба предназначена для предоставления широкого спектра вариантов производительности, включая высокоскоростные соединения, например 155 Мбит/с.

Интересным свойством SMDS является то, что данные могут быть посланы без открытия виртуальной цепи — *без создания соединения* (connectionless). На практике логическая подсеть IP может быть сформирована с возможностями региональных сетей, и (см. рис. 4.20) этот логический сегмент региональной сети будет наследовать многие возможности высокоскоростных локальных сетей. Такие свойства делают SMDS привлекательной для создания магистральной структуры региональных сетей.

Протокол интерфейса SMDS (SMDS Interface Protocol — SIP) основан на стандарте IEEE с номером 802.6.

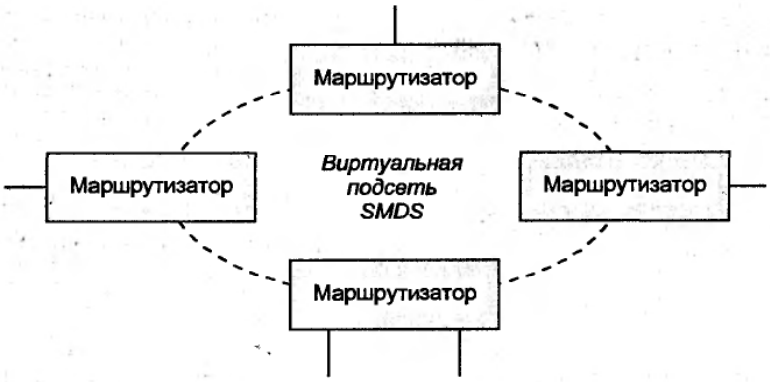

Рис. 4.20. Магистральная региональная сеть SMDS

4.22.1 IP поверх SMDS

На рис. 4.21 показан формат заголовка после вставки заголовка SIP SMDS, что отражает факт присутствия датаграммы IP.

Этот формат подобен используемому в локальных сетях IEEE 802. Первые три октета создают заголовок LLC IEEE 802.2, а содержащий значение X'08-00 подзаголовок SNAP определяет для IP код типа Ethernet.

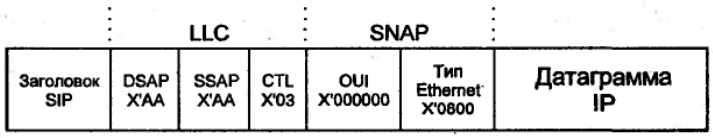

Рис. 4.21. Для идентификации IP в SMDS используются LLC и SNAP.

4.23 ATM

Режим асинхронной пересылки (Asynchronous Transfer Mode — ATM) представляет собой технологию с коммутацией ячеек, подходящую как для локальных, так и для региональных сетей. ATM объединяет преимущества безопасности при коммутируемом доступе с высокой производительностью и гибкостью. Эту технологию можно характеризовать следующим образом:

- Данные коммутируются в 53-октетных ячейках.
- Каждая ячейка имеет пятибайтовый заголовок, содержащий информацию для ее маршрутизации.
- Кадры разбиваются на ячейки в источнике и вновь объединяются в кадры в точке назначения с помощью *уровней адаптации ATM* (ATM Adaptation Layer — AAL).
- Существует несколько AAL, однако к пересылке датаграмм IP имеет отношение только AAL5.
- Работу по сегментации и последующей сборке кадров при пересылке по региональной сети выполняет интерфейс обмена данными (Data Exchange Interface — DXI) — часть оборудования, соответствующая цифровому интерфейсу обычной телефонной линии.

Как в X.25 или Frame Relay, коммуникации ATM формируются путем создания виртуальной цепи и пересылки кадров по этой цепи.

В сетях ATM существуют два метода обслуживания многопротокольного трафика:

- Создание отдельной виртуальной цепи для каждого протокола
- Совместное использование одной виртуальной цепи всеми протоколами

Выбор одного из методов зависит от стоимости, а также от времени установки и закрытия виртуальной цепи.

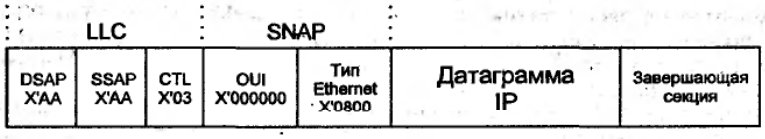

Рис. 4.22. Для идентификации IP в ATM AAL используются LLC и SNAP.

Если для каждого протокола используется отдельная виртуальная цепь (как в X.25), то тип протокола для коммутируемой цепи можно анонсировать только один раз — в сообщении запроса на вызов.

Когда несколько маршрутизируемых протоколов совместно используют одну виртуальную цепь (см. рис. 4.22), кадр AAL5 начинается с уже известных нам заголовков LLC и SNAP. Тип IP Ethernet заключается в подзаголовке SNAP (см. рис. 4.22).

Отметим, что кадр AAL5 не имеет в заголовке полей с адресами источника и назначения. Дело в том, что после вызова устанавливается виртуальная цепь от источника до точки назначения, а необходимая для коммутации в точке назначения информация находится в 5-октетном заголовке ячейки.

Заключительная часть AAL5 содержит байты-заполнители (для выравнивания), поле данных пользователя, поле *payload lenght* (длина полезной нагрузки) и проверочную последовательность кадра (FCS). Полезная нагрузка учитывает размеры заголовков LLC и SNAP и самой датаграммы.

4.24 Максимальное число пересылаемых элементов

Каждая из рассмотренных нами технологий имеет различные максимальные размеры для своих кадров. После исключения заголовка кадра, заключительной части, а также заголовков LLC и SNAP (если они присутствуют), полученный результат будет определять максимально возможный размер датаграммы, которую можно переслать по носителю. Эта величина называется *максимальным пересылаемым элементом* (Maximum Transmission Unit — *MTU*).

Например, максимальный размер кадра для сети 802.3 10BASE5 равен 1518 октетам. Вычитая длину MAC-заголовка и завершающей части (18 октетов), поле управления связи Type 1 и заголовок SNAP (8 октетов), мы получим MTU, равный 1492 октетам.

В таблице 4.1 приведены MTU для различных технологий.

Таблица 4.1

Максимальный пересылаемый элемент

Протокол	Максимальное количество октетов в датаграмме (MTU)
По умолчанию для PPP	1500
PPP (с небольшой задержкой)	296
SLIP	1006 (исходное ограничение)
X.25	1600 (отличается для некоторых сетей)
Frame Relay	Обычно не менее 1600
SMDS	9235
Ethernet версии 2	1500
IEEE 802.3/802.2	1492
IEEE 802.4/802.2	8166
16 Mb IBM Token-Ring	Максимально 17914
IEEE 802.5/802.2 4-Mb Token-Ring	Максимально 4464
FDDI	4352
Hyperchannel	65535
ATM	По умолчанию 9180 Максимально возможно 16K-1

Специальным случаем является линия "точка-точка". Она реально не наследует ограничений на размер датаграммы. Оптимальный размер зависит от уровня ошибок в данной линии связи. Если он высок, то лучшая производительность достигается при более коротких элементах данных. Максимальное значение по умолчанию в 1500 байт используется наиболее часто.

Первоначально протокол SLIP был специфицирован с максимальной длиной датаграммы в 1006 байт. Некоторые реализации могут поддерживать до 1500 байт, преобразуя SLIP в другие форматы пересылки данных по последовательной линии "точка-точка".

Для Token-Ring показано предельное значение MTU. Реально MTU для Token-Ring зависит от множества факторов, включая время удержания маркера в кольце.

4.25 Создание туннелей

Всегда придерживаться структуры деления на уровни — хорошая идея, но часто используется более простой способ пересылки данных из одной точки в другую с помощью другого протокола. Такой процесс называется созданием *туннеля* (tunneling) — возможно, по причине временного скрытия данных в другом протоколе до момента достижения выходной точки туннеля.

Создание туннеля не представляет особых сложностей — просто вокруг элемента данных создается один или несколько дополнительных заголовков, маршрутизация выполняется средствами другого протокола, а извлечение полезной информации происходит в точке назначения.

Мы уже рассматривали применение туннеля. Когда датаграмма IP перемещается по сети X.25, она обрамляется заголовком сетевого уровня X.25. В этом случае трафик IP пересылается через туннель в среде X.25.

На практике применяется множество других вариантов использования туннелей. Иногда трафик IPX операционной системы Novell NetWare пересылается по туннелю в сети IP. Сообщения из NetWare обрамляются заголовками IP или UDP, маршрутизация производится в сети IP, а доставка выполняется на удаленный сервер NetWare. Многие разработчики предлагают продукты для пересылки по туннелю трафика SNA в сети IP.

Туннели всегда приводят к дополнительной нагрузке. Поскольку часть сетевого пути скрыта внутри внешнего протокола, использование туннеля сокращает возможности по управлению и обслуживанию в сети и часто создает дополнительный трафик, не имеющий отношения к пересылке полезной информации.

4.26 Совместное использование сетевого интерфейса

Как уже отмечалось, несложно найти локальные и региональные сети, использующие одновременно несколько протоколов. На практике один сетевой узел иногда посылает и принимает данные по нескольким протоколам через единый сетевой интерфейс.

Чтобы понять, как это происходит, рассмотрим конкретный интерфейс — локальную сеть Ethernet. Если персональный компьютер или сервер станут использовать интерфейс Ethernet для TCP/IP, IPX или DECnet, то как будут сосуществовать эти протоколы?

Мы уже знаем ответ на этот вопрос. Заголовок уровня связи данных содержит поле, идентифицирующее протокол сетевого уровня для конкретного сообщения.

На рис. 4.23 показан интерфейс Ethernet, совместно используемый стеками протоколов TCP/IP, IPX и DECnet. Посредничающий при выполнении операций программный уровень *драйверов устройств* скрывает действия по вводу/выводу от стеков протоколов более высокого уровня.

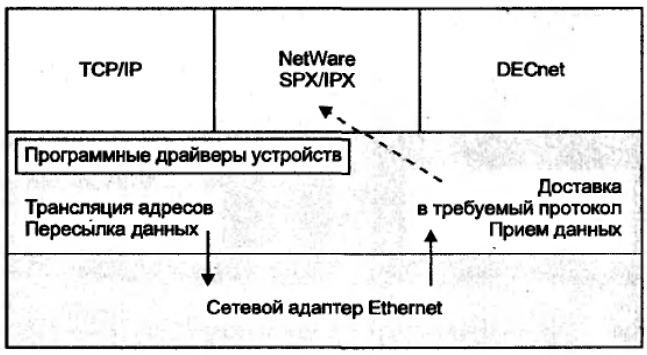

Рис. 4.23. Протоколы совместно используют сетевой интерфейс.

4.27 Замечания об уровне связи данных

Доля датаграмм, управляющих информацией, оказывает влияние на общую производительность. Разумеется, когда нужно переслать в сети большой объем данных, лучше заполнять датаграммы как можно плотнее.

Для разных типов сетей максимальные размеры датаграмм различны. В главе 6 мы познакомимся с предоставляемым в IP механизмом фрагментации — разделением больших датаграмм с последующей пересылкой в кадрах с датаграммами меньшего размера. Такая возможность обеспечивает доставку данных, даже если превышается используемый размер MTU. Однако можно предположить, что фрагментация и последующее воссоздание снижают время ответа сети.

Если пара взаимодействующих хостов подключена к одной и той же локальной сети, то желательно оптимизировать пересылку данных за счет использования максимально возможного размера датаграммы. Однако при работе с удаленным хостом через сеть неизвестного типа некоторые реализации принудительно используют меньшее значение для максимального размера датаграммы (иногда 576 октетов), чтобы избежать фрагментации.

Далее мы увидим, что процедура автоматического определения наибольшего размера датаграммы может выполняться для любого заданного пути пересылки данных (глава 7). Оптимизируя размер датаграмм, можно исключить фрагментацию и пересылать большие массивы данных более эффективно.

4.28 Завершающая часть кадра

Некоторые проблемы возникают при использовании нестандартных форматов протоколов из устаревших версий TCP/IP. Реализация BSD 4.2 предоставляет нестандартный формат для MAC-кадров Ethernet, в котором исключено поле типа кадра, а информация заголовков уровней 3 и 4 перемещена в *завершающую часть кадра* (trailer). Цель этой перестановки — ускорение обработки поступающих кадров за счет снижения количества копирования данных. Такая возможность реализуется в некоторых коммерческих продуктах.

Использование завершающей части кадра в стиле Беркли может привести к несовместимости, но этот вариант становится все более редким на практике. Если все же необходимо воспользоваться этим методом, следует ознакомиться с рекомендациями из RFC 1122 по его безопасному применению.

4.29 Рекомендуемая литература

RFC 1661 описывает протокол PPP. Аутентификация в PPP рассматривается в RFC 1334, а автоматический мониторинг качества линии — в RFC 1333.

Существует несколько RFC, обсуждающих пересылку датаграмм IP поверх средств более низкого уровня: RFC 1356 для X.25, RFC 1490 для Frame Relay, RFC 1209 для SMDS, RFC 1390 для FDDI, RFC 1577, 1932, 1626 и 1755 для ATM, RFC 1088 для NetBIOS, RFC 1055 для SLIP, RFC 1042 для сетей IEEE 802, RFC 894 для Ethernet и RFC 1201 для ARCNET.

Сведения о HDLC можно найти в ISO 3309, 4335 и 7809. Серия IEEE 802 и ISO 8802 описывает физические носители, доступ к носителю, а также протоколы логических связей для локальных и городских сетей.

Рекомендации CCITT по X.25 можно обнаружить в "Красной книге" CCITT 1984 г. Существует несколько документов по стандартам для Frame Relay. Лучше всего начать с ANSI T1.606 и рекомендации CCITT I.122.

RFC 893 обсуждает инкапсуляцию в завершающей части кадра.

Глава

5

Именование и адресация

5.1 Введение

Каждый сетевой узел должен иметь имя и адрес. Каким образом производится их присваивание? Для небольшой независимой локальной сети это не проблема, но если количество компьютеров составляет сотни или тысячи, выбор хорошей схемы для присваивания имен и адресов имеет большое значение, поскольку он позволит избежать неприятностей при удалении, добавлении или перемещении хостов и маршрутизаторов.

Администраторы Интернета сталкиваются с обслуживанием имен и адресов в огромной сети, размер которой ежегодно удваивается. Однако в Интернете выбрана удачная стратегия — делегирование полномочий.

Схема имен и адресов Интернета TCP/IP позволяет:

- Делегировать присваивание имен и адресов тем, кто несет ответственность за работу всей или части отдельной сети
- Позволить именам отражать логическую структуру организации
- Присваивать адреса, отражающие топологию физической сети в организации

Далее мы увидим, что в Интернете применяется метод иерархического именования, позволяющий администратору создавать для систем описательные и простые в запоминании имена.

5.2 Примеры имен Интернета

Некоторые имена Интернета достаточно эксцентричны. Например, группа хостов Медицинской школы Йельского университета имеет следующие названия:

blintz.med.yale.edu

couscous.med.yale.edu

gazpacho.med.yale.edu

lazagne.med.yale.edu

paella.med.yale.edu

sukiyaki.med.yale.edu

strudel.med.yale.edu

Серверам часто дают такие имена, чтобы пользователи могли легко их найти. Например:

www.whitehouse.gov (Белый дом — резиденция президента США.— *Прим. пер.*)

ftp.microsoft.com (ftp-сайт компании Microsoft.— *Прим. пер.*)

gopher.jvnc.net (служба Gopher.— *Прим. пер.*)

Имена сайтов (узлов) Интернета не зависят от регистра символов. Например, *www.whitehouse.gov* можно записать как *WWW.WHITEHOUSE.GOV* или *WWW.Whitehouse.Gov*. В книге мы будем использовать имена из строчных, прописных или из комбинации строчных и прописных символов.

5.3 Иерархическая структура имен

Иерархическая структура имен достаточно проста. Каждая организация имеет содержательное имя верхнего уровня, подобное *yale.edu*, *whitehouse.gov* или *microsoft.com*. Схему именования внутри этих имен организация выбирает самостоятельно. Например, в Йельском и во многих других университетах именование делегировано факультетам и подразделениям. Следовательно, появляются имена, заканчивающиеся на:

cs.yale.edu

math.yale.edu

geology.yale.edu

Некоторые подразделения университета создают дополнительные подимена (имена более низкого уровня). Например, компьютеры вычислительного центра, расположенные в здании с неформальным прозвищем *The Zoo* (зоопарк), именуются по названиям различных животных:

lion.zoo.cs.yale.edu

leopard.zoo.cs.yale.edu

tiger.zoo.cs.yale.edu

Все компьютеры этого зоопарка находятся в одной локальной сети. Однако имена могут присваиваться на основе концепций *администрирования*, и компьютеры другой группы вычислительного центра с другим подименем *не подключены* к той же локальной сети, но имеют имена:

hickory.theory.cs.yale.edu

pecan.theory.cs.yale.edu

olive.theory.cs.yale.edu

walnut.theory.cs.yale.edu

5.4 Администрирование имен

Использование иерархической структуры имен упрощает проверку уникальности имени компьютера, поскольку она возлагается на соответствующий персонал. Отметим следующее:

lion	Администрируется в пределах вычислительного центра *зоопарка*, что позволяет иметь для каждого компьютера уникальное имя (*lion*, *tiger* и т.д.).
lion.zoo	Администрируется в пределах всего вычислительного центра. Для каждой подгруппы используется уникальное имя (*zoo*, *theory* и т.д.).
lion.zoo.cs	Администратор всей компьютерной сети Йельского университета присваивает каждому факультету и подразделению уникальное имя (*cs*, *math*, *geology*), что обеспечивает уникальность имен компьютеров во всем университете.
lion.zoo.cs.yale.edu	Обслуживается официальным комитетом по регистрации, что обеспечивает уникальность имен для всех организаций (*yale.edu*, *microsoft.com*). Следовательно, компьютеры во всем мире могут иметь уникальные имена.

Для обеспечения всемирной уникальности имен Интернета необходима служба регистрации имен, следящая за тем, чтобы *каждая* компания и организация имела уникальное (отличное от всех других) имя.

Первоначально сеть Интернет спонсировалась Министерством обороны США, создавшим *собственный информационный центр сетей* (Department of Defence Network Information Center — DDN NIC), который и занимался администрированием и регистрацией всех имен и адресов.

В 1993 г. ответственность за гражданские имена и адреса принял на себя Национальной научный фонд (National Science Foundation — NFS), а обслуживанием военных систем продолжал заниматься DDN NIC.

NFS организовал *службу регистрации InterNIC* Registration Service (InterNIC) — главную организацию по именованию и адресации во всемирном масштабе. Однако такая централизация привела к ненужным задержкам в работе этой службы. Поэтому InterNIC делегировала авторизацию имен в два главных центра региональной регистрации:

- Азиатско-Тихоокеанский сетевой информационный центр (The Asia Pacific Network Information Center)
- Европейский координационный сетевой центр RIPE (The RIPE Network Coordination Center). RIPE означает Европейский исследовательский центр по IP — Reseaux IP Europeens.

InterNIC и два этих региональных центра делегируют полномочия по именованию и адресации национальным и локальным регистрационным центрам, несущим ответственность за свои регионы.

В приложении С представлены почтовые адреса, номера телефонов и адреса электронной почты InterNIC, а также главных региональных регистрационных центров. Там же приведены ссылки на архивы регистрационных форм и сведения для доступа к региональным регистрационным центрам.

5.5 Формальная структура имен

Имя состоит из последовательности меток, разделенных символами точки. Очень часто в имени присутствует две, три, четыре или пять меток. Ниже представлены допустимые имена для компьютеров:

bellcore.com

www.apple.com

ftp.ncsa.uiuc.edu

lion.zoo.cs.yale.edu

Более длинные имена сложны для запоминания и ввода пользователями. Однако стандарт Интернета допускает для каждого маркера длину до 63 символов, а общую длину всего имени — до 255 символов.

5.6 Всемирное дерево имен

Имена Интернета структурированы как дерево (см. рис. 5.1). Каждому узлу дерева присвоена метка. Каждый узел дерева имеет имя, называемое *именем домена* (domain name). Имя домена для узла создается из меток, проходимых по пути *от* этого узла *до* вершины дерева. Имена доменов узлов записываются как последовательность меток, разделенных точками.

Кроме того, *доменом* называется часть дерева имен, содержащая один из узлов и все нижестоящие узлы. Другими словами, домен создается из всех имен с одинаковыми окончаниями. Примеры доменов:

- *edu* и все имена ниже этого узла (заканчиваются на *edu*)
- *yale.edu* и все имена ниже этого узла (заканчиваются на *yale.edu*)
- *cs.yale.edu* и все имена ниже этого узла (заканчиваются на *cs.yale.edu*)

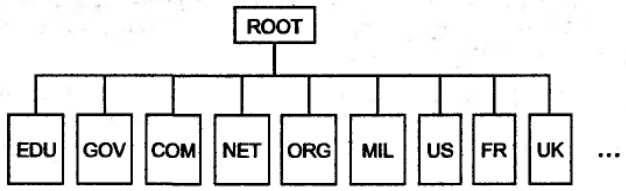

Рис. 5.1. Всемирное дерево имен

Доменами верхнего уровня (top-level domain) являются (см. рис. 5.1):

- *edu* — учебные заведения с четырехгодичным обучением
- *gov* — учреждения федерального правительства США
- *com* — коммерческие организации
- *net* — организации сетевых служб Интернета

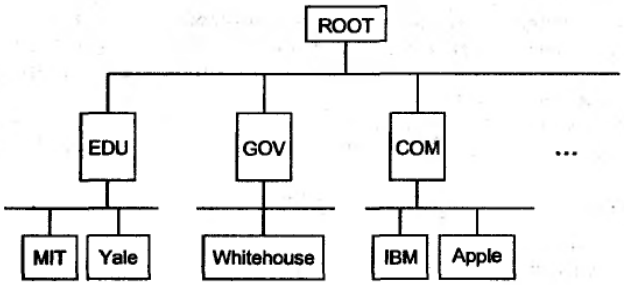

Рис. 5.2. Домены второго уровня

- *org* — некоммерческие организации (*96olympics.org*, *npr.org*)
- *int* — международные организации (*gopher.nato.int*). Редко используются и не видны в сети.
- *mil* — военные организации (*army.mil*, *navy.mil*)
- *us* — организации штатов США и региональных правительств, школы, двухгодичные колледжи, библиотеки и музеи
- *Countries* — двухсимвольный код ISO, идентифицирующий десятки других доменов высокого уровня: *fr* — Франция, *uk* — Великобритания, *de* — Германия и т.д. (Для России: *su* — старый код и *ru* — новый.— *Прим. пер.*) Структура дерева внутри кода страны администрируется в пределах данной страны.

Домены *yale.edu*, *whitehouse.gov* и *ibm.com* называются *доменами второго уровня* (second-level domain) — (см. рис. 5.2).

Есть еще одно ограничение. Меткой для *корня* (root) дерева имен служит символ точки. Именно поэтому именем системы *lion* вычислительного центра Йельского университета реально является:

lion.zoo.cs.yale.edu.

Однако большинство пользователей (в том числе и автор этой книги) опускают последнюю точку при вводе имен.

5.7 Конфигурирование имен систем

Конфигурирование имени системы различается для разных систем. Наиболее часто администратор вводит это имя в меню или указывает при выполнении команды.

Для имени *tiger* в системе Unix от *SunOS* команда *hostname* позволяет указать или просмотреть имя хоста:

```
> hostname
tiger.jvnc.net
```

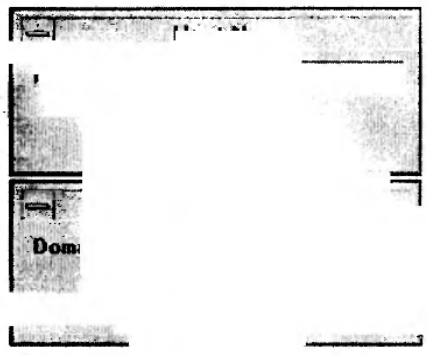

Рис. 5.3. Конфигурирование имени системы

Некоторые системы разделяют имя на две части — начальную метку и остаток от имени домена. Это делается с целью применения автоматического использования коротких имен для систем одного узла домена. Например, если пользователь работает на компьютере домена *jvnc.net* и вводит *mickey*, то автоматически будет использовано имя *mickey.jvnc.net*.

Пользователи программного продукта *Chameleon* для Windows вводят имя своего компьютера в двух раскрывающихся меню (см. рис. 5.3).

5.8 Адреса

В протоколе IP используются *IP-адреса*, которые идентифицируют хосты и маршрутизаторы для пересылки на них информации. Каждому хосту нужно присвоить уникальный IP-адрес, который и будет использоваться в реальном взаимодействии. Имена хостов транслируются в IP-адреса с помощью поиска в базе данных, содержащей пары имя-адрес.

Когда разрабатывалась адресация для IP, никто не предполагал, что миллионам компьютеров по всему миру потребуются IP-адреса. В то время разработчики исходили только из требований сообщества университетов, исследовательских центров, военных и правительственных организаций.

Был выбран резонный для того времени метод. В соответствии с 32-разрядным регистром компьютера IP-адрес имеет длину в 32 бита (4 октета): результирующее *адресное пространство* (address space) — множество всех возможных адресов — составляет 2^{32}, т.е. 4 294 967 269 номеров.

Нотация с символом *точки* упрощает чтение и запись IP-адресов. Каждый октет (8 бит) адреса преобразуется в десятичное число и точкой отделяется от других. Например, адрес для *blitz.med.yale.edu* имеет в двоичной записи и нотации с точками следующие значения:

10000010 10000100 00010011 00011111

130 . 132 . 19 . 31

Отметим, что наибольшим числом в записи с точками может быть 255, что соответствует 11111111.

5.9 Форматы адресов

Как показано на рис. 5.4, IP-адрес состоит из двух частей: *адреса сети* (network address) и *локального адреса* (local address). Адрес сети идентифицирует сеть, к которой подключен узел, а локальный адрес определяет отдельный узел внутри сети организации.

Каждый компьютер должен иметь IP-адрес, уникальный среди всех систем, с которыми он будет взаимодействовать.

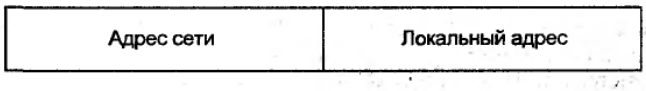

Рис. 5.4. Формат IP-адреса

5.10 Классы адресов

Организация, планирующая подключение к Интернету, должна получить для себя блок уникальных IP-адресов. Этот блок выделяется соответствующей регистрационной службой.

По соглашению, регистрационная служба делегирует выделение больших блоков пространства IP-адресов своим провайдерам, что позволяет организациям получать адреса непосредственно у провайдеров, а не в самой службе.

Многие годы существовало только три размера блоков адресов — большой, средний и малый. Соответственно, было три различных формата сетевого адреса:

- Класса A для очень больших сетей
- Класса B для средних сетей
- Класса C для небольших сетей

Форматы для классов A, B и C показаны на рис. 5.5. Характеристики для адресов каждого класса представлены в таблице 5.1.

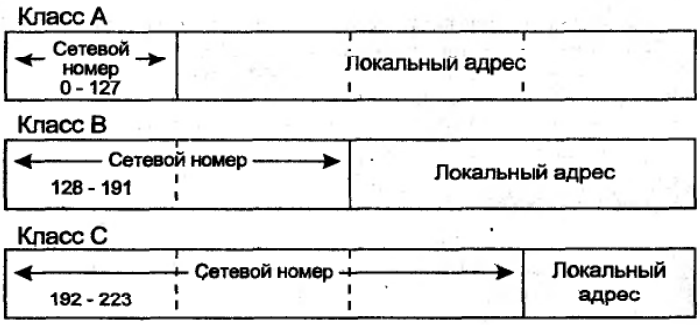

Рис. 5.5. Традиционные классы адресов

В первые дни существования Интернета все адреса класса A получили организации с очень большими сетями, например Военно-морской флот США или корпорация DEC. Сетевая часть таких адресов имеет длину в 1 октет, а оставшиеся 3 октета могут использоваться как локальная часть адреса и присваиваться как номера для узлов сетей.

Существует очень мало адресов класса A, и даже большим организациям часто вполне достаточно адресов класса B. Сетевая часть адреса класса B имеет длину в 2 октета, а 2 оставшихся октета служат для нумерации узлов.

Небольшим организациям присваиваются один или несколько адресов класса C. Сетевая часть в таком адресе имеет длину в 3 октета, а оставшийся октет используется как локальная часть и служит для нумерации узлов.

Это простейший способ распределения IP-адресов. Нужно просто проанализировать первое число в нотации с точками. Диапазоны чисел для каждого класса показаны в таблице 5.1 и на рис. 5.5.

Таблица 5.1

Характеристики классов адресов

Класс	Длина сетевого адреса (в октетах)	Первое число	Количество локальных адресов
A	1	0-127	16 777 216
B	2	128-191	65 536
C	3	192-223	256

Кроме классов A, B и C, существуют специальные адресные форматы: классы E и D. Адреса класса D применяются для *многоадресных* рассылок в IP, когда одно сообщение распространяется среди группы разбросанных по сети компьютеров. Многоадресная рассылка необходима для приложений проведения конференций, которые мы рассмотрим ниже.

Адреса класса E используются в экспериментальных целях.

- Адреса класса D начинаются с номеров между 224 и 239.
- Адреса класса E начинаются с номеров между 240 и 255.

5.11 Адреса не подключенных к Интернету систем

Несколько блоков адресов зарезервировано для сетей, которые не подключены к Интернету и их системам *не требуются* соединения с другими организациями. К этим адресам относятся:

10.0.0.0 - 10.255.255.255

172.16.0.0 - 172.31.255.255

192.168.0.0 - 192.168.255.255

Многие организации используют эти адреса. Но если такая компания впоследствии сольется с другой компанией или решит организовать соединения со своими клиентами или поставщиками через TCP/IP, произойдет конфликт адресов. Однако можно зарегистрировать адреса класса C и использовать их для внешних коммуникаций. Можно купить программное обеспечение агентов-прокси (proxy)

для преобразования информации между внутренними адресами компьютеров и внешним миром через зарегистрированные адреса класса С. (Локальные сети, которые никогда не предполагается соединять с Интернетом, могут иметь произвольные IP-адреса.— *Прим. пер.*)

Все "за" и "против" использования зарезервированных адресов можно узнать из RFC 1918 *Address Allocation for Private Internet* (Присваивание адресов в частных сетях Интернета).

5.12 Примеры адресации

В этом разделе мы познакомимся с несколькими примерами глобально уникальных адресов классов А, В и С. Позднее мы рассмотрим новый *бесклассовый* (classless) метод присваивания сетевых адресов.

5.12.1 Присваивание сети адресов класса А

Некоторые очень большие организации имеют адреса класса А. В этом случае при регистрации присваивается фиксированное значение первого октета адреса, а три оставшихся октета администрируются внутри самой организации. Например, следующие адреса и имена хостов предназначены для компании Hewlett-Packard, которая имеет адрес класса А со значением 15.

15.255.152.2 *relay.hp.com*

15.254.48.2 *hpfcla.fc.hp.com*

Компания Hewlett-Packard владеет номерами от 15.0.0.0 до 15.255.255.255. Эти номера создают *адресное пространство* данной организации.

5.12.2 Присваивание сети адресов класса В

Зарегистрированное авторство на адреса позволяет присвоить фиксированное значение первым двум октетам в адресе класса В. Последние два октета администрируются в пределах самой организации. Например, следующие адреса и имена хостов предназначены для провайдера Global Enterprise Systems Service Provider, которому был присвоен адрес класса В со значением 128.121.

128.121.50.145 *tigger.jvnc.net*

128.121.50.143 *mickey.jvnc.net*

128.121.51.51 *camel-gateway.jvnc.net*

Системы Global Enterprise владеют адресами от 128.121.0.0 до 128.121.255.255.

Адреса класса В очень популярны, и многие организации стремятся зарегистрировать и получить именно их. К сожалению, хотя и существует более 16 000 возможных идентификаторов для сетей класса В, их выделение уже завершено.

5.12.3 Присваивание сетям адресов класса С

Организациям с небольшими сетями, которым требуются глобально уникальные адреса, предоставляются адреса класса С. Это означает, что регистрационное авторство присваивается на три первых октета полного адреса организации. Сама организация управляет только последним октетом. Например, компании WAIS, Inc. был присвоен адрес класса С 192.216.46. Некоторыми из ее адресов и имен хостов могут быть:

192.216.46.4 *ns.wais.com*

192.216.46.5 *webworld.wais.com*

192.216.46.98 *wais.wais.com*

WAIS, Inc. владеет номерами от 192.216.46.0 до 192.216.46.255.

5.13 Трансляция имен в адреса

Конечному пользователю проще вводить легко запоминаемые имена, когда требуется указать IP-адрес для системы назначения. Многие компьютеры сконфигурированы с созданием небольшого файла *hosts*, в котором перечислены имена и адреса всех локальных систем. Часть такого файла, хранимого на хосте компании Global Enterprise Systems с именем *tigger.jvnc.net*, может выглядеть как:

```
128.121.50.2        r2d2.jvnc.net    r2d2
128.121.50.7        nisc.jvnc.net    nisc
```

```
128.121.50.141     minnie.jvnc.net  minnie
128.121.50.143     mickey.jvnc.net  mickey
128.121.50.144     donald.jvnc.net  donald
128.121.50.145     tigger.jvnc.net  tigger
128.121.50.148     chip.jvnc.net    chip
128.121.50.149     bambi.jvnc.net   bambi
128.121.50.152     sleepy.jvnc.net  sleepy
```

Все остальные примеры этой главы получены на *tigger.jvnc.net.*

Запрос к распределенной базе данных системы именования доменов (Domain Name System — DNS) применяется при глобальном преобразовании имен в адреса. Предположим, приложение *nslookup* посылает запрос на трансляцию имени в Domain Name Server, называемую *r2d2.jvnc.net.* Мы решили выяснить адрес WWW сервера Белого дома (White House) и сервера пересылки файлов компании Novell, Inc.:

```
> nslookup
Default Server: r2d2.jvnc.net
Address: 128.121.50.2

> www.whitehouse.gov.
Server: r2d2.jvnc.net
Address: 128.121.50.2

Name: www.whitehouse.gov
Address: 128.102.252.1

> ftp.novell.com.
Server: r2d2.jvnc.net
Address: 128.121.50.2

Name: bantu.Provo.Novell.COM
Address: 137.65.1.3
Aliases: ftp.novell.com
```

Ответ на второй запрос показывает, что имя *ftp.novell.com* в действительности является *псевдонимом* (alias) для компьютера *bantu.Provo.Novell.COM.*

5.14 Псевдонимы имен

Часто по соглашению можно присвоить компьютеру дополнительно к его реальному имени некоторый псевдоним (или краткое имя — nickname). Например, *хост nicol.jvnc.net* обеспечивает пересылку файлов, службу gopher и службу World Wide Web (WWW). По соглашению, ему дополнительно присвоены следующие краткие имена:

ftp.jvnc.net

gopher.jvnc.net

www.jvnc.net:

```
> ftp.jvnc.net.
Server: r2d2.jvnc.net
Address: 128.121.50.2

Name: nicol.jvnc.net
Address: 128.121.50.10
Aliases: ftp.jvnc.net
```

```
> gopher.jvnc.net.
Server: r2d2.jvnc.net
Address: 128.121.50.2

Name: nicol.jvnc.net
Address: 128.121.50.10
Aliases: gopher.jvnc.net

> www.jvnc.net.
Server: r2d2.jvnc.net
Address: 128.121.50.2

Name: nicol.jvnc.net
Address: 128.121.50.10
Aliases: www.jvnc.net
>
```

Когда нагрузка на *nicol* становится слишком большой, одну из его служб (и краткое имя этой службы) можно перенаправить на другой хост. Такой способ дает пользователю возможность достичь службы по неизменному имени, даже если ее домашний сайт будет изменен. Реальное имя хоста называется *каноническим именем* (canonical name).

5.15 Неэффективность классов адресов

Сеть класса A охватывает 16 777 216 адресов, класса B — 65 536, а сеть класса C содержит только 256 номеров. Огромная разница между этими значениями делает неэффективным выделение адресных блоков и приводит к истощению адресного пространства IP.

Более эффективный *бесклассовый* метод выделения адресов для организации рассмотрен в разд. 5.19.

5.16 Сети и подсети TCP/IP

Организации с адресами сетей класса A или B, как правило, имеют очень большие сети, состоящие из множества локальных и нескольких региональных сетей. В этом случае имеет смысл разделить адресное пространство так, чтобы оно отражало структуру сети в виде нескольких подсетей. Для этого локальная часть адреса разделяется на *часть для подсети* (subnet part) и *системную часть* (system part) любым выбранным организацией способом (см. рис. 5.6).

Определение размера части адреса для подсети и присваивание номеров подсетям производится организацией, владеющей данной частью адресного пространства.

Адреса подсети часто создаются в соответствии с байтовой границей. Организация с адресом класса B, например 128.21, может использовать для идентификации подсети третий байт:

128.121.1

128.121.2

128.121.3

Четвертый байт будет использоваться для идентификации отдельных хостов в подсети.

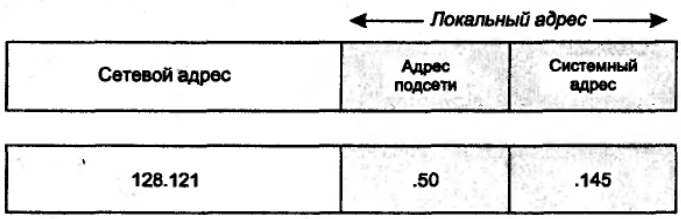

Рис. 5.6. Деление локального адреса на подсеть и системную часть

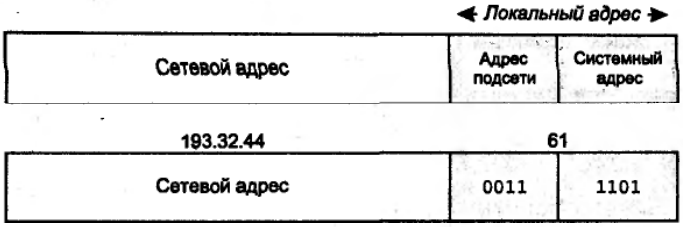

Рис. 5.7. Четырехбитовая часть для подсети в адресе класса C

Организация с адресом класса C имеет только однобайтовое адресное пространство. Она мо вообще не проводить деления на подсети или использовать первые 4 бита для адреса подсети и 4 б для адресов хостов (см. рис. 5.7). На рисунке видно, что локальный адрес (61) имеет двоичное пр ставление 0011 1101. Первые 4 бита идентифицируют подсеть, а последние 4 бита определяют систем

5.17 Маска подсети

Маршрутизация трафика на хост выполняется посредством анализа сетевой части и части для п сети IP-адреса. Сетевые части адресов классов A, B и C имеют фиксированный размер. Однако орга зация может указать собственный размер для поля подсети, и тут возникает вопрос о распознава этой части в хостах и маршрутизаторах. На рис. 5.8 показано меню программы *Chameleon* для вв размера поля подсети.

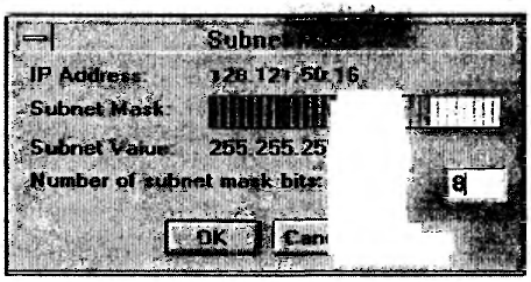

Рис. 5.8. Конфигурирование маски подсети

Размер поля подсети реально хранится в конфигурационном параметре, называемом *маской подсе* (subnet mask). Маска подсети имеет длину в 32 бита. Эти биты отражают для заданной сети длину по подсети в адресе: для поля подсети в маске располагаются единицы, а для системного поля — нул

Например, если для идентификации подсети используется третий байт, а сеть имеет адрес 128.1 то маска подсети будет:

11111111 11111111 11111111 00000000

Часто маска подсети записывается десятичной нотацией с точками:

255.255.255.0

Иногда применяется шестнадцатеричный формат:

X'FF-FF-FF-00

Подключенные к подсети хосты и маршрутизаторы конфигурируются с маской подсети. Общепр нятым способом является использование одной маски подсети для всей интернет-сети организаци Однако из этого правила есть исключения, и некоторые организации применяют несколько размер для различных подсетей.

Например, если сеть содержит большое количество линий "точка-точка", то номера подсети буд использованы очень неэкономно, поскольку в коммуникации участвуют только две системы в кажд из подсетей "точка-точка". Организация может решить использовать 14-битовую маску (255.255.255.25 для соединений "точка-точка".

Подсети в се

Биты подсети	К
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	

В таблице 5.
иведено количе
едположить, по
Например, ес

11111111 11

о можно записа
15 (1 бит для п
В приложени
дсетей, что поз

.18 Специ

Для номера п
ат для широкове
ководствоваться
ак и в поле хост

.18.1 Идент

Для указания
и заполнении
1.18.0.0 — на се
Аналогичным
ску подсети, т
писи сети или
тому, что таки
ля как номера
не применение
ль как маску п

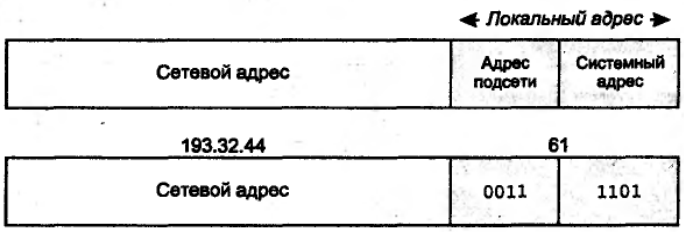

Рис. 5.7. Четырехбитовая часть для подсети в адресе класса С

Организация с адресом класса С имеет только однобайтовое адресное пространство. Она может вообще не проводить деления на подсети или использовать первые 4 бита для адреса подсети и 4 бита для адресов хостов (см. рис. 5.7). На рисунке видно, что локальный адрес (61) имеет двоичное представление 0011 1101. Первые 4 бита идентифицируют подсеть, а последние 4 бита определяют системы.

5.17 Маска подсети

Маршрутизация трафика на хост выполняется посредством анализа сетевой части и части для подсети IP-адреса. Сетевые части адресов классов A, B и C имеют фиксированный размер. Однако организация может указать собственный размер для поля подсети, и тут возникает вопрос о распознавании этой части в хостах и маршрутизаторах. На рис. 5.8 показано меню программы *Chameleon* для ввода размера поля подсети.

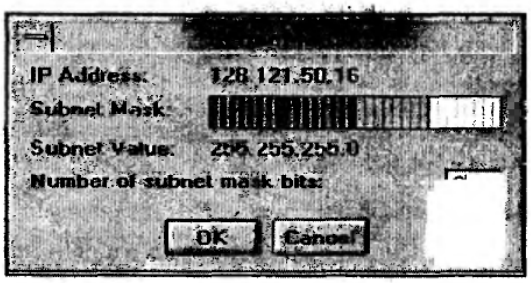

Рис. 5.8. Конфигурирование маски подсети

Размер поля подсети реально хранится в конфигурационном параметре, называемом *маской подсети* (subnet mask). Маска подсети имеет длину в 32 бита. Эти биты отражают для заданной сети длину поля подсети в адресе: для поля подсети в маске располагаются единицы, а для системного поля — нули.

Например, если для идентификации подсети используется третий байт, а сеть имеет адрес 128.121, то маска подсети будет:

11111111 11111111 11111111 00000000

Часто маска подсети записывается десятичной нотацией с точками:

255.255.255.0

Иногда применяется шестнадцатеричный формат:

X'FF-FF-FF-00

Подключенные к подсети хосты и маршрутизаторы конфигурируются с маской подсети. Общепринятым способом является использование одной маски подсети для всей интернет-сети организации. Однако из этого правила есть исключения, и некоторые организации применяют несколько размеров для различных подсетей.

Например, если сеть содержит большое количество линий "точка-точка", то номера подсети будут использованы очень неэкономно, поскольку в коммуникации участвуют только две системы в каждой из подсетей "точка-точка". Организация может решить использовать 14-битовую маску (255.255.255.252) для соединений "точка-точка".

Таблица 5.2

Подсети в сети класса B

Биты подсети	Количество подсетей	Биты для хостов	Количество хостов	Маска подсети
0	0	16	65 534	255.255.0.0
1	-	15	-	Недопустимая комбинация
2	2	14	16 382	255.255.192.0
3	6	13	8 190	255.255.224.0
4	14	12	4 094	255.255.240.0
5	30	11	2 046	255.255.248.0
6	62	10	1 022	255.255.252.0
7	126	9	510	255.255.254.0
8	254	8	254	255.255.255.0
9	510	7	126	255.255.255.128
10	1 022	6	62	255.255.255.192
11	2 046	5	30	255.255.255.224
12	4 096	4	14	255.255.255.240
13	8 190	3	6	255.255.255.248
14	16 382	2	2	255.255.255.252
15	-	1	-	Недопустимая комбинация

В таблице 5.2 показаны способы разделения локального адреса для сети класса B. В ней также приведено количество подсетей и хостов в разделах. Это количество на 2 меньше, чем можно было предположить, поскольку существуют некоторые ограничения, которые будут рассмотрены ниже.

Например, если подсеть использует 6 бит, шаблон маски подсети будет:

11111111 11111111 11111100 00000000,

что можно записать как 255.255.252.0. Далее мы рассмотрим, почему нельзя использовать комбинации 1/15 (1 бит для подсети и 15 бит для адресов хостов) и 15/1.

В приложении D представлены примеры использования в одной сети нескольких различных масок подсетей, что позволяет эффективно присваивать адреса.

5.18 Специальные зарезервированные адреса

Для номера подсети или хоста нельзя использовать любое число. Например, некоторые адреса служат для широковещательных рассылок, а другие — резервируются для таблиц маршрутизации. Следует руководствоваться правилом: *никогда не применять блоки из одних нулей или единиц — как в поле подсети, так и в поле хостов.* Также не существует сетевых номеров, состоящих из одних нулей или единиц.

5.18.1 Идентификация сети и подсети

Для указания сети удобно использовать формат адреса с точками. По соглашению, это делается при заполнении локальной части адреса нулями. Например, 5.0.0.0 указывает на сеть класса A, 131.18.0.0 — на сеть класса B, а 201.49.16.0 — на сеть класса C.

Аналогичным образом указываются подсети. Например, если сеть 131.18.0.0 использует 8-битовую маску подсети, то 131.18.5.0 и 131.18.6.0 будут определять подсети. Эта же нотация применяется для записи сети или подсети назначения в таблице маршрутизации IP. Данное соглашение приводит к тому, что такие адреса нельзя присваивать хостам и маршрутизаторам. Кроме того, использование нуля как номера подсети делает некоторые адреса неоднозначными, например 130.15.0.0. По этой причине применение нулей в поле подсети запрещено в стандарте (см. RFC 1122). Сайты, использующие ноль как маску подсети, тем самым нарушают соглашение.

5.18.2 Широковещательная рассылка в локальной подсети

Несколько IP-адресов используется для указания на широковещательную рассылку. В такой рассылке датаграммы можно направить на заданный набор систем в пределах ограниченной области.

IP-адрес 255.255.255.255 (т.е. адрес, содержащий 32 единицы) рассылает датаграмму всем системам локальной связи. (Некоторые продукты, и в частности BSD 2.4 TCP/IP, используют для широковещательных рассылок нули вместо единиц. Это нестандартизованный способ, и с течением времени такие операционные системы должны быть заменены на правильные.) Такие широковещательные рассылки применяются, например, в протоколах *BOOTP* и *DHCP*, когда при загрузке система запрашивает для себя IP-адрес и инициализационные данные у загрузочного сервера. Клиент посылает boot-запрос по адресу 255.255.255.255 и использует зарезервированный адрес 0.0.0.0 как IP-адрес источника.

Широковещательные рассылки в локальных сетях реализуются путем обрамления IP-датаграммы кадром, заголовок которого содержит в поле адреса назначения все единицы, что соответствует физическому адресу широковещательной рассылки.

5.18.3 Широковещательные рассылки к подсети

Широковещательную рассылку можно направить к заданной подсети, которая непосредственно подключена к подсети-источнику или может быть удаленной подсетью для хоста источника. Например, если 131.18.7.0 является подсетью сети класса B, то для широковещательного сообщения ко всем узлам этой подсети нужно использовать адрес 131.18.7.255.

Если подсеть назначения является удаленной, то в результате отправки датаграммы IP по широковещательному адресу одна ее копия будет предназначена маршрутизатору, подключенному к сети 131.18.7.0. Предполагая, что подсеть является локальной, маршрутизатор применит адрес физической широковещательной рассылки в поле назначения кадра MAC для пересылки сообщения всем хостам подсети.

Отметим, что при этом подразумевается отсутствие у систем зарезервированного IP-адреса 130.18.7.255.

5.18.4 Широковещательные рассылки в сети

Допустимо посылать датаграмму IP на каждый хост заданной удаленной сети. Это выполняется при установке всей локальной части адреса в единицы. Например, если администратору нужно послать объявление на все узлы сети 201.49.16.0 класса C с топологией Ethernet, то для такой широковещательной рассылки подойдет IP-адрес:

201.49.16.255

Однако этот адрес не должен быть присвоен ни одному из хостов.

Адрес 131.18.255.255 должен применяться для отправки сообщения на все узлы сети класса C. Отметим, что, хотя и допустимо присваивать номер 255 одной из подсетей, это приведет к проблемам: неясно, предназначена ли широковещательная рассылка 130.15.255.255 для подсети или для всей сети. Чтобы исключить такие ситуации, никогда не следует присваивать номера из всех единиц (например, 255) для подсетей.

5.18.5 Ограничения на IP-адрес

Набор доступных IP-адресов существенно сокращается из-за применения специальных форматов для широковещательных рассылок и таблиц маршрутизации. Стандарт RFC 1122 *Requirements for Internet Hosts — Communication Layers* (Требования к хостам Интернета — уровни взаимодействия) гласит:

- Поля сети, подсети или хоста не должны содержать одни нули.
- Поля сети, подсети или хоста не должны содержать одни единицы.

Следовательно, на практике поле должно быть длиной не менее 2 бит.

5.18.6 Кольцевой адрес

Полной противоположностью широковещательной рассылке является метод, когда сообщение вообще не покидает хоста. Существует множество хостов, совмещающих функции клиента и сервера. Локальные сервер и хост взаимодействуют друг с другом через IP внутри данного компьютера. Для этого служит специальный адрес, называемый *кольцевым* (loopback). По соглашению, для этого используется любой адрес, начинающийся на 127. На практике обычно применяют только адрес 127.0.0.1. Отметим, что для такого адреса резервируется адресное пространство целой сети класса C.

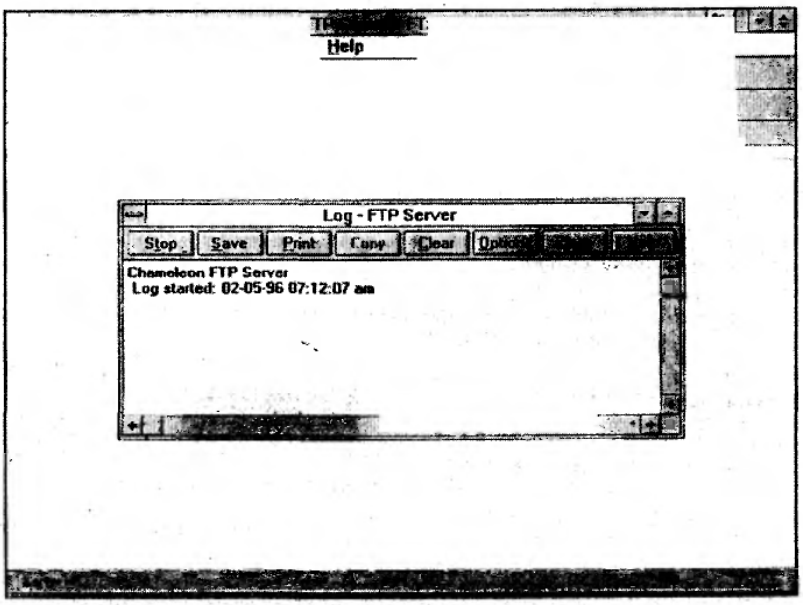

Рис. 5.9. Сервер FTP в среде Windows

Работу кольцевого адреса легко увидеть. Например, клиент и сервер FTP программы *Chameleon* могут одновременно соединяться в среде *Microsoft Windows*. После запуска сервера выводится экран, показанный на рис. 5.9.

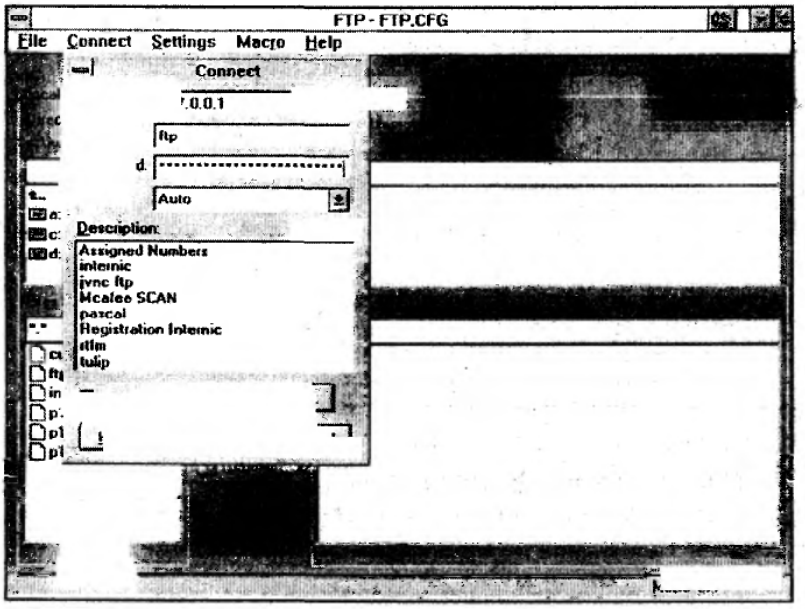

Рис. 5.10. Клиент FTP соединяется с локальным сервером

Клиент соединяется с сервером посредством кольцевого адреса 127.0.0.1 (см. рис. 5.10). Любые выполняемые клиентом пересылки файлов просто копируют файлы из одного каталога персонального

компьютера в другой каталог того же компьютера. Журнал регистрации сервера позволяет записать выполняемые при этом операции с адресом 127.0.0.1 (см. рис. 5.11).

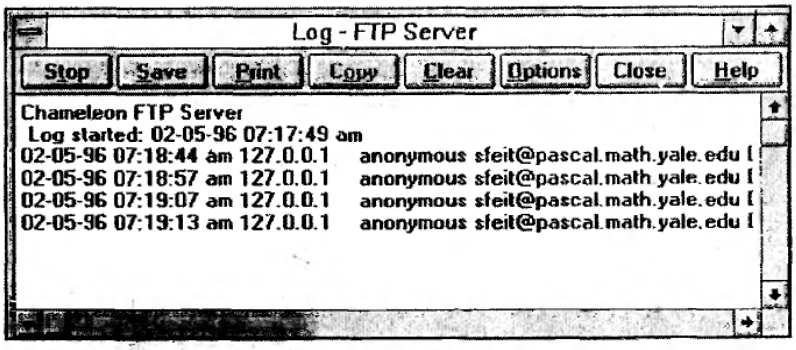

Рис. 5.11. Операции клиента с сервером FTP

5.18.7 Заключение о зарезервированных специальных адресах

Различные типы специальных адресов представлены в таблице 5.3.

Таблица 5.3

Специальные адреса

Адреса	Описание
0.0.0.0	Используется как адрес источника в конфигурационном запросе при загрузке. Также отмечает маршрутизатор по умолчанию в таблице маршрутизации.
127.0.0.0	Зарезервирован
127.0.0.1	Кольцевой адрес. Клиентом и сервером является один и тот же хост.
127.0.0.2-127.255.255.255	Зарезервированы
128.0.0.0	Зарезервирован
191.255.0.0	Зарезервирован
192.0.0.0	Зарезервирован
255.255.255.0	Зарезервирован
240.0.0.0-255.255.255.254	Зарезервированы
255.255.255.255	Широковещательная рассылка на локально подключенные локальные сети.

5.19 Суперсети и CIDR

Методы присваивания адресов с использованием классов A, B и C крайне неэффективны. Адрес класса C предоставляет не более 254 доступных вариантов (0 и 255 нельзя использовать как адреса узлов). С другой стороны, если организации требуется несколько сотен или тысяч адресов, то ей нужно присвоить адрес класса B, и многие адреса такого пространства не будут задействованы.

Больший смысл имеет побитовое выделение адресного пространства в соответствии с реальными потребностями организации. Это сделать очень просто. Например, если организации нужно 4000 адресов, то ей предоставляется 12 бит для применения в локальной части ее адресного пространства. Оставшиеся 20 бит образуют фиксированный префикс, используемый как адрес новой *суперсети* или *префиксной* части адреса. Общепринятым способом указания размера такой бесклассовой части адреса является */20*.

Первоначально выделение адресов для суперсетей производилось из доступного пространства номеров класса С. Получение 20-битового префикса эквивалентно получению 16 последовательных адресов класса С.

Таблица 5.4

Блоки CIDR из адресного пространства класса С

Размер сетевой части	Количество бит в локальной части	Эквивалентное число сетей класса С	Количество адресов для организации
/24	8	1	256
/23	9	2	512
/22	10	4	1 024
/21	11	8	2 048
/20	12	16	4 096
/19	13	32	8 192
/18	14	64	16 384
/17	15	128	32 768

В таблице 5.4 показаны различные адресные блоки, которые могут присваиваться из адресного пространства класса С. Для направления информации в организацию с такими адресами маршрутизатор Интернета должен знать:

- Количество бит в сетевом префиксе
- Реальный битовый шаблон, присвоенный как сетевой префикс для организации

После этого маршрутизатор может направлять трафик в организацию, используя единственную строку из своей таблицы маршрутизации. Такой механизм называется *маршрутизацией бесклассовых доменов Интернета* (Classless Internet-Domain Routing — CIDR).

Неиспользуемые части пространства номеров класса А могут быть поделены аналогичным способом. Организации должна быть присвоена строка бит как сетевой префикс, а оставшиеся биты можно применять для номеров систем этой организации. Все, что нужно,— это провести работу по включению длины сетевого префикса в информацию о маршрутизации.

Маршрутизация Интернета является более эффективной благодаря делегированию больших адресных блоков провайдерам. Далее провайдер присваивает подблоки адресов своим клиентам. Трафик маршрутизируется к провайдеру с помощью выделенного тому префикса блока. Затем провайдер использует более длинный префикс для маршрутизации трафика к своим клиентам.

Например, провайдеру может быть выделен блок, начинающийся с 10-битового префикса 11000001 11, а одному из клиентов можно присвоить блок, начинающийся с 16-битового префикса 11000001 11011111.

5.20 Необходимость следующего поколения протокола IP

Внедрение бесклассовых адресов суперсетей и бесклассовой маршрутизации стало последней точкой в совершенствовании и использовании текущей схемы адресации протокола IP.

В начале разработки адресов IP никто не мог предположить, что развитие технологий приведет к появлению компьютеров на рабочих местах, в квартирах, что сами компьютеры станут бытовыми приборами, а сети соединят их всех. Текущая схема адресации неудобна и неадекватна выполняемым функциям.

В отличие от иерархической структуры телефонных номеров адреса были разработаны без использования кодов стран или областей, что делает маршрутизацию достаточно сложной. Маршрутизаторы региональных сетей должны хранить сведения о десятках тысяч отдельных сетей.

Для решения данных проблем был разработан протокол IP версии 6 (*Next Generation*), обеспечивающий новые пути в использовании компьютеров и сетей (эта версия рассматривается в главах 22 и 23).

5.21 IP-адреса, интерфейсы и множественное пребывание

Идентификация сетей и подсетей в IP-адресе имеет много достоинств:

- Упрощается работа по присваиванию адресов. Блок адресов можно делегировать для администрирования в отдельной сети или подсети.
- Сокращаются таблицы маршрутизации, которые содержат только краткий список сетей и подсетей, а не список всех хостов интернета.
- Упрощается маршрутизация. Просмотр номеров сетей и подсетей выполняется быстрее и эффективнее.

Это важные достоинства, но существуют и важные следствия применения такой адресной схемы. Рассмотрим рис. 5.12. Маршрутизатор имеет три различных интерфейса, а соединен с двумя локальными сетями и выделенной линией.

Маршрутизатор соединен с внутренними сетями 128.36.2 и 128.36.18, а также с внешней сетью 193.92.45. Так каков же будет IP-адрес этого маршрутизатора?

Ответ прост: *системы* не имеют IP-адресов — адреса присваиваются *интерфейсам* этих систем. Каждый интерфейс имеет IP-адрес, начинающийся с номера сети или подсети, подключенной к локальной или региональной сети. В нашем случае маршрутизатор имеет три интерфейса и три IP-адреса.

Хост также может подключаться более чем к одной сети или подсети. На рис. 5.12 хост имеет интерфейсы для двух сетей Ethernet и два IP-адреса: 128.36.2.51 и 128.36.5.17.

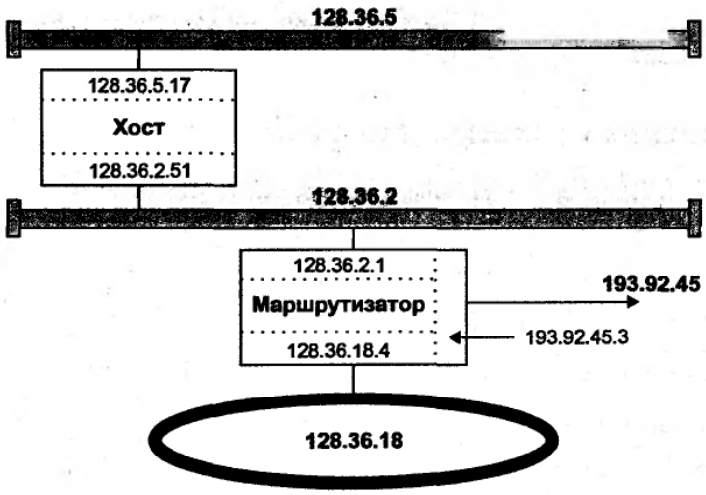

Рис. 5.12. Присвоение IP-адресов интерфейсам

Системы, подключенные более чем к одной подсети, называются *многоадресными* (multihomed). (Отметим, что в WWW этот же термин означает размещение на одном сервере нескольких сайтов и обычно переводится как "множественное присутствие".— *Прим. пер.*) Многоадресный хост вносит определенные сложности в маршрутизацию IP. Данные к такому хосту направляются по разным путям, в зависимости от выбранного для коммуникации IP-адреса. Было бы более приемлемо связать с таким хостом несколько имен, соответствующих различным интерфейсам. Например, пользователи локальной сети 128.36.2 могут взаимодействовать с иным именем хоста, чем пользователи локальной сети 128.36.5 (см. рис. 5.12).

Вопреки недостаткам многоадресных хостов, включение в адрес идентификаторов сетей и подсетей существенно улучшает эффективность маршрутизаторов и позволяет легко расширять сети интернета, работающие по протоколу TCP/IP.

5.22 Конфигурирование адресов и масок подсети

Как мы уже знаем, пользовательский интерфейс конфигурирования TCP/IP различается на разных хостах. В системе *tigger* команда *ifconfig* используется для установки или просмотра связанных с интерфейсом параметров. Ниже показаны параметры Ethernet интерфейса 0 (*le0*):

```
> ifconfig le0
le0: flags = 63 <UP,BROADCAST,NOTRAILERS,RUNNING>
     inet 128.121.50.145 netmask ffffff00 broadcast 128.121.50.255
```

IP-адрес интерфейса — 128.121.50.145. Маска подсети выведена в шестнадцатеричном формате (ffffff00). Адресом широковещательной рассылки в этой подсети является 128.121.50.255.

Эти же сведения были введены через меню *Chameleon*. Например, раскрывающееся меню служит для конфигурирования IP-адреса (см. рис. 5.13).

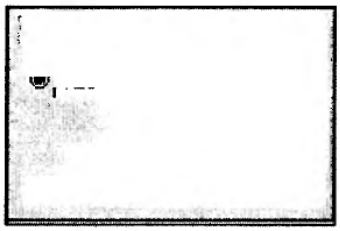

Рис. 5.13. Конфигурирование IP-адреса через меню

5.23 Взаимосвязь имен и адресов

Посмотрев на имя системы (*fermat.math.yale.edu*) и ее IP-адрес в нотации с точками (128.36.23.3), можно подумать, что части имени соответствуют номерам в нотации с точками. Однако на самом деле между ними нет никакой связи.

Действительно, иногда системам локальной сети присваивают имена, которые *выглядят* как соответствующие иерархии адресов. Однако:

- В той же локальной сети *могут* находиться имена, полностью нарушающие это правило.
- Хосты со сходной структурой имен *могут* располагаться в различных локальных сетях или различных сетях других типов.

Для примера рассмотрим следующие имена и адреса:

macoun.cs.yale.edu 128.36.2.5

bulldog.cs.yale.edu 130.132.1.2

Адреса отражают сетевую точку подключения и ограничены в расположении, а имена систем, напротив, не зависят от физического подключения к сети.

Организации могут расширять свои домены именами, подобными *chicago.sales.abc.com* или *newyork.sales.abc.com*. Соответствующие компьютеры могут располагаться в указанных городах (Чикаго или Нью-Йорке).

Трафик направляется в системы на основе адресов, а не имен, и адрес системы всегда определяется перед отправкой на нее данных. Следовательно, организации свободны в выборе гибкой схемы именования, которая будет лучше удовлетворять заданным требованиям.

5.24 Протокол ARP

Перед тем как датаграмма будет передана с одной системы локальной сети на другую, она будет обрамлена заголовком и завершающей частью кадра. Кадр доставляется на сетевой адаптер, физический адрес которого совпадает с физическим адресом назначения из заголовка кадра.

Таким образом, для доставки датаграммы в локальной сети нужно определить физический адрес узла назначения.

Хорошо, что существует процедура автоматического определения физических адресов. *Протокол разрешения адресов* (Address Resolution Protocol — ARP) обеспечивает метод динамической трансляции между IP-адресом и соответствующим физическим адресом на основе широковещательных рассылок.

Система локальной сети самостоятельно использует ARP для исследования информации о физических адресах (сетевой администратор при необходимости может вручную ввести в таблицу ARP постоянный элемент для такой трансляции). Когда хосту нужно начать коммуникацию со своим локальным партнером, он ищет IP-адрес партнера в таблице ARP, которая обычно располагается в оперативной памяти. Если для нужного IP-адреса не находится требуемого элемента таблицы, хост посылает широковещательный запрос ARP, содержащий искомый IP-адрес назначения (см. рис. 5.14).

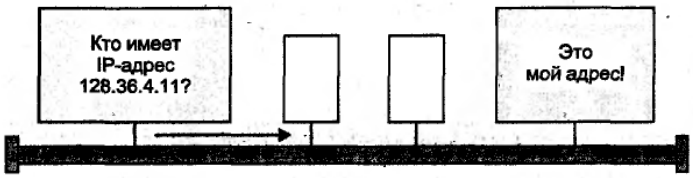

Рис. 5.14. Поиск физического адреса системы

Целевой хост узнает свой IP-адрес и читает запрос. После этого он изменяет собственную таблицу трансляции адресов, включая в нее IP-адрес и физический адрес отправителя широковещательной рассылки, и, наконец, посылает ответ, содержащий аппаратный адрес своего интерфейса.

Когда система-источник получает такой ответ, она обновляет свою таблицу ARP и становится готовой к пересылке данных по локальной сети.

5.24.1 Содержимое сообщения ARP

Запросы ARP первоначально использовались в локальных сетях Ethernet, но структура таких запросов имеет более общую природу, поэтому их можно применять и в Token-Ring, локальных сетях Fiber Distributed Data Interface (FDDI) или в глобальных сетях Switched Multimegabit Data Service (SMDS). Один из вариантов ARP был разработан для региональных сетей с виртуальными цепями (подобных Frame Relay).

Сообщение ARP помещается в поле данных кадра вслед за заголовком (заголовками) нижних уровней. Например, для Ethernet с кадрами DIX сообщение ARP следует за MAC-заголовком, а для сетей типа 802.3 или 802.5 — за MAC-заголовком, заголовком Logic Link Control (LLC) и подзаголовком Sub-Network Access Protocol (SNAP). Тип протокола для таких кадров (ARP через Ethernet) определяется кодом X'0806. В таблице 5.5 показаны поля сообщения ARP.

Таблица 5.5

Формат сообщения ARP

Количество октетов	Поле
2	Тип аппаратного адреса
2	Протокол адресации высокого уровня
1	Длина аппаратного адреса
1	Длина адреса высокого уровня
2	Тип сообщения: 00 01 = запрос, 00 02 = ответ
*	Аппаратный адрес источника
*	Адрес высокого уровня (IP) источника
*	Аппаратный адрес приемника
*	Адрес высокого уровня (IP) приемника

Таблица 5.6

Примеры сообщений для запросов ARP

Количество октетов	Поле	Описание
2	00 01	Ethernet
2	08 00	IP
1	06	Длина физического адреса в 6 октетов для Ethernet
1	04	Длина физического адреса IP
2	00 01	Запрос
6	02 07 01 00 53 23	Аппаратный адрес источника
4	80 24 04 12	Адрес высокого уровня источника
6	00 00 00 00 00 00	Аппаратный адрес назначения
4	80 24 04 0B	Адрес высокого уровня назначения

Длина последних четырех полей зависит от используемой технологии и применяемого протокола. Аппаратный адрес локальной сети 802.X содержит 6 октетов, а IP-адрес — 4 октета. В таблице 5.6 показаны примеры форматов сообщений, запрашивающих трансляцию IP-адресов в адреса Ethernet.

При ответе меняются роли источника и приемника. Например, адресом высокого уровня *источника* в ответе на запрос станет X'80-24-04-0B.

Применение ARP не ограничивается только TCP/IP: во втором поле также можно указать протокол, использующий ARP.

Первичный запрос ARP распространяется в широковещательной рассылке, поэтому любая система локальной сети может использовать сведения из такого запроса для обновления собственной таблицы данными о запрашивающей системе. Однако обычно система обновляет свою таблицу, только когда сама служит целевой системой запроса ARP.

5.24.2 Таблица ARP

Большинство систем обеспечивает для администратора следующие команды:

- Просмотр локальной таблицы ARP
- Ручное удаление или добавление элементов таблицы
- Загрузку в таблицу информации из конфигурационного файла

Диалог пользователя в процессе выполнения команды *arp* -a показывает как изменяется содержимое таблицы ARP системы *tigger* при соединении по *telnet* с хостом *mickey*, сведений о котором ранее не было в таблице. Отметим, что в выводе из команды указываются имена каждой системы, их IP-адреса и 6 октетов физического адреса (шестнадцатеричные числа, разделенные двоеточием).

```
> arp -a
nomad-eth0.jvnc.net (128.121.50.50) at 0:0:c:2:85:11
r2d2.jvnc.net (128.121.50.2) at 8:0:20:a:2c:3f
jim-mac.jvnc.net (128.121.50.162) at 8:0:7:6f:a6:65
tom-mac.jvnc.net (128.121.50.163) at 8:0:7:ff:96:9e
chip.jvnc.net (128.121.50.148) at 0:0:3b:86:6:4c
nisc.jvnc.net (128.121.50.7) at 8:0:20:11:d2:b7
nicol.jvnc.net (128.121.50.10) at 0:0:3b:30:32:34
minnie.jvnc.net (128.121.50.141) at 8:0:20:7:b5:da
>

> telnet mickey.jvnc.net
Trying 128.121.50.143 ...
Connected to mickey.jvnc.net.
Escape character is '^]' .
```

```
SunOS UNIX (mickey.jvnc.net)

login:
. . .
logout

> arp -a
nomad-eth0.jvnc.net (128.121.50.50) at 0:0:c:2:85:11
r2d2.jvnc.net (128.121.50.2) at 8:0:20:a:2c:3f
jim-mac.jvnc.net (128.121.50.162) at 8:0:7:6f:a6:65
tom-mac.jvnc.net (128.121.50.163) at 8:0:7:ff:96:9e
chip.jvnc.net (128.121.50.148) at 0:0:3b:86:6:4c
nisc.jvnc.net (128.121.50.7) at 8:0:20:11:d2:b7
nicol.jvnc.net (128.121.50.10) at 0:0:3b:80:32:34
minnie.jvnc.net (128.121.50.141) at 8:0:20:7:b5:da
mickey.jvnc.net (128.121.50.143) at 8:0:20:7:53:8f
>
```

5.24.3 Обратные запросы ARP

Один из вариантов ARP называется *обратным запросом* (reverse ARP — RARP) и служит для определения узлом *собственного* IP-адреса. Такие запросы предназначены для бездисковых рабочих станций и других устройств, которые получают конфигурационную информацию от сетевого сервера.

В обратном запросе ARP станция указывает собственный физический адрес и по широковещательной рассылке отправляет его, желая получить для себя IP-адрес. Для ответа на такие запросы сетевой сервер должен быть сконфигурирован с таблицей физических адресов и соответствующих им IP-адресов.

Обратные запросы ARP были вытеснены протоколом BOOTP и его улучшенной версией, названной *протоколом динамического конфигурирования хоста* (Dynamic Host Configuration Protocol — DHCP). Этот протокол гораздо мощнее и предоставляет больший набор конфигурационных параметров для систем TCP/IP (BOOTP и DHCP будут рассмотрены в главе 11).

5.25 Множество адресов для одного интерфейса

Некоторые производители маршрутизаторов предусматривают возможность присваивать *несколько* IP-адресов одному интерфейсу маршрутизатора. Для чего же это нужно? Несколько адресов подсетей могут потребоваться, во-первых, в локальной сети, имеющей очень большое количество систем. Во-вторых, в тех случаях, когда отдельные номера подсетей применяются для создания различных правил фильтрации трафика для систем из двух различных рабочих групп. Причем каждая рабочая группа принадлежит отдельной *логической* подсети, хотя они совместно используют один *физический* носитель информации.

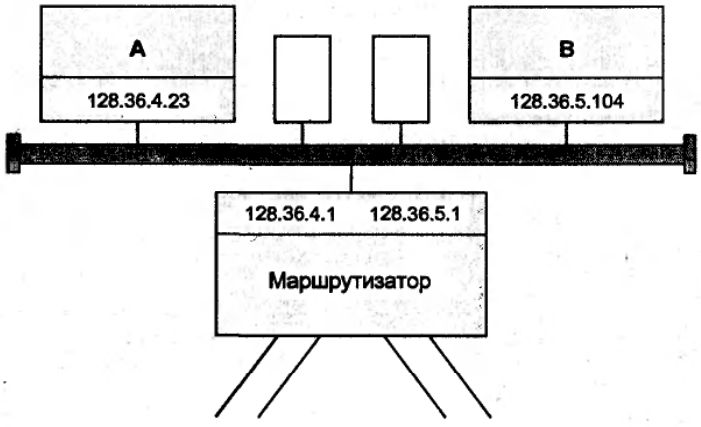

Рис. 5.15. Интерфейс маршрутизатора с двумя IP-адресами

На рис. 5.15 показана локальная сеть с двумя логическими подсетями — 128.36.4.0 и 128.36.5.0. Интерфейсу локальной сети маршрутизатора присвоено два IP-адреса: 128.36.4.1 и 128.36.5.1. В такой сети трафик будет успешно маршрутизироваться, однако потребуется дополнительная работа по правильной маршрутизации датаграмм, направленных на хосты этой сети.

Предположим, что система А имеет 8-битовую маску подсети. Когда А захочет послать датаграмму в В, она пошлет ее маршрутизатору. Чтобы этого избежать, хост локальной сети нужно сконфигурировать с 7-битовой маской подсети, при этом 4 будет соответствовать 0000 0100, а 5 — 0000 0101.

5.26 Прокси ARP

Предположим, что в сети нельзя использовать смежные номера. Например, 128.36.4.0 и 128.36.20.0 совместно используют носитель. В этом случае хосты локальной сети можно конфигурировать с маской 255.255.0.0, т.е. без выделения подсетей. Затем хосты смогут использовать ARP для *всех* точек назначения сети 128.36. Этот метод прекрасно подходит для сетей с совместным использованием носителя, но что делать с трафиком в подсеть сети 128.36, которая не принадлежит общей локальной сети?

Маршрутизатор локальной сети будет управлять внешним трафиком в том случае, если он поддерживает *прокси* (прокси иногда называют посредником. — *Прим. пер.*) *ARP* (Proxy ARP). При обнаружении запросов ARP, направляемых в точки назначения, которые являются для локальной сети внешними, маршрутизатор пошлет ответ ARP, содержащий физический адрес самого *маршрутизатора.* Если в локальной сети несколько маршрутизаторов, выбирается тот, у которого будет наилучший путь для ответа на запрос о точке назначения. Хосту потребуется заключить датаграмму в кадр и переслать ее маршрутизатору, который перешлет ее дальше.

5.27 Многоадресные рассылки

Широковещательные рассылки в IP позволяют доставить датаграмму на все системы сети или подсети. Вариант с большей избирательностью называется *многоадресной* (multicasting) рассылкой. В этом случае датаграммы пересылаются группе систем (см. рис. 5.16).

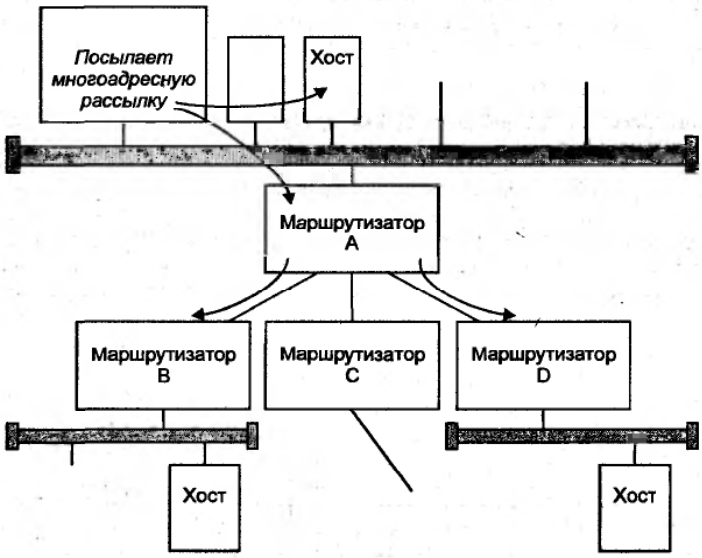

Рис. 5.16. Распространение датаграммы в многоадресной рассылке

Многоадресные рассылки в IP — очень полезный сетевой инструмент. Например, одно сообщение может использоваться для одновременного обновления конфигурационных параметров однородной группы хостов или для задания статуса группы маршрутизаторов. Многоадресные рассылки служат основой приложений для пользовательских конференций.

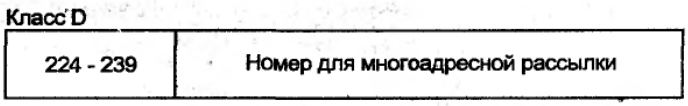

Рис. 5.17. Адрес класса D для многоадресной рассылки в IP

Для многоадресной рассылки используются IP-адреса класса D, формат которых представлен на рис. 5.17. Определен протокол для стандартной многоадресной рассылки, однако число поддерживающих этот протокол хостов и маршрутизаторов в настоящее время ограничено. Возможно, через несколько лет это положение изменится.

5.27.1 Группы многоадресной рассылки

Группа многоадресной рассылки (multicast group) — это набор систем, которым присвоен IP-адрес многоадресной рассылки. Члены группы продолжают использовать собственные IP-адреса, однако они имеют возможность принимать данные, посланные в многоадресной рассылке. Любая система может принадлежать нескольким группам многоадресной рассылки или ни одной из них.

Адреса класса D для многоадресных рассылок находятся в диапазоне номеров от 224 до 239. Некоторые IP-адреса многоадресных рассылок являются постоянными (они перечислены в RFC о *присвоенных номерах* Интернета). К таким адресам относятся:

224.0.0.1	Все хосты локальной подсети
224.0.0.2	Все маршрутизаторы локальной подсети
224.0.0.5	Все маршрутизаторы, поддерживающие протокол Open Shortest Path First (OSPF)

Многоадресные рассылки могут применяться для временной группы систем, создаваемой или ликвидируемой по мере надобности, например для аудио- или видеоконференций.

Хост должен поддерживать несколько определенных функций, чтобы участвовать в одной или нескольких группах многоадресных рассылок:

- Реализацию команды для объединения с многоадресной группой и идентификации интерфейса, который будет отслеживать соответствующие адреса
- Распознавание на уровне IP многоадресной рассылки для входящих и исходящих датаграмм
- Кроме того, должна существовать команда, позволяющая хосту исключить себя из группы многоадресной рассылки

Многоадресные рассылки не ограничиваются только локальными сетями. Маршрутизаторы с программным обеспечением для таких рассылок способны распространять датаграммы IP среди систем интернета.

Для более эффективного выполнения рассылки маршрутизатор должен знать, принадлежит ли хост локальной сети одной из многоадресных групп. Кроме того, маршрутизаторам необходимо обмениваться информацией между собой для определения многоадресных групп в удаленных сетях, куда должны направляться датаграммы.

Хосты используют *протокол обслуживания групп Интернета* (Internet Group Management Protocol — IGMP) для отчета о своем членстве в группе перед ближайшим маршрутизатором, поддерживающим многоадресные рассылки. Такой отчет посылается по IP-адресу многоадресной рассылки, присвоенному данной группе. Маршрутизатор не транслирует такой отчет вне пределов локальной сети, поэтому он будет услышан только маршрутизаторами и другими членами локальной группы.

Так как протокол IGMP предполагает полноту информации о членстве в группе, то он разрешает маршрутизаторам периодически опрашивать хосты о членстве в различных текущих группах. Опрос проводится по IP-адресу многоадресной рассылки 224.0.0.1 на все хосты.

5.27.2 Трансляция многоадресных рассылок в адреса Ethernet и FDDI

Физическим интерфейсам локальных сетей Ethernet и FDDI могут присваиваться один или несколько адресов для многоадресных рассылок. Это логическое присваивание предполагает выбор из нескольких подходящих для этого значений, что существенно упрощает трансляцию IP-адресов многоадресных рассылок в физические адреса таких рассылок. Отметим, что для этого не нужен протокол ARP.

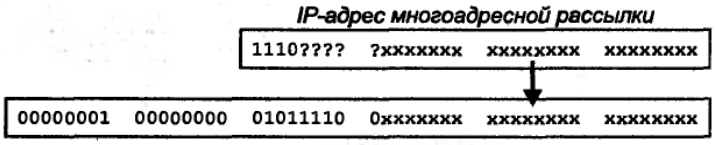

Рис. 5.18. Отображение части IP-адреса на физический адрес

Для локальных сетей Ethernet и FDDI применяются следующие правила:

- Первые 3 октета физического адреса для многоадресной рассылки имеют значение 01-00-5E.
- Следующий далее бит должен быть установлен в 0, а последние 23 бита должны иметь значение младших 23-х битов IP-адреса многоадресной рассылки.

Такое отображение показано на рис. 5.18:

- Последние 23 бита IP-адреса многоадресной рассылки отмечены как "x". Эти биты копируются в младшие биты физического адреса многоадресной рассылки.
- Отмеченные символами "?" позиции IP-адреса многоадресной рассылки могут быть заполнены произвольными битами. Они не копируются в физический адрес многоадресной рассылки.

Таким образом, три IP-адреса многоадресной рассылки

11100000 00010001 00010001 00010001

11100000 10010001 00010001 00010001

11100001 10010001 00010001 00010001

будут отображаться на *один и тот же* физический адрес многоадресной рассылки:

00000001 00000000 01011110 00010001 00010001 00010001

Интерфейсы систем, принадлежащих одной из трех групп, будут реагировать на многоадресные рассылки в своих группах. Однако каждый из хостов на уровне IP будет отбрасывать (игнорировать) посторонние многоадресные рассылки.

Хорошим способом исключения дополнительной обработки является выбор адресов многоадресных рассылок, в которых в позициях "?" стоят нули. При этом все равно остается 2^{23} (примерно 9 млн.) адресов для многоадресных рассылок.

5.27.3 Трансляция адресов многоадресных рассылок в адреса Token-Ring

К сожалению, рассмотренную выше схему для Ethernet и FDDI почти никогда нельзя применить в Token-Ring (по крайней мере, на момент написания этой книги), поскольку многие аппаратные интерфейсы Token-Ring не могут быть сконфигурированы на произвольные адреса многоадресных рассылок. Следовательно, остается применить один из трех методов трансляции (в зависимости от оборудования):

1. Вставить 23 бита IP-адреса многоадресных рассылок (этот метод рассмотрен выше)
2. Выбрать и использовать один из *функциональных* (functional) адресов Token-Ring
3. Применить *широковещательную рассылку по всему кольцу* Token-Ring

Существует 31 *функциональный* физический адрес. Они применяются для идентификации систем со специальными свойствами (например, мостов, концентраторов кольцевых подключений или мониторов ошибок в кольце). При выборе второго метода многоадресную рассылку нужно направить по функциональному физическому адресу:

03-00-00-20-00-00

Когда станция получит кадр, содержащий датаграмму многоадресной рассылки, по IP-адресу будет проверено, действительно ли станция является членом группы многоадресной рассылки.

Поскольку один функциональный адрес применяется для всех адресов многоадресных рассылок, такой метод не очень эффективен. Однако он гораздо лучше, чем третий вариант, когда используется широковещательная рассылка по всем станциям.

5.28 Дополнительная литература

Классы адресов определены в стандарте IP RFC 791. Выделение подсетей описывается в RFC 950, а формирование суперсетей — в RFC 1519. Широковещательные рассылки рассмотрены в RFC 919 и RFC 922.

Протокол Address Resolution Protocol специфицирован для Ethernet в RFC 826. Обратные ARP обсуждаются в RFC 903.

RFC 1112 посвящен многоадресным рассылкам в IP. RFC 1390 определяет трансляцию между IP-адресами многоадресных рассылок и адресами FDDI. RFC 1469 специфицирует трансляцию между IP-адресами многоадресных рассылок и адресами Token-Ring.

RFC 1178 содержит как серьезные, так и не совсем серьезные советы по выбору имени для компьютера. RFC 1034 и 1101 подробно обсуждают именование доменов. RFC 1035 описывает протоколы для создания Domain Name System и реализацию этой системы.

Стандарт Hosts Requirements (Требования к хостам), RFC 1122, предоставляет дополнительные сведения об именовании и адресации, равно как и корректирует неточности в некоторых стандартах.

Глава

6

Протокол интернета

6.1 Введение

Вспомним, что интернет — это набор сетей, соединенных маршрутизаторами (во многих ранних документах RFC использовался термин "шлюз" вместо "маршрутизатор"), а IP — это протокол сетевого уровня, обеспечивающий маршрутизацию данных в интернете. При создании IP исследователи и разработчики руководствовались следующими требованиями Министерства обороны США:

- Приспособить к взаимодействию хосты и маршрутизаторы различных производителей
- Объединить расширяющееся множество сетей различного типа
- Обеспечить расширение сети без прерывания работы сетевых служб
- Реализовать поддержку высокоуровневых сеансов и служб, ориентированных на сообщения

Всем этим требованиям удовлетворяет архитектура сетевого уровня IP.

Более того, она позволяет интегрировать островки локальных сетей (разбросанных по различным организациям) таким образом, чтобы обеспечить подключение новых островков без изменений в уже объединенных.

Все это сделало IP основным сетевым протоколом для правительственных агентств, университетов и коммерческих организаций.

6.2 Датаграммы IP

Протокол IP предоставляет механизм для пересылки по интернету элементов, называемых *датаграммами IP* (IP datagram). Как показано на рис. 6.1, датаграмма IP формируется из заголовка IP и перемещаемой по сети порции данных.



Рис. 6.1. Формат датаграммы

Протокол IP можно назвать "протоколом наилучшей попытки". Это означает, что IP гарантирует не целостность доставки датаграммы в пункт назначения, а только наилучшую попытку выполнить доставку (см. рис. 6.2). Датаграмма может разрушиться по следующим причинам:

- Ошибка в одном из битов во время пересылки в носителе.
- Перегруженный маршрутизатор отбросил датаграмму, чтобы освободить свое буферное пространство.
- Временно недоступен путь к точке назначения.

Все операции по обеспечению надежности доставки данных осуществляются на уровне TCP. Восстановление испорченных данных зависит от действий на этом уровне.

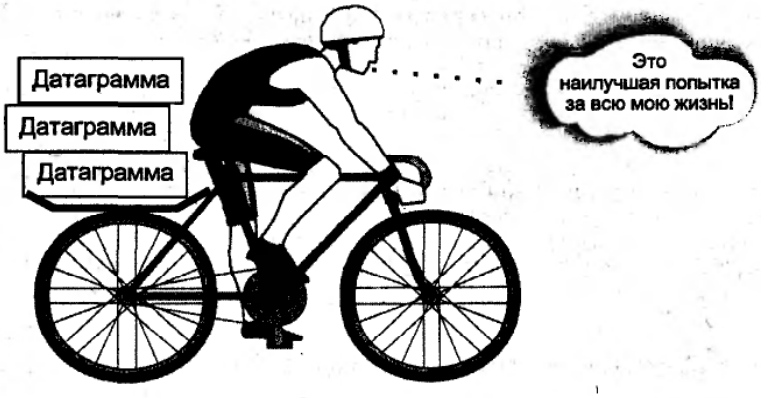

Рис. 6.2. Доставка в IP по принципу наилучшей попытки

6.3 Основные функции IP

Основными функциями IP являются: прием данных от TCP или UDP, создание датаграммы, маршрутизация ее по сети и доставка приложению-получателю. Каждая датаграмма IP маршрутизируется отдельно. Для маршрутизации датаграммы в IP существуют два средства:

- *маска подсети*
- *таблица маршрутизации* IP (таблица маршрутов)

6.4 Использование маски подсети

Предположим, что компьютер имеет IP-адрес 130.15.12.131 и подключен к локальной сети, а данные нужно послать:

Из: 130.15.12.131

В: 130.15.12.22

Можно предположить, что обе системы находятся в одной и той же подсети. Компьютер должен проверить, верно ли такое предположение. Проверка выполняется по маске подсети. Допустим, что хост имеет маску подсети:

255.255.255.0

т.е. есть маска состоит из 24 единиц и 8 нулей:

11111111111111111111111100000000

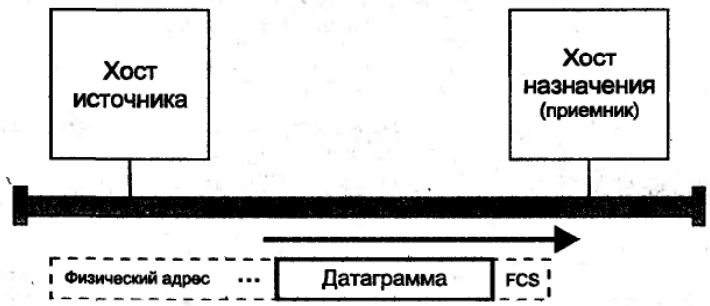

Рис. 6.3. Обрамление кадром и передача датаграммы

Вспомним, что единицы в маске подсети идентифицируют *сеть и часть адреса для подсетей*. Так как части для сети и подсети в адресах источника и назначения — 130.15.12, значит оба хоста находятся в одной подсети.

Компьютер фактически выполняет операцию "логическое И" между маской и каждым из IP-адресов. В результате нули маски подсети очищают часть адреса для хоста, оставляя только части для сети и подсети.

В этом примере маршрутизация является *прямой*. Это означает, что датаграмма должна быть помещена в кадр и передана непосредственно в точку назначения локальной сети, как показано на рис. 6.3.

Адрес назначения, помещенный в заголовок кадра, должен быть физическим адресом системы назначения. Чтобы определить существование элемента для физического адреса 130.15.12.22, проверяется таблица протокола ARP. Если в таблице нет нужной записи, для ее формирования используется протокол ARP.

6.5 Хост в таблице маршрутизации IP

Предположим, что нужно переслать данные:

Из: 130.15.12.131

В: 192.45.89.5

Быстрая проверка маски подсети показывает, что система назначения *не принадлежит* локальной подсети. В этом случае IP должен обратиться к локальной таблице маршрутизации.

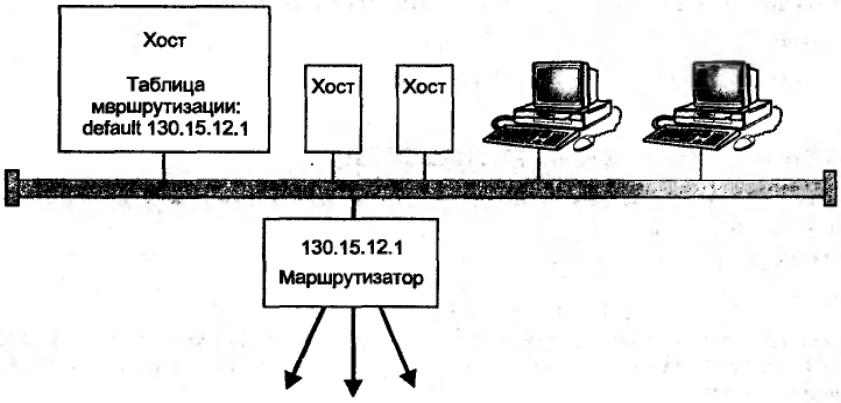

Рис. 6.4. Перенаправление трафика через маршрутизатор по умолчанию

Таблица маршрутизации хоста обычно очень проста. На рис. 6.4 показана локальная сеть, которая связана с удаленными сайтами посредством единственного маршрутизатора. Если точка назначения не находится в локальной сети, у хоста нет другого выбора, как обратиться к маршрутизатору.

Каждый настольный компьютер или хост локальной сети имеет таблицу маршрутизации, которая сообщает IP, как маршрутизировать датаграммы к системам, не подключенным к локальной сети. Для указания пути к удаленному месту эта таблица нуждается в единственной записи (для маршрутизации по умолчанию):

default 130.15.12.1

Другими словами, *нужно направлять любые нелокальные датаграммы на маршрутизатор по умолчанию с IP-адресом 130.15.12.1* (отметим, что адрес назначения 0.0.0.0 используется в таблице маршрутизации для значения по умолчанию).

6.6 Маршрутизация по следующему попаданию

Для сохранения простоты таблицы маршрутизации хоста IP может не анализировать полный маршрут к точке назначения. *Требуется только выяснить следующее попадание* (next hop иногда переводится как следующий участок.— *Прим. пер.*) *и направить датаграмму туда.*

Чтобы отправить датаграмму на интерфейс маршрутизатора 130.15.12.1, ее надо поместить в кадр, заголовок которого содержит физический адрес сетевого адаптера этого маршрутизатора.

Когда маршрутизатор получит кадр, он удалит заголовок и завершающую часть кадра, а также исследует заголовок датаграммы IP, чтобы решить, куда ее нужно направить далее.

6.7 Еще один пример таблицы маршрутизации хоста

Иногда таблицы маршрутизации хостов не столь просты. Рассмотрим, например, два маршрутизатора подсети 128.121.50.0 (см. рис. 6.5). Второй маршрутизатор управляет небольшой локальной сетью с несколькими рабочими станциями.

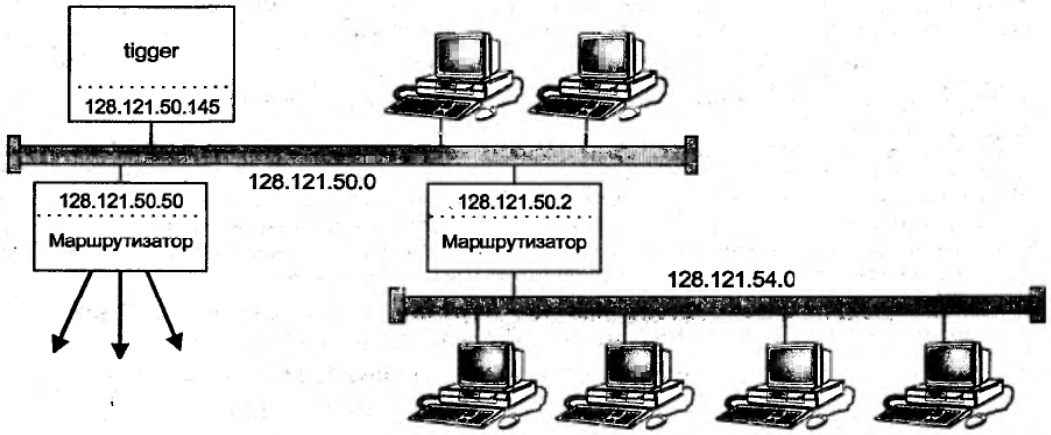

Рис. 6.5. Выбор маршрутизатора

Маршрутизатор *tigger* управляет локальной сетью, и его таблицу маршрутизации можно вывести командой *netstat -nr*. В выводе используется термин *шлюз — gateway*, а не *маршрутизатор — router*. (Другие компьютеры могут выводить таблицу в несколько ином формате. Она будет содержать похожую, но не идентичную информацию. Например, некоторые системы могут выводить столбец со сведениями о расстоянии до следующей точки назначения.)

```
> netsatat -nr
Routing tables
Destination    Gateway          Flags     Refcnt     Use          Interface

127.0.0.1      127.0.0.1          UH          6       62806          lo0
Default        128.121.50.50      UG         62     2999087          le0
128.121.54.0   128.121.50.2       UG          0           0          le0
128.121.50.0   128.121.50.145     U          33     1406799          le0
```

Командой *netstat* выводятся сведения о том, где и как будет маршрутизироваться трафик *tigger*:

- Первое место назначения в таблице — это *кольцевой* адрес 127.0.0.1, который служит обозначением для трафика между клиентами и серверами в пределах системы *tigger*.
- Запись *default* используется для выполнения маршрутизации к любой точке назначения, которая не указана в таблице. Трафик должен быть направлен на интерфейс маршрутизатора по IP-адресу 128.121.50.50.
- Датаграммы к любой системе подсети 128.121.54.0 должны быть направлены на интерфейс маршрутизатора по IP-адресу 128.121.50.2.
- Последняя запись не обеспечивает получения новой информации для маршрутизации, но позволяет получить интересную статистику о местном трафике. Чтобы маршрутизировать трафик к любой системе подсети 128.121.50.0, нужно направить его на адрес 128.121.50.145. При этом 128.121.50.145 — это собственный адрес *tigger*, а 128.121.50.0 — собственный адрес локальной сети *tigger*.

Команда *netstat* выводит и другую интересную информацию:

- *Флаги* (Flags) сообщают, является ли маршрут пригодным для использования и будет ли следующее попадание хостом (H) или шлюзом (G).
- *REFcnt* отслеживает текущее количество активных применений маршрута.
- Столбец *Use* подсчитывает число датаграмм, которые были посланы по маршруту (после последней инициализации).
- Интерфейс *lo0* является *логическим* интерфейсом для кольцевого трафика. Весь внешний трафик проходит через один интерфейс Ethernet — *le0*.

Отметим, что включение в отчет локальной подсети 128.121.50.0 позволило обнаружить, что посланный вовне трафик вдвое больше, чем трафик, направленный к системам локальной сети.

6.8 Правило просмотра таблицы маршрутизации

Каждая запись в таблице маршрутизации обеспечивает информацию о маршрутизации к отдельной точке назначения, которая может быть отдельным хостом, сетью, суперсетью или *значением по умолчанию.*

Существует общее правило использования в протоколе IP таблицы маршрутизации независимо от расположения этой таблицы — на хосте или маршрутизаторе. Выбираемый в таблице элемент должен *наиболее точно соответствовать* IP-адресу назначения. Другими словами, когда IP просматривает адреса хостов назначения, концептуально выполняются следующие действия:

- Сначала в таблице ищется адрес, полностью совпадающий с IP-адресом назначения. Если он будет найден, эта запись используется для маршрутизации трафика.
- Если такого адреса нет, в таблице ищется запись для подсети системы назначения.
- Если нет и такого адреса, в таблице проводится поиск сети назначения.
- Если отсутствует и этот адрес, в таблице проводится поиск элемента с соответствующим префиксом маршрутизации.
- Если не будет найден и этот адрес, используется маршрутизатор по умолчанию.

Разумеется, реальное выполнение предполагает однократный просмотр таблицы с отбрасыванием всех найденных, но менее точных совпадений.

6.9 Таблицы маршрутизатора

В отличие от таблиц маршрутизации хостов, которые могут быть очень простыми, таблицы маршрутизаторов часто содержат намного больше информации. Маршрутизатор имеет два или более интерфейсов, и каждая датаграмма должна быть передана через соответствующий ей интерфейс. Маршрутизатору могут потребоваться записи о следующих попаданиях для множества различных сетей и подсетей (см. рис. 6.6).

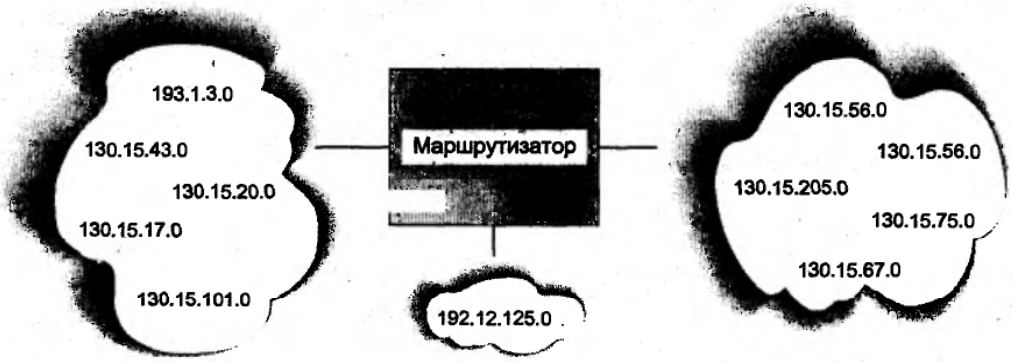

Рис. 6.6. Маршрутизация по многим направлениям



Рис. 6.7. Маршрутизация в филиале компании

6.10 Таблица маршрутизации филиала компании

Некоторые маршрутизаторы имеют очень простые таблицы маршрутизации. Например, маршрутизатор филиала компании (см. рис. 6.7) направляет трафик из главного офиса в локальные сети и перенаправляет весь выходящий трафик по региональной сети в главный офис компании.

Этот маршрутизатор имеет два интерфейса:

Интерфейс	IP-адрес
1	130.15.40.1
2	130.15.201.2

Таблица маршрутизации будет содержать:

Назначение	Интерфейс	Следующее попадание	Тип	Протокол
130.15.40.0	1	130.15.40.1	Прямая	Вручную
0.0.0.0	2	130.15.201.1	Косвенная	Вручную

Первая запись описывает только прямое соединение с локально подключенной подсетью 130.15.40.0. Подсеть достигается *непосредственно* через собственный интерфейс.

Вторая запись указывает маршрут по умолчанию к остальной части сети. Маршрутизатор для следующего попадания — 130.15.201.1 — доступен через интерфейс 2. Главный офис компании достигается *косвенным* путем, через маршрутизатор следующего попадания. Оба маршрута были введены вручную.

6.11 Операции глобальной маршрутизации

Пока мы рассматривали только выбор единственного направления к точке назначения. Рисунок 6.8 поясняет действия при глобальной маршрутизации в IP. Если протоколы TCP или UDP хоста A захотят послать данные своему партнеру на хосте B, они передадут эти данные IP, сопроводив их IP-адресом хоста назначения. IP добавит заголовок, содержащий IP-адрес назначения для данных.

- IP хоста A исследует адрес назначения, чтобы проверить, не находится ли он в локальной подсети. Если нет, IP выполнит поиск в таблице маршрутизации.
- Из таблицы видно, что следующим попаданием является маршрутизатор X. Датаграмма будет заключена в кадр, а в его заголовок будет помещен физический адрес локальной сети для маршрутизатора X.
- Когда датаграмма прибудет на маршрутизатор X, удаляется ее обрамление кадром. IP маршрутизатора X сравнивает IP-адрес назначения со всеми своими адресами (по маске подсети) и проверяет, не находится ли точка назначения в локально подключенной подсети.

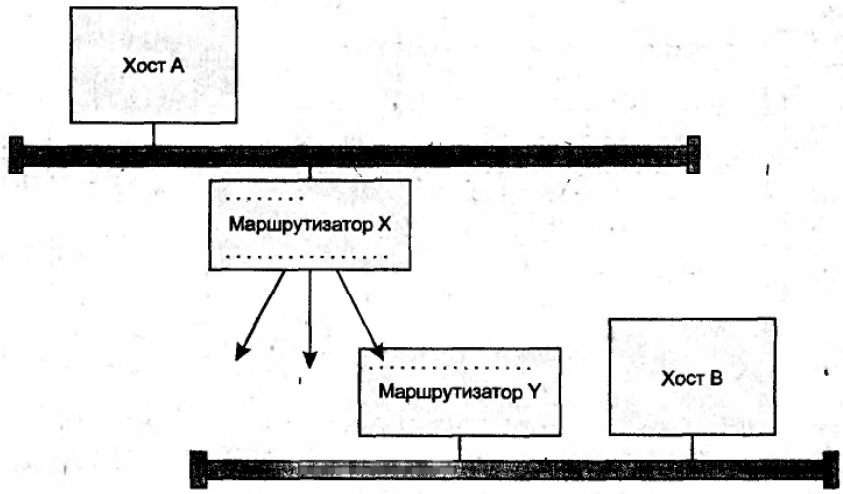

Рис. 6.8. Глобальная маршрутизация

- Если нет, IP выполнит поиск в таблице маршрутизации. Следующим попаданием станет маршрутизатор Y, куда и будет направлена датаграмма после обрамления ее новым кадром.
- Когда датаграмма поступит на маршрутизатор Y, будет удалено обрамление кадром. Протокол IP маршрутизатора Y сравнит IP-адрес назначения со всеми своими адресами (по маске подсети) и проверит, не находится ли точка назначения в локально подключенной подсети. Для нашего примера поиск будет успешным и датаграмма будет послана хосту В.

Маршрут от хоста А к хосту В содержал три попадания (участка): A-X, X-Y и Y-B.

6.12 Возможности IP

В IP существует несколько возможностей, обеспечивающих гибкость и пригодность этого протокола к различным окружениям. Среди прочих следует упомянуть *адаптивную маршрутизацию* (adaptive routing), а также *фрагментацию и сборку датаграммы* (datagram fragmentation and reassembly).

6.12.1 Адаптивная маршрутизация

Маршрутизация датаграмм *адаптивна* по своей природе. Лучший вариант для следующего попадания в любом из устройств выполняется при поиске в таблице маршрутизации текущего сетевого узла. Записи таблицы маршрутизации могут изменяться с течением времени, отражая текущее состояние сети.

Если одна из связей (см. рис. 6.9) будет разорвана, датаграмма может переключиться на другой маршрут (если он будет доступен).

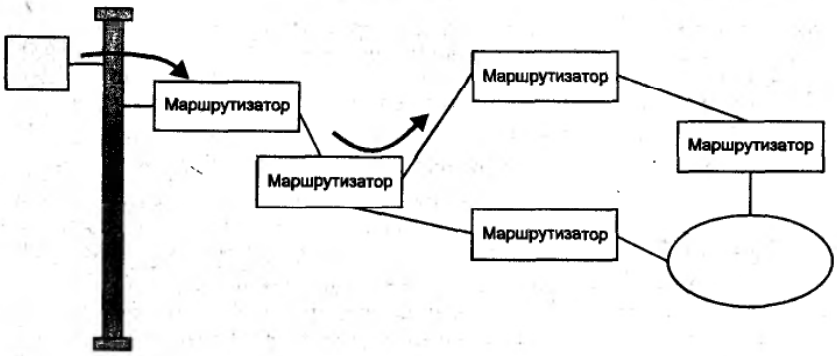

Рис. 6.9. Адаптивная маршрутизация

Изменение в топологии сети приводит к автоматическому перенаправлению датаграммы по другому маршруту. Адаптивная маршрутизация характеризуется гибкостью и надежностью.

С другой стороны, заголовок IP может содержать точный маршрут для перемещения к точке назначения. Это позволяет маршрутизировать важный трафик по засекреченному сетевому пути.

6.12.2 MTU, фрагментация и сборка

Перед тем как датаграмма отправится по сети к участку следующего попадания, она инкапсулируется внутри заголовка (заголовков) второго уровня, требующегося для данной сетевой технологии (см. рис. 6.10). Например, для прохождения сети 802.3 или 802.5 добавляются: заголовок LLC, подзаголовок SNAP, MAC-заголовок и завершающая часть MAC.

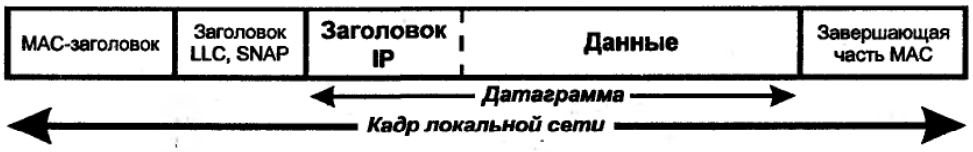

Рис. 6.10. Формат пересылки кадра локальной сети

Как было показано в главе 4, каждая технология локальной или глобальной сети имеет собственные ограничения на длину кадров. Датаграмма должна размещаться внутри кадра, и, следовательно, его максимальная длина будет ограничивать размер датаграммы, пересылаемой по носителю.

Максимальная длина датаграммы для конкретного носителя вычисляется как разность максимального размера кадра, длины заголовка кадра, длины завершающей части кадра и размера заголовка уровня связи данных:

Максимальный размер кадра – длина заголовка кадра – длина завершающей части кадра – размер заголовка уровня связи данных

Максимально возможная длина датаграммы в заданном носителе называется *максимальным элементом пересылки* (Maximum Transmission Unit — MTU). Например, для DIX Ethernet значение MTU равно 1500 октетам, для 802.3 — 1492 октетам, для FDDI — 4352, для SMDS — 9180 октетам.

В больших сетях интернета хост источника может не знать размеров всех ограничений по пути пересылки датаграммы. Что же произойдет, если хост отправит слишком большую для одной из промежуточных сетей датаграмму?

Когда такая датаграмма достигнет маршрутизатора, подключенного к промежуточной сети, IP решит проблему с размером датаграммы, разделив ее на несколько небольших *фрагментов*. *Хост назначения* далее должен будет провести сборку всех полученных кадров и восстановить исходную датаграмму.

Фрагментация наиболее часто выполняется в маршрутизаторах, однако приложения UDP могут разделить длинное сообщение на фрагменты датаграмм сразу в хосте источника.

6.13 Механизмы протокола IP

Рассмотрим более детально характеристики протокола IP версии 4, в том числе элементы формата этого протокола — формат заголовка IP и правила управления датаграммой, пересылаемой по сети. Протокол IP версии 6 рассмотрен в главе 22 (IP версии 5 не существует).

6.13.1 Заголовок датаграммы

Заголовок датаграммы организован как 5 или более 32-разрядных *слов*. Максимальная длина заголовка — 15 слов (т.е. 60 октетов), но на практике большинство заголовков датаграмм имеют минимально возможную длину в 5 слов (20 октетов).

Поля заголовка показаны на рис. 6.11. Они структурированы как последовательность слов. Отметим, что биты слов пронумерованы от 0 до 31.

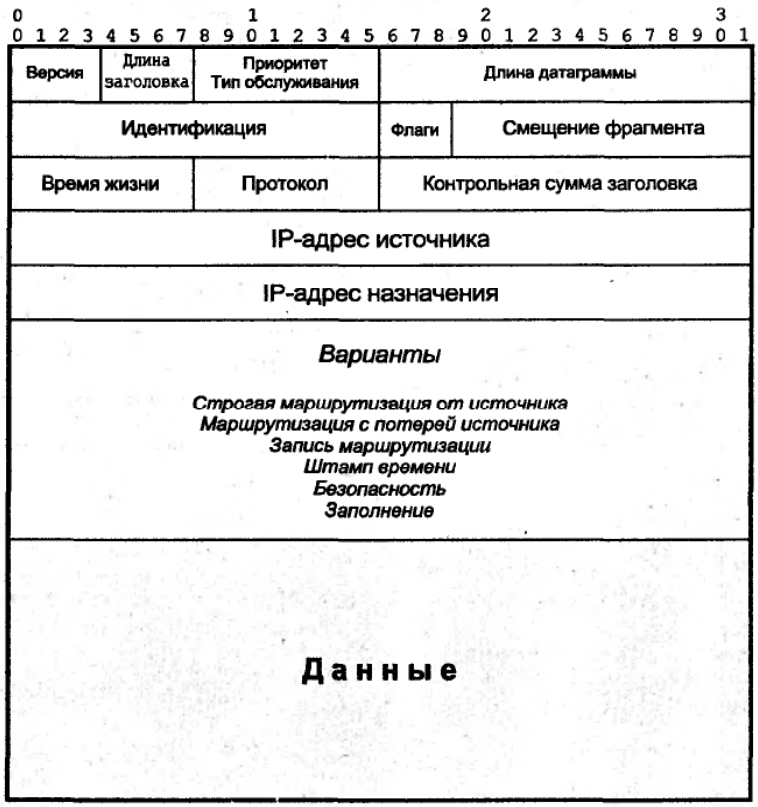

Рис. 6.11. Формат датаграммы протокола IP

6.13.2 Поля назначения, поле источника и поле протокола

Наиболее важными полями заголовка являются: *Destination IP Address* (IP-адрес назначения), *Source IP Address* (IP-адрес источника) и *Protocol* (протокол).

IP-адрес назначения позволяет маршрутизировать датаграмму. Как только она достигает точки назначения, поле *протокола* позволяет доставить ее в требуемую службу, подобную TCP или UDP. Кроме TCP и UDP, существует еще несколько протоколов, способных посылать и получать датаграммы. Организация IANA отвечает за координацию присваивания значений параметрам TCP/IP, включая значения в поле *протокола*. Некоторые значения из этого поля имеют лицензионный, специфичный для конкретного производителя смысл.

В таблице 6.1 показаны наиболее распространенные значения из поля *протокола*.

Таблица 6.1

Общепринятые номера из поля протокола заголовка IP

Номер	Название	Протокол	Описание
1	ICMP	Internet Control Message Protocol	Переносит сообщения об ошибках и поддерживает отдельные сетевые утилиты
2	IGMP	Internet Group Management Protocol	Обеспечивает группы для многоадресных рассылок
6	TCP	Transmission Control Protocol	Обслуживает сеансы
8	EGP	Exterior Gateway Protocol	Устаревший протокол для маршрутизации во внешней сети

Таблица 6.1 (*продолжение*)

Общепринятые номера из поля протокола заголовка IP

Номер	Название	Протокол	Описание
17	UDP	User Datagram Protocol	Обслуживает доставку независимых блоков данных
88	IGRP	Interior Gateway Routing Protocol	Обеспечивает взаимный обмен информацией о маршрутизации между маршрутизаторами компании Cisco

6.13.3 Версия, длина заголовка и длина датаграммы

В настоящее время используется четвертая версия IP (версия "Следующее поколение" имеет номер 6).

Длина заголовка измеряется в 32-разрядных словах. Если не нужны дополнительные варианты, можно ограничиться длиной заголовка в 5 слов (т.е. 20 октетов). Если задействованы один или больше дополнительных вариантов, может потребоваться заполнить конец заголовка незначащими нулями до границы 32-разрядного слова.

Поле *длины датаграммы* определяет размер датаграммы в октетах. В это значение включается как заголовок, так и часть данных датаграммы. В таком 16-разрядном поле можно указывать значения до $2^{16} - 1$ октет = 65 535 октетов.

Сетевые технологии — не единственная причина ограничений на размер датаграмм. Различные типы компьютеров, поддерживающих IP, имеют разные ограничения, связанные с размером буферов памяти, используемых для сетевого трафика (стандарт IP требует, чтобы все хосты были способны принимать датаграммы не менее чем из 576 октетов).

6.13.4 Приоритет и тип обслуживания

Первоначальным спонсором набора протоколов TCP/IP было Министерство обороны США, для которого было важно задание приоритетов датаграмм. Приоритеты мало используются вне военных и правительственных организаций. Для приоритета предназначены 3 бита, обеспечивающие 8 различных уровней.

Стандарт IP не регламентирует действия с битами приоритета. Первоначально они предназначались для установки параметров подсетей, которые будет пересекать датаграмма при следующем попадании. Например, на основе битов приоритета управляется протокол Token-Ring. В этом случае IP должен отображать биты приоритета в соответствующие уровни Token-Ring.

Тип обслуживания (Type of Service — TOS) содержит биты, определяющие качество обслуживания информации, которое может повлиять на обработку датаграмм. Например, когда маршрутизатору не хватает памяти, он вынужден отклонять некоторые датаграммы. Он мог бы рассматривать только датаграммы, у которых бит надежности установлен в единицу, и отбрасывать датаграммы с нулевым битом надежности.

Положение приоритета и типа обслуживания:

Биты	Тип	Описание
0-2	Приоритет	Уровни 0-7 Уровень 0 — обычный приоритет Уровень 7 — самый высокий приоритет
3-6	TOS	Задержка, надежность, пропускная способность, стоимость или безопасность
7	Зарезервировано для будущего использования	

Тип обслуживания определяет (как описано в текущем документе *Assigned Numbers*) значения, приведенные в таблице 6.2. Это взаимоисключающие значения — для любой IP-датаграммы требуется только одно значение TOS. Стандарт *Assigned Numbers* рекомендует использовать специальные значения для каждого из приложений. Например для *telnet* — минимизировать задержку, для копирования файлов — максимизировать производительность и надежность при доставке управляющих сетевых сообщений.

Таблица 6.2

Значения поля типа обслуживания (TOS)

Значение TOS	Описание
0000	По умолчанию
0001	Минимизировать денежную стоимость
0010	Максимизировать надежность
0100	Максимизировать производительность
1000	Минимизировать задержку
1111	Максимизировать безопасность

Некоторые маршрутизаторы полностью игнорируют поле типа обслуживания, в то время как другие могут использовать его при выборе трафика, который следует предохранить на случай недостатка оперативной памяти. Можно надеяться, что в будущем поле типа обслуживания будет играть гораздо большую роль. Рекомендуемые в документе *Assigned Numbers* значения представлены в таблице 6.3.

Таблица 6.3

Рекомендуемые значения поля типа обслуживания

Протокол	Значение TOS	Описание
Telnet и другие протоколы для регистрации	1000	Минимизировать задержку
Управляющий сеанс FTP	1000	Минимизировать задержку
Сеанс FTP по пересылке данных	0100	Максимизировать производительность
TFTP	1000	Минимизировать задержку
Фаза команд SMTP	1000	Минимизировать задержку
Фаза данных SMTP	0100	Максимизировать производительность
Запрос DNS к UDP	1000	Минимизировать задержку
Запрос DNS к TCP	0000	Без специального управления
Преобразование зон в DNS	0100	Максимизировать производительность
NNTP	0001	Минимизировать денежную стоимость
Ошибки ICMP	0000	Без специального управления
Запросы ICMP	0000	Обычно 0000, но иногда посылаются с другим значением
Ответы ICMP		То же, что и у запроса, для которого формируется ответ
Любые IGP	0010	Максимизировать надежность
EGP	0000	Без специального управления
SNMP	0010	Максимизировать надежность
BOOTP	0000	Без специального управления

6.13.5 Поле времени жизни

Когда в интернет-системе IP происходит изменение топологии, например обрыв связи или инициализация нового маршрутизатора, некоторые датаграммы могут сбиться со своего маршрута за тот короткий период времени, пока не будет выбран новый маршрутизатор.

Более серьезные проблемы возникают из-за ошибок при ручном вводе информации о маршрутизации. Такие ошибки могут привести к потере датаграммы или зацикливанию ее по круговому маршруту на длительное время.

Поле времени жизни (Time-To-Live — TTL) ограничивает время присутствия датаграммы в интернете. TTL устанавливается хостом-отправителем и уменьшается каждым маршрутизатором, через который проходит датаграмма. Если датаграмма не достигает пункта назначения, а ее поле TTL становиться нулевым, она отбрасывается.

Хотя формально время жизни оценивается в секундах, реально TTL реализуется как простой счетчик попаданий, значение которого уменьшается (обычно на единицу) в каждом маршрутизаторе. Можно указывать большее уменьшение счетчика для датаграмм, которые перемещаются по очень медленным соединениям или требуют длительного времени для пересылки.

Рекомендуемое значение по умолчанию для TTL — примерно в 2 раза больше, чем максимально возможный путь в сети. Длина такого максимального пути часто называется *диаметром* (diameter) интернета.

6.13.6 Заголовок контрольной суммы

Контрольная сумма (checksum) находится в 16-разрядном поле и вычисляется по значению остальных полей заголовка IP как сумма всех дополнений до единицы 16-разрядных слов заголовка. До вычисления поле контрольной суммы содержит 0. Контрольная сумма должна пересчитываться при перемещении датаграммы по сети, поскольку в датаграмме изменяется поле TTL. Могут изменяться и другие значения из заголовка вследствие фрагментации или записи информации в дополнительные поля.

6.14 Фрагментация

Поля *идентификации* (Identification), *флагов* (Flags) и *смещения фрагмента* (Fragment Offset) позволяют фрагментировать и восстанавливать (собирать) датаграмму. Когда IP нужно переслать датаграмму большего размера, чем MTU следующего участка, то:

1. Сначала проверяется содержимое поля *флагов*. Если значение "Не фрагментировать" установлено в 1, ничего делать не нужно — датаграмма отбрасывается и перестает существовать.
2. Если флаг "Не фрагментировать" установлен в 0, то поле данных разделяется на отдельные части в соответствии с MTU следующего участка. Полученные части выравниваются по 8-октетной границе.
3. Каждой части присваивается заголовок IP, подобный заголовку исходной датаграммы, в частности копируются значения полей источника, назначения, протокола и *идентификации*. Однако следующие поля устанавливаются индивидуально для каждой из частей:
   a. Длина датаграммы будет отражать текущую длину полученной датаграммы.
   b. Флаг More из поля *флагов* устанавливается в 1 для всех частей, кроме последней.
   c. Поле *смещения фрагмента* будет указывать позицию полученной части относительно начала исходной датаграммы. Начальная позиция принимается за 0. Смещение фрагмента равно реальному смещению, разделенному на 8.
   d. Для каждого фрагмента вычисляется собственная контрольная сумма.

Теперь настало время более подробно рассмотреть поля при фрагментации датаграммы.

6.14.1 Поле идентификации

Поле *идентификации* содержит 16-разрядное число, помогающее хосту назначения распознать фрагмент датаграммы при сборке.

6.14.2 Поле флагов

Поле *флагов* содержит три бита:

Бит 0	Бит 1	Бит 2
0 = Зарезервировано	0 = Можно фрагментировать 1 = Нельзя фрагментировать	0 = Последний фрагмент (Last) 1 = Есть еще фрагменты (More)

Бит 0 зарезервирован, но должен иметь значение 0. Отправитель может указать в следующем бите значение 1, и датаграмму нельзя будет фрагментировать. Если ее нельзя будет доставить без фрагментации, а бит фрагментации равен 1, то датаграмма будет отброшена с посылкой сообщения отправителю.

Бит 2 устанавливается в 0 для последней или единственной части датаграммы. Бит 2, установленный в 1, указывает, что датаграмма фрагментирована и имеет следующие далее части.

6.14.3 Поле смещения фрагмента

Блок фрагментации (fragment block) — это 8-октетная порция данных. Число в поле *смещения фрагмента* (Fragment Offset) указывает величину смещения данного фрагмента (относительно начала датаграммы) в единицах блоков фрагментирования. Это поле имеет длину 13 бит (т.е. смещение может быть от 0 до 8192 блоков фрагментирования — или от 0 до 65 528 октетов). Предположим, что маршрутизатор разделил датаграмму (с идентификатором 348) из 3000 байт данных на три датаграммы по 1000 байт. Каждый фрагмент будет содержать собственный заголовок и 1000 байт данных (125 блоков фрагментирования). Содержимое полей *идентификации*, *флагов* и *смещений фрагментов* будет следующим:

Фрагмент	Идентификатор	Флаги	Смещение фрагмента
1	348	Можно фрагментировать, More	0 блоков от начала
2	348	Можно фрагментировать, More	125 блоков (1000 октетов) от начала
3	348	Можно фрагментировать, Last	250 блоков (2000 октетов) от начала

Когда датаграмма доставляется без фрагментации, значения полей будут следующими:

Идентификатор	Флаги	Смещение фрагмента
348	Можно фрагментировать, Last	0 блоков от начала

Хост получателя, приняв датаграмму, помеченную как "Last" и имеющую смещение 0, знает, что она не фрагментирована.

6.14.4 Сборка фрагментированной датаграммы

Сборка фрагментированной датаграммы выполняется хостом-получателем. Отдельные части такой датаграммы могут прибывать в произвольном порядке. Когда в точке назначения появляется первый фрагмент, IP выделяет определенную область памяти для последующей сборки всей датаграммы. Поле *смещения фрагмента* указывает на байтовую границу для данных полученного фрагмента.

Совпадающие по полям *идентификации*, *IP-адреса источника*, *IP-адреса назначения* и *протокола* фрагменты составляются вместе по мере их поступления. Однако в протоколе IP имеется небольшой недостаток: хост получателя не может узнать общей длины датаграммы, пока не получит последний фрагмент. Поле *общей длины* (Total Length) содержит сведения только о *данном* фрагменте, а не об общей длине датаграммы.

Таким образом, система-получатель должна иметь возможность предвидеть, сколько именно буферного пространства нужно зарезервировать для принимаемой датаграммы. Разработчики решают эту проблему различными способами. Некоторые последовательно выделяют для буфера небольшие части памяти, другие сразу предоставляют единый большой буфер.

В любом случае при реализации *необходимо* обслуживать поступающую датаграмму с общей длиной, как минимум, в 576 октетов. Или, что более точно, система должна быть способна обрабатывать датаграммы с общим размером не менее чем MTU интерфейса, по которому поступают датаграммы.

6.14.5 Тайм-аут сборки датаграммы

Рассмотрим следующую последовательность событий:

- Пересылается датаграмма.
- Пославший ее процесс аварийно завершается.
- Датаграмма фрагментируется при пересылке.
- По пути следования теряется один из фрагментов.

При потере отправленного фрагмента хост получателя должен ждать, пока этот фрагмент не будет отправлен повторно. При этом, разумеется, необходимо *ограничить время ожидания*. Когда тайм-аут сборки завершится, хост назначения отбросит уже полученные фрагменты и отправит источнику сообщение об ошибке. Обычно величину тайм-аута сборки можно конфигурировать. Ее значение рекомендуется устанавливать в диапазоне от 60 до 120 с.

6.14.6 Фрагментировать или не фрагментировать

Учитывая все проблемы поддержки фрагментации, можно сказать, что она приводит к снижению производительности. Поэтому многие программисты стремятся аккуратно разрабатывать приложения, чтобы формируемые датаграммы были достаточно малы и не фрагментировались при пересылке.

В главе 7 мы познакомимся с протоколом исследования MTU, позволяющим исключить фрагментирование датаграмм и использовать наибольший размер MTU при пересылке информации.

6.15 Просмотр статистики IP

Узнать о том, как работает IP, можно по достаточно приблизительным статистическим отчетам. Команда *netstat -s* выводит содержимое счетчиков для наиболее важных событий в IP. Нижеприведенный отчет получен на сервере *tigger.jvnc.net*, который доступен хостам всей сети Интернет. Отметим, что в отчете вместо более точного термина "*датаграмма*" используется термин "*пакет*" (packet).

```
> netstat -s
ip:
13572051 total packets received
0 bad header checksums
0 with size smaller than minimum
8 with data size < data length
0 with header length < data size
0 with data length < header length
90 fragments received
0 fragments dropped (dup or out of space)
2 fragments dropped after timeout
0 packets forwarded
10 packets not forwardable
0 redirects sent
0 ip input queue drops
```

За отчетный период не было ни одной датаграммы с плохой контрольной суммой (checksums) и *tigger* не отбросил ни одной датаграммы из-за недостатка памяти. Было принято 90 фрагментов, что составляет 0,00066% от общего объема информации. Два фрагмента отброшены по тайм-ауту, а 10 непересылаемых (nonforwardable) датаграмм, возможно, возникли при попытке маршрутизации от источника через *tigger*.

6.16 Варианты

Для одного или нескольких дополнительных вариантов доступно 40 специальных октетов в заголовке IP. Варианты датаграмм выбираются отсылающими их приложениями. Применяются они крайне редко. Список вариантов включает:

- Strict Source Route (Точный маршрут от источника)
- Loose Source Route (Произвольный маршрут от источника)

- Record Route (Запись маршрута)
- Timestamp (Временная метка)
- Department of Defense Basic Security (Базовая безопасность Министерства обороны)
- Department of Defense Extended Security (Улучшенная безопасность Министерства обороны)
- No Operation (Без операций)
- End of Option List (Padding) — Конец списка вариантов (заполнитель)

Варианты безопасности используются Министерством обороны и некоторыми правительственными агентствами. Предложено также несколько других вариантов (полный список вариантов и их текущий статус можно найти в последних изданиях *Assigned Number* и *Internet Official Protocol Standard*).

6.16.1 Маршрутизация от источника

Существуют два источника: *Strict Source Route*, определяющий полный путь к точке назначения, и *Loose Source Route*, идентифицирующий контрольные точки по пути следования (milestones). Между контрольными точками можно использовать любые маршруты.

Strict Source Routes иногда применяется для повышения безопасности данных. Однако, как мы увидим далее, этот источник используется и хакерами при взломе систем компьютерной безопасности.

Иногда этот вариант используется при тестировании сетей. Loose Source Route предназначен для помощи при маршрутизации в удаленную точку.

Механизмы обоих вариантов похожи. Единственным отличием является то, что в Strict Source Route можно посещать *только* системы из заранее заданного списка.

6.16.2 Обратный маршрут

Если используется маршрутизация от источника, обратный трафик от точки назначения к источнику должен повторять тот же путь (набор маршрутизаторов), но в обратном порядке.

При этом возникает одна сложность: с точки зрения источника и системы назначения адреса маршрутизаторов различны. На рис. 6.12 показан путь между двумя хостами. Маршрут от хоста A к хосту B предполагает прохождение через маршрутизаторы, адресами которых для хоста A являются 130.132.9.29 и 130.132.4.11. Путь от хоста B к хосту A проходит через маршрутизаторы с IP-адресами, известными хосту B как 128.36.5.2 и 130.132.4.16. Адреса интерфейсов маршрутизатора различны, поскольку они подключены к разным подсетям.

Решить эту проблему просто: при каждом посещении маршрутизатора входной адрес заменяется в поле *Source Route* на выходной адрес, а система назначения получает уже результирующий список в обратном порядке и может использовать маршрутизацию от источника для обратного перемещения.

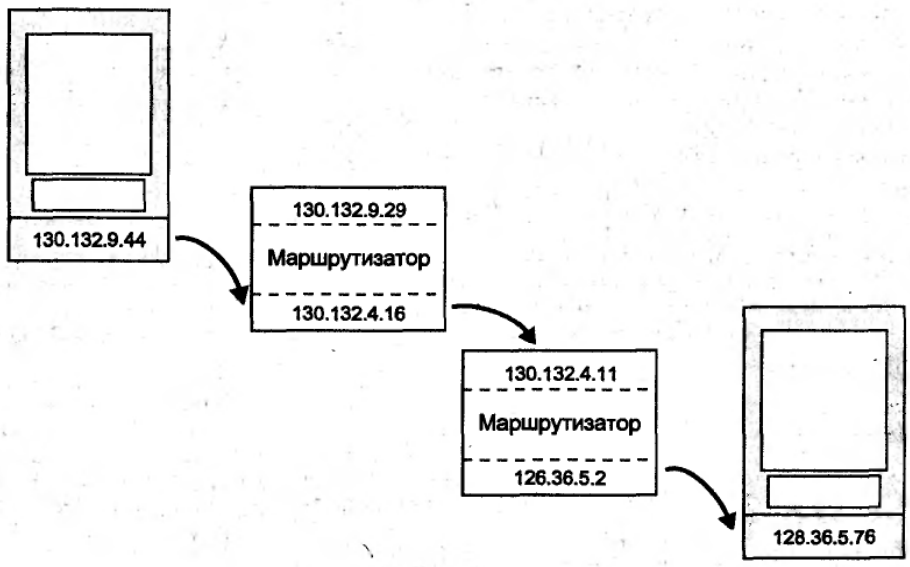

Рис. 6.12. Пути с точки зрения хостов A и B

6.16.3 Описание маршрута

Можно подумать, что для маршрутизации от источника достаточно создать список маршрутизаторов *между* источником и точкой назначения. Однако это не так. В таблице 6.4 представлено содержимое полей *IP-адреса источника* (Source IP Address), *IP-адреса места назначения* (Destination IP Address) и поля *маршрутизации от источника* (Source Route) на каждом шаге по пути перемещения:

- На шаге 1 поле *IP-адреса назначения* содержит адрес первого маршрутизатора. Указатель из поля *Source Route* определяет следующее попадание (в таблице — полужирным шрифтом).
- На шаге 2 поле *IP-адреса назначения* содержит адрес второго маршрутизатора. Указатель из поля *Source Route* определяет следующее попадание. В нашем примере — это реальная точка назначения датаграммы.
- На шаге 3 датаграмма достигает назначения. Ее поля *IP-адреса источника* и *назначения* содержат правильные значения, а в *Source Route* перечислены все пройденные маршрутизаторы.

Таблица 6.4

Маршрутизация от источника

Шаг	IP-адрес источника	IP-адрес назначения	Поле Source Route
1	130.132.9.44	130.132.9.29	**130.132.4.11** 128.36.5.76
2	130.132.9.44	130.132.4.11	130.132.4.16 **128.36.5.76**
3	130.132.9.44	128.36.5.76	130.132.4.16 128.36.5.2

6.16.4 Маршрутизация от источника и безопасность

Маршрутизация от источника стала у сетевых хакеров частью арсенала инструментов для взлома. Этот метод может быть использован для проникновения из Интернета в сети, администраторы которых не беспокоятся о безопасности.

Маршрутизаторы, фильтрующие поступающий в организацию трафик, должны позволять блокировать трафик с маршрутизацией от источника или проверять поле *Source Route* на соответствие *реальной* точке назначения датаграммы.

Еще одна проблема возникает с многоадресными хостами, подключенными к одной или нескольким подсетям. Дело в том, что такие хосты могут пропускать датаграммы с маршрутизацией от источника, открывая доступ к трафику сети с "черного хода". Многоадресные хосты также должны уметь запрещать маршрутизацию от источника.

6.16.5 Запись пути

Поле *записи пути* (Record Route) содержит список IP-адресов маршрутизаторов, пройденных датаграммой. Каждый встретившийся по пути следования маршрутизатор пытается добавить свой выходной адрес в такой список.

Но длина списка задается отправителем, и, возможно, что для записи всех адресов по пути следования датаграммы может не хватить места. В этом случае маршрутизатор будет просто пересылать датаграмму без добавления своего адреса.

6.16.6 Временная метка

Существуют три формата для поля *временной метки* (Timestamp), которое может содержать:

- Список 32-разрядных временных меток
- Список IP-адресов и соответствующих им пар временных меток.
- Список предварительно выбранных в источнике адресов со следующим за ним пространством для записи временной метки (сетевые узлы будут записывать туда временные метки, только когда их адреса будут совпадать с адресами из списка)

В первом и втором случаях может не хватить места для записи. Тогда создается подполе переполнения (overflow) для подсчета узлов, которые не смогли записать свои временные метки.

6.16.7 Базовая и улучшенная безопасность для Министерства обороны

Вариант *базовой безопасности* (Basic Security) используется для проверки того, что источник имеет право на отправку датаграммы, маршрутизатор — на трансляцию, а приемник — на ее получение.

Параметр *Basic Security* состоит из классификационных уровней, изменяющихся от Unclassified (не секретно) до Top Secret (совершенно секретно) и флагов идентификации авторства. Эти уровни определяют правила для датаграммы. Авторство присвоено нескольким организациям, например Агентству национальной безопасности США, ЦРУ и Министерству энергетики.

Датаграмма с Basic Security может содержать и поле *Extended Security*. Для этих полей существует несколько различных подформатов, определяющих требования различных владельцев авторства.

Маршрутизатор или хост должен уничтожить информацию, на которую у него нет права авторства. Системы безопасности конфигурируются с различными классификационными уровнями и могут посылать и получать сведения об авторстве (авторствах), если это разрешено. Отметим, что многие коммерческие продукты не поддерживают таких возможностей.

6.16.8 Конец списка вариантов и отсутствие операций

Вариант "*без операций*" (No Operation) применяется для заполнения промежутков между вариантами датаграмм. Например, он используется для выравнивания следующего варианта по 16- или 32-разрядной границе.

Конец списка вариантов (End of Option List) служит для заполнения полей вариантов до 32-разрядной границы.

6.16.9 Кодирование вариантов

Существуют два однобайтовых варианта, кодируемых следующим образом:

No Operation	00000001
End of Option List	00000000

Оставшиеся варианты задаются несколькими битами. Каждый начинается октетом *типа* и октетом *длины*.

Для рассматриваемых вариантов возникает следующий вопрос: нужно ли их копировать в заголовки получаемых при фрагментации датаграмм? Копирование выполняется для *Security*, *Strict Source Route* и *Loose Source Route*. Поля *Record Route* и *Timestamp* копируются только в первый фрагмент датаграммы.

Октет типа подразделяется на:

Биты	Функция	Описание
0	Флаг копирования	Устанавливается в 1 для копирования при фрагментации
1-2	Класс варианта	0 — для датаграмм или сетевых управляющих сообщений 2 — для отладки и измерений
3-7	Номер варианта	Уникальное значение для каждого из вариантов

В таблице 6.5 показаны значения октета типа и его деление на поля Copy (копирование), Class (класс) и Option Number (номер варианта) для каждого стандартного варианта.

Таблица 6.5

Поля Copy, Class и Option Number

Значение	Copy	Class	Number	Имя
0	0	0	0	End of Options List
1	0	0	1	No Operation
137	1	0	9	Strict Source Route
131	1	0	3	Loose Source Route
7	0	0	7	Record Route
68	0	2	4	Timestamp
130	1	0	2	Security
133	1	0	5	Extended Security

Strict Source Route

137	Длина	Указатель	Список адресов

Loose Source Route

131	Длина	Указатель	Список адресов

Record Route

7	Длина	Указатель	Список адресов

Временная метка: Флаг = 0

68	Длина	Указатель	Переполнение	Флаг 0
Временная метка				
Временная метка				
...				

Временная метка: Флаг = 1 или 3

68	Длина	Указатель	Переполнение	Флаг 1, 3
IP-адрес				
Временная метка				
IP-адрес				
Временная метка				
...				

Рис. 6.13. Форматы полей вариантов

Форматы наиболее общих полей вариантов представлены на рис. 6.13.

6.16.10 Кодирование Strict Source Route

Вариант *Strict Source Route* (точный маршрут от источника) содержит указатели на список адресов. Указатель определяет позицию следующего обрабатываемого адреса. Первоначально указатель имеет значение 4, которое увеличивается на 4 при каждом попадании.

6.16.11 Кодирование Loose Source Route

Вариант *Loose Source Route* (произвольный маршрут от источника) содержит указатели на список адресов. Исходное положение указателя, как в предыдущем случае, здесь 4 и увеличивается на 4 при достижении каждого из адресов списка.

6.16.12 Кодирование Record Route

Вариант *Record Route* (запись маршрута) содержит указатели и место для записи адресов. Первоначально указатель имеет значение 4, а место, предназначенное для записи адреса, пусто.

При достижении каждого маршрутизатора его адрес записывается по указателю, а значение указателя увеличивается на 4. Когда будет занято все выделенное для записи место, датаграмма продолжит путь к точке назначения без записи дополнительных адресов.

6.16.13 Кодирование Timestamp

Вариант *Timestamp* (временная метка) содержит указатель, подполе переполнения и подполе флага. Подполе флага определяет один из трех возможных для временной метки форматов.

Если в подполе флага содержится 0, то при каждом попадании в выделенном месте записывается временная метка, а значение указателя увеличивается на 4. Когда будет заполнено все предварительно выделенное пространство, значение подполя переполнения увеличивается на единицу и все поступающие датаграммы отбрасываются.

Если подполе флага хранит 1, то при каждом попадании по IP-адресу на пустое место будет записываться временная метка, а значение указателя увеличиваться на 8. Когда будет заполнено все предварительно выделенное пространство, значение подполя переполнения увеличивается на единицу и запись меток прекращается. Предположим, что отправитель хочет записать временные метки для списка предварительно выбранных узлов. В этом случае в поле флага нужно указать 3 и заполнить список выбранных адресов интернета. Если в текущий момент указатель установлен на адрес маршрутизатора, это устройство заполнит место для временной метки и увеличит значение указателя на 8.

6.16.14 Кодирование Basic и Extended Security Options

Значения этих полей устанавливаются военными и правительственными агентствами. Дополнительные сведения можно получить в RFC 1108.

6.17 Пример заголовка IP

На рис. 6.14 показан результат работы анализатора *Sniffer* компании Network General для заголовка MAC-кадра сети DIX Ethernet и для заголовка IP.

```
DLC:   — DLC Header —
DLC:
DLC:   Frame 14 arrived at 10:26:10.5797; size is 61 bytes
DLC:   Destination   = Station Sun      076A03,  Sun Atlantis
DLC:   Source        = Station Sun      07FD89,  Sun Jupiter
DLC:   Ethertype     = 0800 (IP)
DLC:

IP:    — IP Header —
IP:
IP:    Version = 4, header length = 20 bytes
IP:    Type of Service = 00
IP:         000. .... = routine
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:    Total length = 47 bytes
IP:    Identification = 4458
IP:    Flags = 0X
IP:    .0.. .... = may fragment
IP:    ..0. .... = last fragment
IP:    Fragment offset = 0 bytes
IP:    Time to Live = 30 seconds/hops
IP:    Protocol = 6 (TCP)
IP:    Header checksum = 12F4 (correct)
IP:    Source address = [192.42.252.1]
IP:    Destination address = [192.42. 252.20]
IP:    No options
IP:

HEX

MAC Header
08 00 20 07 6A 03    (Destination physical address)
08 00 20 07 FD 89    (Source physical address)
08 00                (Protocol Type for IP)

IP Header
45 00 00 2F (Version, Hdr Length, Prec/TOS, Total Length)
11 6A 00 00 (Identification, Flags, Fragment Offset)
1E 06 12 F4 (Time to Live, Protocol, Header Checksum)
C0 2A FC 01 (Source IP Address)
C0 2A FC 14 (Destination IP Address)
```

Рис. 6.14. Интерпретация заголовков MAC и IP

Заголовок MAC начинается 6-байтовыми физическими адресами систем источника и назначения. Отметим, что анализатор *Sniffer* заменяет первые 3 байта каждого физического адреса на соответствующее имя компании — производителя сетевого адаптера (в нашем случае это *Sun*). Поле типа содержит код X'0800, что означает: "данную информацию доставлять в IP".

На рисунке датаграмма IP следует сразу за коротким MAC-заголовком сети DIX Ethernet. Это кадр стандарта 802.3, а за MAC-заголовком располагаются 8-байтовый заголовок LLC с подзаголовком SNAP.

Размер кадра — 61 байт. В эту величину включается 14-байтовый MAC-заголовок кадра, но не учитывается 4-байтовая завершающая часть MAC, поэтому полный кадр имеет длину 65 байт. Кадры Ethernet или 802.3 для носителя на коаксиальном кабеле должны иметь длину не менее 64 байт, следовательно, кадр едва не стал меньше допустимого минимального размера. Датаграмма кадра имеет общую длину только 47 байт.

Как и многие заголовки IP, рассматриваемый в примере заголовок не содержит вариантов и, следовательно, имеет длину 20 байт. На практике поле *Type of Service* имеет, как правило, значение 0.

Можно заметить, что датаграмма не является фрагментом более длинной датаграммы, поскольку поле *Fragment Offset* хранит 0, показывая начало датаграммы, и второй флаг установлен в 0, указывая на ее конец.

Датаграмма зафиксировала информацию о 30 попаданиях в поле *TTL*. Поле *Protocol* имеет значение 6, что указывает на доставку датаграммы TCP на хост назначения.

Анализатор *Sniffer* транслировал IP-адреса источника и назначения в общепринятый формат с точками.

Шестнадцатеричные октеты, создающие исходный заголовок MAC и заголовок IP, показаны в нижней части рисунка. Заданный в *Sniffer* вывод в шестнадцатеричном формате был заменен на соответствующие, но более простые значения в формате с точками.

6.18 Сценарий обработки датаграммы

Для лучшего понимания работы IP рассмотрим операции по обработке датаграммы в маршрутизаторе и хосте назначения. Выполняемые при этом шаги показаны на рис. 6.15.

Маршрутизатор	Хост
Вычисление контрольной суммы заголовка.	Вычисление контрольной суммы заголовка.
Проверка параметров. Отбрасывание датаграммы при неверных параметрах.	Проверка параметров. Отбрасывание датаграммы при неверных параметрах.
Уменьшение времени жизни. Отбрасывание датаграммы при нулевом результате.	Проверка назначения. Если сюда, то сохранить датаграмму.
Проверка безопасности просмотра. Отбрасывание датаграммы при любой неуспешной проверке.	Маршрутизация от источника? Проверка на допустимость пересылки.
Выбор следующего попадания. Отбрасывание датаграммы при необходимости фрагментации и флаге "Не фрагментировать" в значении 1.	Не фрагментирована? Послать службе более высокого уровня.
Обработать варианты, если они есть. Изменить поля для вариантов (например, поле маршрутизации от источника).	Фрагментирована? Вставить фрагмент. Когда сборка датаграммы будет завершена – послать службе более высокого уровня.
Изменить заголовок датаграммы (или заголовки ее фрагментов). Вычислить новые контрольные суммы для заголовка (заголовков).	Завершил работу таймер фрагментации? Отбросить фрагменты.
Послать на систему следующего попадания.	

Рис. 6.15. Обработка датаграммы

Возникающие проблемы и ошибки приводят обычно к отбрасыванию датаграммы и посылке источнику сообщения об ошибке. Эти процессы будут рассмотрены в главе 7, посвященной протоколу *Internet Control Message Protocol* (ICMP).

6.18.1 Обработка в маршрутизаторе

После получения датаграммы маршрутизатор проводит серию проверок, чтобы узнать, не нужно ли отбросить данную датаграмму. Вычисляется контрольная сумма заголовка и сравнивается со значением из поля контрольной суммы.

Просматриваются поля *версии*, *длины заголовка*, *общей длины* и *протокола* для выявления имеющих смысл значений. Уменьшается значение из поля времени жизни. При ошибке в контрольной сумме, параметрах или нулевом значении времени жизни, а также в том случае, когда маршрутизатор не имеет достаточного объема памяти для продолжения ее обработки, датаграмма отбрасывается.

На следующем шаге выполняется анализ безопасности посредством серии предварительно сконфигурированных тестов. Например, маршрутизатор может ограничить входной трафик, чтобы было доступно только несколько серверов назначения.

Далее маршрутизатор выполняет процедуру маршрутизации датаграммы. По указанию в заголовке датаграммы выбирается вариант точного или произвольного маршрута от источника. Затем, если это необходимо и разрешено, осуществляется фрагментирование. Если датаграмма не может быть передана дальше без фрагментации, но поле "Не фрагментировать" имеет значение 1, она отбрасывается.

Имеющиеся варианты обрабатываются. Измененные заголовки должны быть построены для каждой датаграммы или ее фрагмента. Наконец повторно вычисляется контрольная сумма заголовка и датаграмма пересылается системе следующего попадания. Это наиболее общий сценарий обработки датаграммы маршрутизатором. Однако иногда он является конечной точкой назначения датаграммы. Например, запрос сетевой управляющей информации может посылаться на сам маршрутизатор.

6.18.2 Обработка в хосте назначения

В хосте назначения вычисляется *контрольная сумма*, и полученный результат сравнивается с соответствующим полем. Проверяется адрес назначения на принадлежность данному хосту. Проверяется также корректность полей *версии*, *длины заголовка*, *общей длины* и *протокола*. Датаграмма отбрасывается при любой ошибке или при недостатке буферного пространства для ее обработки хостом.

Если датаграмма фрагментирована, то хост проверяет четыре поля: *идентификации*, *адреса источника*, *адреса назначения* и *протокола* для выявления фрагментов с идентичными значениями (т.е. принадлежащих одной датаграмме). Далее используется значение из поля смещения фрагмента для позиционирования фрагментов относительно друг друга.

Целая датаграмма пересылается соответствующей службе высокого уровня, например TCP или UDP.

Хост не ожидает полного завершения сборки датаграммы из фрагментов. Когда поступает первый фрагмент, таймер устанавливается в локально конфигурируемое значение (обычно между 1 и 2 минутами). Фрагменты датаграммы, не собранные за это время, отбрасываются.

6.19 Средства защиты и безопасность

Все хотят получить максимальные преимущества от коммуникаций, но благоразумный сетевой администратор всегда принимает меры, чтобы защитить ресурсы компьютеров от воздействия извне, в первую очередь от хакеров. *Маршрутизаторы со средствами защиты* (firewall router; иногда используется дословный перевод — брандмауер, т.е. противопожарная стенка.— *Прим. пер.*) стали наиболее популярными устройствами в оборонительном арсенале сетевого администратора.

Маршрутизаторы со средствами защиты устанавливаются для фильтрации трафика с целью обеспечения безопасности сайта. Как показано на рис. 6.16, такие маршрутизаторы могут конфигурироваться на разрешение или запрещение трафика на основе:

- IP-адреса источника
- IP-адреса назначения
- Протокола
- Приложения

Например, внутренним пользователям можно разрешить посылку и прием сообщений электронной почты и доступ к внешним серверам WWW, а внешним пользователям — только к небольшому подмножеству серверов сайта.



Рис. 6.16. Маршрутизатор со средствами защиты

Дополнительная защита обеспечивается интеллектуальным фильтрующим хостом со средствами защиты. В некоторых реализациях внутренние пользователи должны соединится со средством защиты и аутентифицировать себя для соединения с внешним миром. Пользователям могут индивидуально присваиваться привилегии. Весь трафик из внешнего мира будет фильтроваться средством защиты хоста и может анализироваться по определенным критериям.

Некоторые хосты со средствами защиты работают как прокси. Когда внутренний пользователь запрашивает информацию из внешнего мира, производится соединение с прокси, который и получает эту информацию, а затем передает ее внутреннему пользователю.

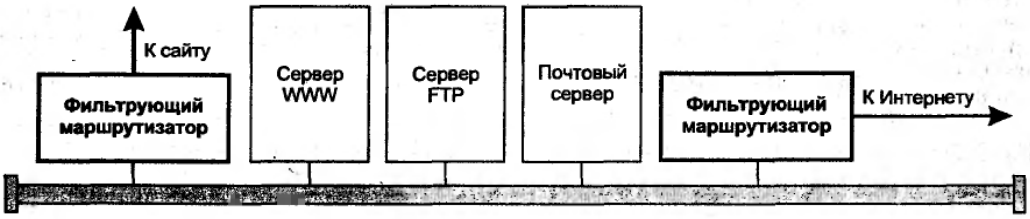

Рис. 6.17. Защита сайта с помощью демилитаризованной зоны

Для большей защиты сайты можно установить в режим "демилитаризованной зоны" локальной сети, разместив хосты защиты и все внутренне доступные прикладные серверы в локальной сети, защищенной фильтрующим маршрутизатором. На рис. 6.17 показана такая зона локальной сети, используемая для защиты от внешнего вторжения.

Применение защитного прокси позволяет присваивать компьютерам сайта личные IP-адреса (не известные внешним пользователям, которым доступен только адрес прокси или средства защиты.— *Прим. пер.*). В этом случае только для систем из демилитаризованной зоны локальной сети потребуются уникальные общедоступные адреса.

6.20 Замечания о производительности IP

Производительность интернета зависит от количества доступных ресурсов на хостах и маршрутизаторах и от эффективности их использования. К таким ресурсам относятся:

- Полоса пропускания пересылки информации
- Объем буферной памяти
- Скорость работы центрального процессора (ЦП)

Совершенных механизмов работы протоколов не существует. Разработка протоколов требует компромисса между широтой возможностей и эффективностью.

6.20.1 Полоса пропускания

IP эффективно использует полосу пропускания. Датаграммы помещаются в очередь для пересылки в точку следующего попадания, как только станет доступна полоса пропускания (bandwidth; по традиции мы будем использовать термин "полоса пропускания", хотя больший смысл имеет термин "доля производительности сети".— *Прим. пер.*). В результате удается избежать потерь от резервирования полосы пропускания для конкретного трафика или ожидания подтверждения пересылки.

Более того, существуют новые протоколы маршрутизации IP с большими возможностями: они могут распараллеливать трафик по нескольким путям и динамически выбирать маршрутизаторы, чтобы исключить перегрузки на отдельных участках пути следования датаграмм. Применение таких протоколов позволяет улучшить использование доступных ресурсов для пересылки информации.

Однако появляется небольшая перегрузка из-за управляющих сообщений, единственным источником которых становится протокол ICMP.

В результате проявляются и некоторые негативные свойства. Когда трафик направляется из высокоскоростной локальной сети в линию "точка-точка" с малой полосой пропускания, датаграммы начинают скапливаться в очереди маршрутизатора. Увеличивается время доставки от источника к точке назначения, и некоторые датаграммы отбрасываются. В этом случае требуется повторная пересылка датаграмм, еще более увеличивающая нагрузку на сеть и уменьшающая ее пропускную способность.

Отметим также, что при перегрузке сети, доставка датаграмм замедляется и становится менее надежной. Однако некоторые очень эффективные алгоритмы позволяют TCP немедленно реагировать на перегрузки посредством сокращения объема пересылаемых данных и снижения уровня ретрансляции.

Эти алгоритмы оказывают существенное влияние на производительность сети и поэтому стали неотъемлемой частью стандарта TCP (см. главу 10).

Производители маршрутизаторов энергично создают все более совершенные устройства, позволяющие обрабатывать десятки тысяч датаграмм в секунду. Для получения высокой производительности следует также внимательно отнестись к конфигурированию сети, чтобы предполагаемое максимальное использование памяти составляло примерно 50% от общего объема буферной памяти.

6.20.2 Использование буфера

Протокол IP, производящий пересылку датаграммы, несет ответственность за ее доставку. Для тех случаев, когда датаграмма по тем или иным причинам не попала в точку назначения, предусмотрен буфер датаграмм, позволяющий произвести операцию пересылки снова. В свою очередь, IP хоста назначения должен выделить некоторое буферное пространство для сборки фрагментированных датаграмм.

6.20.3 Ресурсы центрального процессора

Обработка датаграмм не приводит к большой загрузке центрального процессора (ЦП). Анализ заголовка достаточно прост. Не требуется сложного программного обеспечения для обслуживания тайм-аутов и повторной трансляции.

Вследствие динамических изменений и отсутствия соединений протокол IP требует обработки информации о маршрутизации на каждой системе попадания. Однако это реализуется простым просмотром таблицы, что выполняется достаточно быстро даже при большом размере таблиц.

Выполняемый маршрутизаторами анализ безопасности замедляет обработку, особенно при длинном списке условий для проверки каждой датаграммы.

6.21 Дополнительные сведения о многоадресных рассылках

Существует класс IP-адресов, используемых в *многоадресных* рассылках (см. главу 5), позволяющий маршрутизировать датаграмму от источника к группе систем, заданной одним из адресов класса D. Технологии и протоколы поддержки многоадресных рассылок в приложениях (например, в конференциях) существенно улучшились и расширили свои возможности за последние несколько лет.

В этом разделе мы кратко рассмотрим некоторые из используемых в настоящее время реализаций многоадресных рассылок. Но сначала приведем следующие факты:

- Отправитель многоадресной рассылки может не являться членом группы этой рассылки.
- Некоторые адреса для многоадресной рассылки стандартизированы и неизменны. Они зарегистрированы в IANA и опубликованы в RFC *Assigned Numbers*.
- Временные адреса многоадресной рассылки выбираются некоторым текущим процессом администрирования — их уникальность не гарантируется.
- Адрес многоадресной рассылки "*все хосты*" — 224.0.0.1 — уникален. Датаграммы, посланные группе всех хостов, никогда не могут покинуть локальную подсеть.
- Протокол *IGMP* обеспечивает механизм для разрешения многоадресным маршрутизаторам определять принадлежность хостов к группе многоадресной рассылки. IGMP рассматривается как составная часть IP. Сообщения IGMP переносятся датаграммами IP со значением 2 в поле протокола.

- *Многоадресный маршрутизатор* — это любая система, выполняющая специальное программное обеспечение многоадресной маршрутизации, которое может выполняться на обычных маршрутизаторах или хостах, сконфигурированных для выполнения многоадресной рассылки.

Рассмотрим сценарий работы многоадресного хоста:

- Хост, который хочет подключиться к группе и получать многоадресные рассылки, начинает прослушивать адрес многоадресной рассылки для всех хостов (224.0.0.1).
- Если хост хочет подключиться к конкретной группе, он должен сообщить об этом всем многоадресным маршрутизаторам по локальной связи. Для этого он отправляет *сообщение-отчет* IGMP по адресу нужной группы многоадресной рассылки. Поле TTL такого сообщения установлено в 1, и сообщение не может покинуть локальную подсеть.
- Затем хост начинает прослушивать датаграммы, посланные по адресу многоадресной рассылки.
- Кроме того, хост реагирует на периодические запросы от локальных маршрутизаторов и отвечает соответствующим отчетом.
- Для выхода из группы хост просто прекращает прослушивание на адрес этой группы и перестает направлять отчеты в группу.

Рассмотренные действия хоста слишком прямолинейны. Маршрутизация должна быть несколько сложнее и поэтому находится в стадии совершенствования. Рассмотрим действия в маршрутизаторе:

- Многоадресный маршрутизатор прослушивает все интерфейсы для получения отчетов от хостов. Для каждого из его интерфейсов создается список всех многоадресных групп, имеющих не менее одного активного члена в подсети, доступ к которой выполняется через данный маршрутизатор.
- Маршрутизатор должен посылать другим маршрутизаторам список адресов активных групп для каждой из подключенных к нему подсетей.
- Поскольку хосты достаточно молчаливы, маршрутизатору приходится периодически проверять локальные системы на принадлежность к конкретной группе. Для этого он время от времени отправляет *запрос* по адресу "*все хосты*". Каждый хост группы будет ожидать в течение произвольного промежутка времени. Первый из откликнувшихся укажет в своем ответе *адрес группы*. Маршрутизатор и все системы этой группы услышат такой ответ. Поскольку маршрутизатор после этого уже знает, что в группе есть хотя бы один активный член, остальные ответы уже не требуются.
- Когда маршрутизатор получает датаграмму многоадресной рассылки, он пересылает ее в каждую подключенную к нему подсеть, в которой находится член этой группы. Маршрутизатор может также переслать датаграмму другому многоадресному маршрутизатору.

IGMP-сообщение хоста имеет формат, показанный на рис. 6.18. Значение типа 1 определяет Host Membership Query (запрос о членстве хоста), а значение 2 — Host Membership Report (отчет о членстве хоста).

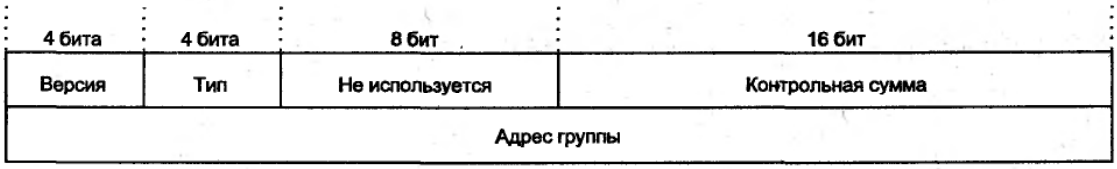

Рис. 6.18. Формат сообщения IGMP от хоста

6.22 Рекомендуемая литература

Протокол IP определен в RFC 791. Изменения, корректировки и требования совместимости специфицированы в RFC 1122. RFC 1812 подробно описывает требования IP версии 4 к маршрутизаторам и содержит подробные сведения об операциях таких маршрутизаторов.

Варианты безопасности Министерства обороны обсуждаются в RFC 1108. Вычислению контрольной суммы в Интернете посвящены RFC 1071, 1141 и 1624. RFC 815 представляет эффективные алгоритмы для сборки после фрагментации датаграммы в хосте получателя.

RFC 1112 специфицирует расширения хостов для многоадресных рассылок в IP.

Глава

7

Протокол ICMP

7.1 Введение

Протокол IP имеет ясную и элегантную структуру. В нормальных ситуациях IP очень эффективно использует для пересылки память и ресурсы. Однако что произойдет в нестандартной ситуации? Что может прервать бесцельное блуждание датаграммы до завершения ее времени жизни после краха маршрутизатора и неисправности в сети? Кто предупредит приложение о прекращении отправки датаграмм в недостижимую точку назначения?

Средства для лечения таких неисправностей предоставляет *протокол управляющих сообщений Интернета* (Internet Control Message Protocol — ICMP). Он выполняет роль сетевого помощника, способствуя маршрутизации в хостах и обеспечивая сетевого администратора средствами определения состояния сетевых узлов. Функции ICMP являются важной частью IP. Все хосты и маршрутизаторы должны быть способны генерировать и обрабатывать сообщения ICMP. При правильном использовании эти сообщения могут улучшить выполнение сетевых операций.

Сообщения ICMP пересылаются в датаграммах IP с обычным заголовком IP (см. рис. 7.1), имея в поле *протокола* значение 1.

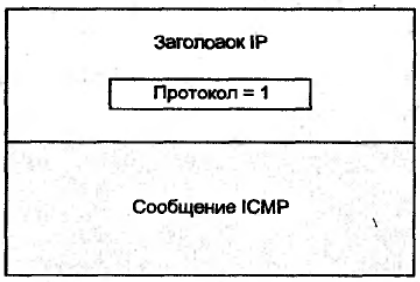

Рис. 7.1. Пакетирование сообщения ICMP

7.2 Сообщения об ошибках ICMP

Бывают ситуации, приводящие к отбрасыванию (удалению из сети) датаграммы IP. Например, точка назначения может стать недоступной из-за обрыва связи. Или может завершиться время жизни датаграммы. Маршрутизатор не сможет переслать длинную датаграмму при запрещении фрагментации.

При отбрасывании датаграммы по адресу ее источника направляется сообщение ICMP, указывающее на возникшую проблему. На рис. 7.2 показано сообщение ICMP, направленное к источнику датаграммы.

ICMP быстро сообщит системе о выявленной проблеме. Это очень надежный протокол, поскольку указание на ошибки не зависит от наличия сетевого центра управления.

Однако в использовании сообщений ICMP имеются некоторые недостатки. Например, если недостижима точка назначения, то сообщение будет распространяться до источника по всей сети, а не на станцию сетевого управления.

Реально ICMP не имеет средств предоставить отчет об ошибках выделенному операционному центру. Для этого служит протокол SNMP (см. главу 20).

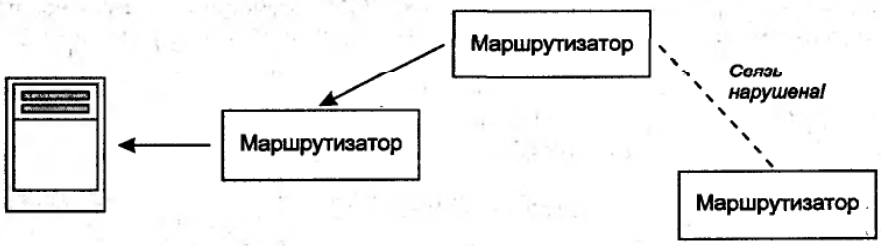

Рис. 7.2. Сообщение ICMP направляется по пути трафика.

7.2.1 Типы сообщений об ошибках

На рис. 7.3 показаны обобщенные сообщения, формируемые маршрутизатором и хостом назначения для отчета о возникшей проблеме. В таблице 7.1 перечислены формальные имена сообщений об ошибках ICMP.

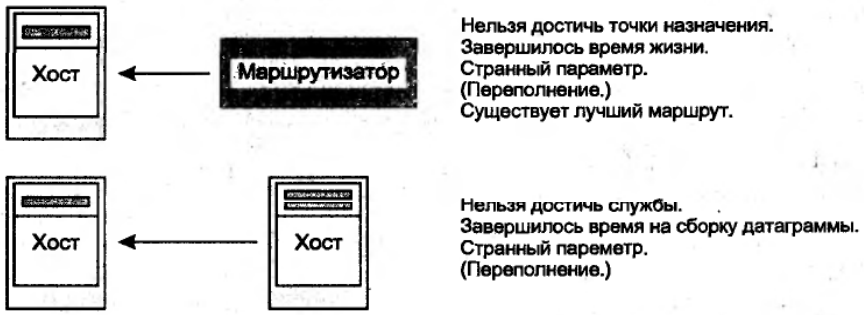

Рис. 7.3. Типы сообщений об ошибках ICMP

Таблица 7.1

Сообщения об ошибках ICMP

Сообщение	Описание
Destination Unreachable (недостижимая точка назначения)	Датаграмма не может достичь хоста назначения, утилиты или приложения.
Time Exceeded (время закончилось)	Маршрутизатор определил завершение времени жизни, или закончилось время на сборку фрагментов в хосте назначения.
Parameter Problem (проблема с параметром)	В заголовке IP неверный параметр.
Source Quench (подавление источника)	Перегружен маршрутизатор или система назначения (системам рекомендуется *не отправлять* это сообщение).
Redirect (перенаправление)	Хост направил датаграмму на неверный локальный маршрутизатор.

7.2.2 Обязанность по отправке сообщения ICMP

Протокол ICMP определяет, что сообщения *могут* или *должны* быть посланы в каждом случае, но он не требует выдавать сообщения ICMP о *каждой* ошибке.

В этом есть здравый смысл. Основным назначением маршрутизатора в сети является пересылка датаграмм. Перегруженный хост назначения должен уделять больше времени доставке датаграмм в приложения, а не указанию на ошибки удаленному хосту. Именно поэтому не формируются сообщения о случайном отбрасывании датаграммы.

7.2.3 Входящие сообщения ICMP

Что происходит при получении хостом сообщения ICMP? Рассмотрим пример, когда производится попытка обращения по зарезервированному (и, следовательно, недостижимому) адресу сети:

```
> telnet 10.1.1.1
Trying 10.1.1.1 ...
telnet: connect: Host is unreachable
```

Произошло то, что и должно было произойти,— в сообщении указано на недостижимость хоста (Host is unreachable).

Чтобы определить, *какой* из маршрутизаторов послал сообщение ICMP, можно использовать команду *traceroute*:

```
> traceroute 10.1.1.1
traceroute to 10.1.1.1 (10.1.1.1), 30 hops max, 40 byte packets
1 nomad-gateway (128.121.50.50) 2 ms 2 ms 2 ms
2 liberty-gateway (130.94.40.250) 91 ms 11 ms 78 ms
3 border2-hssi2-0.NewYork.mci.net (204.70.45.9) !H !H !H
```

Маршрутизатор New York послал сообщение *Destination Unreachable*, которое отображается на экране как !H.

Функции *traceroute* основаны на ICMP-сообщении *Time Expired* и формируются следующим образом:

- Создается короткое сообщение UDP, которое имеет заголовок IP с установленным в 1 полем TTL.
- Трижды отправляется датаграмма.
- Первый маршрутизатор (в примере — *nomad-gateway*) устанавливает значение Time-to-Live (время жизни) в 0, отбрасывает датаграмму и отправляет источнику ICMP-сообщение *Time Expired*.
- Функция *traceroute* идентифицирует пославший сообщение маршрутизатор и трижды выводит само сообщение.
- Значение Time-to-Live устанавливается в 2, и сообщение посылается дальше.
- Процесс повторяется с увеличением Time-to-Live на каждом шаге.

Если можно достичь точки назначения, то в итоге можно получить полный путь до него.

7.3 Когда не нужно посылать сообщение ICMP

Напомним, что ICMP-сообщение об ошибке посылается, когда в сети не все благополучно. Важно обеспечить, чтобы трафик ICMP не перегружал сети, делая ситуацию еще хуже. Для этого протокола требуется ввести несколько очевидных ограничений. ICMP не должен формировать сообщения о:

- Маршрутизации и доставке ICMP-сообщений messages
- Широковещательных и многоадресных датаграммах
- Фрагментах датаграмм, кроме первых
- Сообщениях, чей адрес источника не идентифицирует уникальный хост (например, IP-адреса источников 127.0.0.1 или 0.0.0.0)

7.4 Формат сообщения ICMP

Сообщение ICMP переносится в части данных датаграммы IP. Каждое сообщение ICMP начинается тремя одинаковыми полями: полем *типа* (Type), полем *кода* (Code), обеспечивающим более подробное описание ошибки, и полем *контрольной суммы* (Checksum). Формат оставшейся части сообщения определяется типом сообщения.

Сообщение об ошибке ICMP обрамляется заголовком IP. Добавляются первые 8 октетов датаграммы, которая привела к ошибке. Эти сведения позволяют проанализировать причину ошибки, поскольку

содержат информацию о предполагаемом назначении датаграммы и целевом протоколе четвертого уровня. Дополнительные 8 байт позволяют определить коммуникационный элемент приложения (более подробно об этом см. в разделе о протоколах TCP и UDP).

В сообщение включается и контрольная сумма ICMP, начиная от поля *Type*.

7.4.1 Сообщение Destination Unreachable

Существует много причин прекращения доставки датаграммы. Разорванная связь физически не позволит маршрутизатору достичь подсети назначения или выполнить пересылку в точку следующего попадания. Хост назначения может стать недоступным при отключении его для проведения профилактики.

Как уже отмечалось в главе 6, современные маршрутизаторы имеют хорошие средства обеспечения безопасности. Они могут быть сконфигурированы для просмотра входящего в сеть трафика. При запрещении сетевым администратором доступа к точке назначения датаграмма также не может быть доставлена.

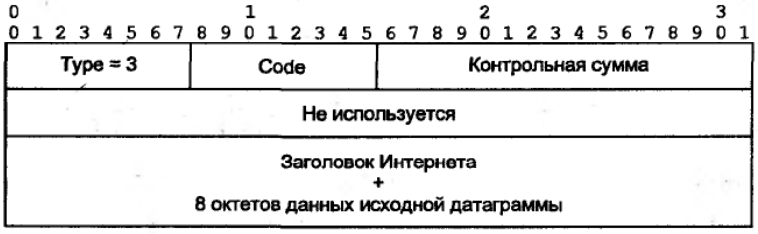

Рис. 7.4. Формат ICMP-сообщения Destination Unreachable

Формат сообщения *Destination Unreachable* показан на рис. 7.4. Поле Type (в нашем случае 3) идентифицирует именно этот *тип* сообщения. Поле *Code* отражает причину отправки сообщения. Полный список кодов этого поля представлен в таблице 7.2.

Таблица 7.2

Коды ошибок сообщения Destination Unreachable

Код	Смысл
0	Сеть недостижима.
1	Хост недостижим.
2	Запрашиваемый протокол не поддерживается в точке назначения.
3	Порт недостижим (недоступно удаленное приложение).
4	Необходима фрагментация, но установлен флаг "Не фрагментировать".
5	Неверен маршрут от источника.
6	Неизвестна сеть назначения.
7	Неизвестен хост назначения.
8	Хост источника изолирован.
9	Административно запрещены коммуникации с сетью назначения.
10	Административно запрещены коммуникации с хостом назначения.
11	Сеть недостижима для заданного типа обслуживания.
12	Хост недостижим для заданного типа обслуживания.

7.4.2 Сообщение Time Exceeded

Пересылаемая датаграмма может быть отброшена по тайм-ауту при уменьшении до нуля ее времени жизни (TTL). Еще один тайм-аут может возникнуть в хосте назначения, когда завершится время, выделенное на сборку, а прибыли еще не все фрагменты датаграммы. В обоих случаях формируется сообщение *Time Exceeded* для источника датаграммы. Формат этого сообщения показан на рис. 7.5.

Значения кодов (см. таблицу 7.3) отражают причину тайм-аута.



Рис. 7.5. Формат ICMP-сообщения Time Exceeded

Таблица 7.3

Коды сообщения Time Exceeded

Код	Смысл
0	Завершилось время жизни датаграммы.
1	Завершилось время на сборку фрагментов датаграммы.

7.4.3 Сообщение Parameter Problem

ICMP-сообщение *Parameter Problem* используется для отчета об ошибках, не специфицированных в кодах других сообщений. Например, в полях вариантов может появиться неверная информация, не позволяющая правильно обработать датаграмму, в результате чего датаграмма будет отброшена. Более часто проблемы с параметрами возникают из-за ошибок в реализации, когда система пытается записать параметры в заголовок IP.

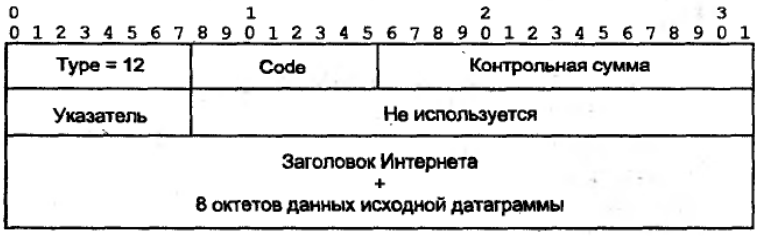

Рис. 7.6. Формат ICMP-сообщения Parameter Problem

Поле *Pointer* (указатель) сообщения *Parameter Problem* идентифицирует октет, в котором выявлена ошибка. На рис. 7.6 показан формат сообщения *Parameter Problem*, а в таблице 7.4 — значения кодов ошибок.

Таблица 7.4

Коды сообщения Parameter Problem

Код	Смысл
0	Значение в поле указателя специфицирует ошибочный октет.
1	Отсутствует требуемый вариант (используется военными для указания на отсутствие параметров безопасности)
2	Неверная длина

7.4.4 Проблемы перегрузок

Протокол IP очень прост: хост или маршрутизатор обрабатывают датаграмму и посылают ее как можно быстрее. Однако доставка не всегда проходит гладко. Могут возникнуть различные проблемы.

Когда один или несколько хостов отправляют трафик UDP на медленный сервер, то на последнем может возникнуть перегрузка, что приведет к отбрасыванию сервером некоторой части этого трафика.

Маршрутизатор может переполнить свои буферы и далее будет вынужден отбрасывать некоторые поступающие датаграммы. Медленное соединение через региональную сеть (например, на скорости 56 Кбит/с) между двумя скоростными локальными сетями (например, в 10 Мбит/с) может создать затор на пути следования датаграмм. Из-за этого в сети возникнут перегрузки, которые также приведут к отбрасыванию датаграммы и, следовательно, к созданию еще большего трафика.

7.4.5 Сообщение Source Quench

Сообщение *Source Quench* (подавление источника) показано на рис. 7.7. Оно позволяет попытаться решить проблему перегрузок, хотя и не всегда успешно. Механизмы для подавления источника перегрузки сети должны создавать разработчики конкретных продуктов, но остается открытым конкретный вопрос:

Когда и кому маршрутизатор или хост должен отправлять сообщение Source Quench?

Обычно ICMP-сообщение указывает хосту источника на причину отбрасывания посланной им датаграммы. Однако при перегрузке такое сообщение может не дойти до этого хоста, генерирующего очень напряженный сетевой трафик. Кроме того, очень расплывчаты требования к обработке поступающих сообщений *Source Quench*.

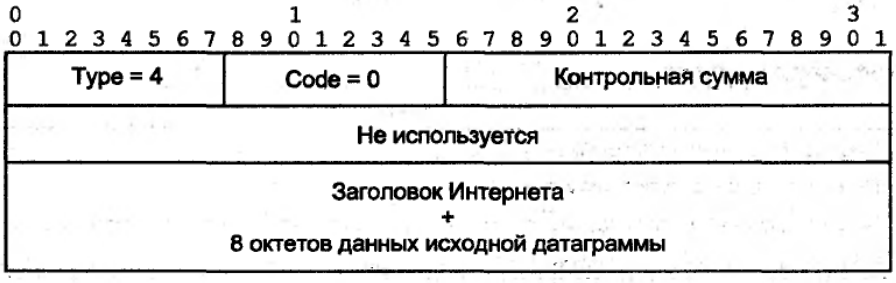

Рис. 7.7. Формат ICMP-сообщения Source Quench

Текущий документ по *требованиям к хостам* (RFC 1812) оговаривает в качестве особого пункта, что сообщения *Source Quench* вовсе *не нужно* посылать. Работа должна выполняться более совершенным механизмом управления нагрузкой в сети.

7.4.6 Сообщения Redirect

К локальной сети может быть подключено более одного маршрутизатора. Когда локальный хост посылает датаграмму не на тот маршрутизатор, последний пересылает ее и отправляет хосту источника ICMP-сообщение *Redirect* (перенаправление), как показано на рис. 7.8. Хост должен переключить последующий трафик на более короткий путь.

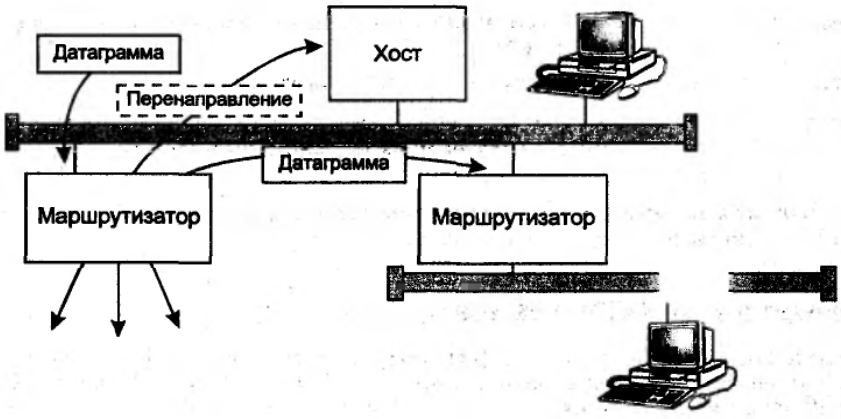

Рис. 7.8. Коррекция маршрутизации на хосте посредством сообщения Redirect

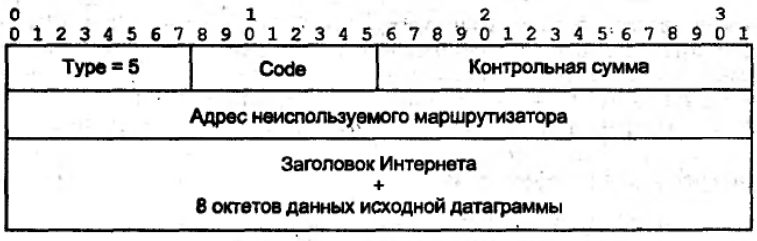

Рис. 7.9. Формат ICMP-сообщения Redirect

Сообщение *Redirect* используется и для выключения маршрутизатора системным администратором. Хост может быть сконфигурирован с единственным маршрутизатором по умолчанию; при этом он будет динамически определять возможности пересылки через другие маршрутизаторы.

Формат сообщения о перенаправлении показан на рис. 7.9. Коды этого сообщения перечислены в таблице 7.5. Некоторые протоколы маршрутизации способны выбирать путь доставки на основе содержимого поля *типа обслуживания* (TOS) датаграммы. Коды 2 и 3 предоставляют некоторые сведения для такого выбора.

Таблица 7.5

Коды перенаправления

Код	Смысл
0	Перенаправление датаграммы в сеть
1	Перенаправление датаграммы в хост
2	Перенаправление датаграммы в сеть на основе значения из поля типа обслуживания
3	Перенаправление датаграммы в хост на основе значения из поля типа обслуживания

7.4.7 Управление поступающими сообщениями ICMP

Что должен делать хост, получивший сообщение ICMP? Реализации различных разработчиков по-разному отвечают на этот вопрос. В некоторых из них хосты игнорируют все или многие такие сообщения. Стандарты TCP/IP оставляют большую свободу выбора в решении этого вопроса. Для различных типов сообщений ICMP предлагаются следующие рекомендации:

Destination Unreachable	Доставить ICMP-сообщение на транспортный уровень. Выполняемые действия должны зависеть от того, является ли причина вывода сообщения временной или постоянной (например, административный запрет на пересылку).
Redirect	Хост *обязан* обновить таблицу маршрутизации.
Source Quench	Доставить ICMP-сообщение на транспортный уровень или в модуль обработки ICMP.
Time Exceeded	Доставить на транспортный уровень.
Parameter Problem	Доставить ICMP-сообщение на транспортный уровень с необязательным уведомлением пользователя.

Иногда ошибки должны обрабатываться совместно операционной системой, коммуникационным программным обеспечением и сетевым приложением.

7.5 Исследование MTU по пути

При пересылке большого объема данных (например, при копировании файлов по сети) с одного хоста на другой размер датаграмм существенно влияет на производительность. Заголовки IP и TCP требуют не менее 40 дополнительных байт.

- Если данные пересылаются в 80-байтовых датаграммах, дополнительная нагрузка составит 50%.

- Если данные пересылаются в 400-байтовых датаграммах, дополнительная нагрузка составит 10%.
- Если данные пересылаются в 4000-байтовых датаграммах, дополнительная нагрузка составит 1%.

Для минимизации дополнительной нагрузки лучше отсылать датаграммы наибольшего размера. Однако этот размер ограничивается значением максимального элемента пересылки (Maximum Transmission Unit — MTU) для каждого из носителей. Если датаграмма будет слишком большой, то она будет фрагментирована, а этот процесс снижает производительность. (С точки зрения пользователя, качество сети определяется двумя параметрами: интервалом пересылки (от начала пересылки до ее завершения) и временем ожидания (задержкой доступа к сети, занятой другими пользователями). Увеличение размера датаграммы приводит к снижению интервала пересылки, но увеличению ожидания для других пользователей. Грубо говоря, нагрузка на сеть будет выглядеть как пиковые импульсы с очень небольшой нагрузкой между ними, что считается самым неудачным вариантом загрузки сети. Гораздо лучше, когда сеть нагружается равномерно.— *Прим. пер.*)

Многие годы хосты избегали фрагментации, устанавливая эффективное значение MTU для пересылки в 576 октетов для всех нелокальных хостов. Это часто приводило к ненужному снижению производительности.

Гораздо полезнее заранее знать наибольший допустимый размер датаграммы, которую можно переслать по заданному пути. Существует очень простой механизм *исследования MTU по пути* (Path MTU discovery), позволяющий узнать это значение. Для такого исследования:

- Флаг *"Не фрагментировать"* заголовка IP устанавливают в 1.
- Размер MTU по пути первоначально устанавливают в значение MTU для локального интерфейса.
- Если датаграмма будет слишком велика для одного из маршрутизаторов, то он пошлет обратно ICMP-сообщение *Destination Unreachable* с кодом 4.
- Хост источника уменьшит размер датаграммы и повторит попытку.

Какое же значение нужно выбрать для следующей попытки? Спецификация IP предполагает сохранение значения MTU и его доступность для протоколов транспортного уровня. Если маршрутизатор имеет современное программное обеспечение, то он будет включать в пересылаемое дальше по сети сообщение *Destination Unreachable* размер MTU (см. рис. 7.10). Иногда средства защиты конфигурируются на полное исключение *всех* входящих сообщений ICMP, что не позволяет использовать механизм определения MTU по пути следования датаграммы.

Поскольку пути пересылки могут меняться динамически, флажок "*Не фрагментировать*" нужно устанавливать во всех коммуникационных датаграммах. При необходимости маршрутизатор будут посылать сведения об обновлениях.

Если маршрутизатор использует устаревшее программное обеспечение, он не сможет предоставить значение MTU для следующего попадания. В этом случае значение для следующей попытки будет выбираться из списка стандартных размеров MTU (см. главу 4) с постепенным уменьшением для каждой новой попытки до достижения значения, нужного для коммуникации с удаленным хостом.

Разумеется, изменение пути следования может создать предпосылки для использования большего размера MTU. В этом случае система, согласовавшая небольшой размер MTU, будет пытаться его увеличить, если такое улучшение будет возможно.

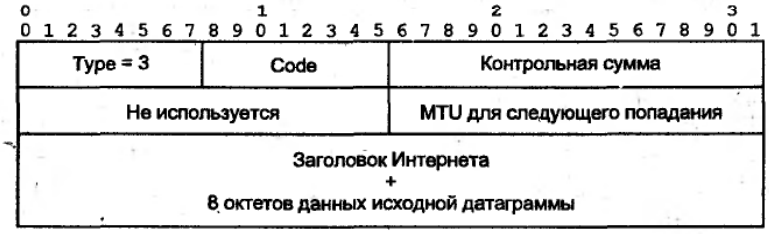

Рис. 7.10. Сообщение Destination Unreachable приносит результат исследования размера MTU.

7.6 Сообщения запросов ICMP

Не все сообщения ICMP сигнализируют об ошибках. Некоторые из них извлекают из сети полезные сведения. Работает ли хост X? Не выключен ли хост Y? Как долго движется датаграмма до хоста Z и обратно? Какова маска подсети хоста источника?

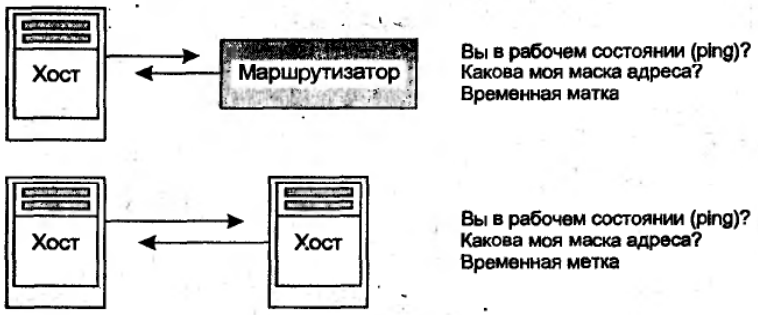

Рис. 7.11. Сообщение запроса ICMP

Ответы на эти вопросы дают следующие сообщения ICMP:

- *Эхо-запросы* и *эхо-ответы* обеспечивают обмен информацией между хостами и маршрутизаторами.
- Запросы и ответы о *маске адреса* позволяют системе исследовать присвоенную интерфейсу маску адреса.
- Запросы и ответы *временной метки* служат для извлечения сведений об установке времени на целевой системе. Ответы на такие запросы дают информацию, необходимую для оценки времени обработки датаграмм на хосте.

На рис. 7.11 представлено обслуживание запросов ICMP. Программа *Ping* посылает эхо-сообщение "Вы в рабочем состоянии?", которое используется в ежедневной работе сетевого администратора. Запросы о *маске адреса* применяются от случая к случаю, а запросы о *временной метке* вообще редко.

7.6.1 Эхо-запросы и эхо-ответы

Эхо-запросы (Echo Request) и *эхо-ответы* (Echo Reply) применяются для проверки активности системы. Код типа 8 применяется в запросах, а код 0 — в ответах. Количество октетов в поле данных переменно и может выбираться отправителем.

Отвечающая сторона должна послать обратно те же самые данные, которые были получены. Поле *идентификатора* служит для сравнения ответа с исходным запросом. Последовательный номер эхо-сообщения может применяться для тестирования, на каком участке произошел обрыв сети, и для вычисления приблизительного времени на путь туда и обратно. При этом идентификатор не меняется, а последовательный номер (начиная от 0) увеличивается на единицу для каждого сообщения. Формат эхо-сообщения показан на рис. 7.12.

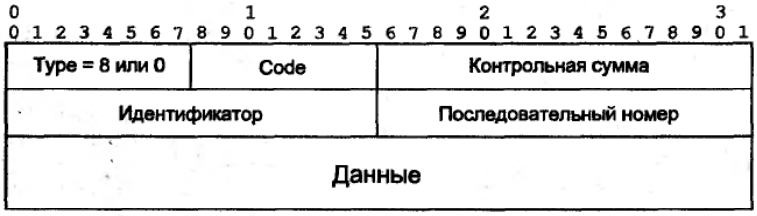

Рис. 7.12. Формат ICMP-сообщений Echo Request и Echo Reply

Широко известная команда *ping* доступна почти во всех системах TCP/IP, а ее работа основана на ICMP-сообщениях для эхо-запросов и эхо-ответов. В приведенном ниже диалоге сначала тестируется хост *ring.bell.com*. Затем отсылается последовательность из 14 сообщений, содержащих по 64 октета каждое. Отметим, что сообщения 0, 1 и 2 были потеряны. Справа приводятся сведения о пути туда и обратно.

```
> ping ring.bell.com
ring.bell.com is alive
```

```
> ping -s ring.bell.com 64 14
64 bytes from ring.bell.com: icmp_seq=3. time = 21. ms
64 bytes from ring.bell.com: icmp_seq=4. time = 18. ms
64 bytes from ring.bell.com: icmp_seq=5. time = 17. ms
64 bytes from ring.bell.com: icmp_seq=6. time = 19. ms
64 bytes from ring.bell.com: icmp_seq=7. time = 17. ms
64 bytes from ring.bell.com: icmp_seq=8. time = 17. ms
64 bytes from ring.bell.com: icmp_seq=9. time = 17. ms
64 bytes from ring.bell.com: icmp_seq=10. time = 18. ms
64 bytes from ring.bell.com: icmp_seq=11. time = 17. ms
64 bytes from ring.bell.com: icmp_seq=12. time = 17. ms
64 bytes from ring.bell.com: icmp_seq=13. time = 17. ms

—ring.bell.com PING Statistics—
14 packets transmitted, 11 packets received, 21% packet loss
round-trip (ms) min/avg/max = 17/17/21
```

7.6.2 Маска адреса

Напомним, что организация может разделить поле своего локального адреса на часть подсети и часть хоста. Когда включается система, она может быть сконфигурирована так, что не будет заранее знать, сколько бит было присвоено полю адреса подсети. Чтобы выяснить этот вопрос, система посылает широковещательный *запрос на определение маски адреса* (Address Mask Request).

Ответ должен быть получен от сервера, авторизированного для управления маской адреса сервера. Обычно в качестве такого сервера применяется маршрутизатор, но может использоваться и хост. В ответе в полях сети и подсети установлены единицы, определяя 32-разрядное поле маски адреса.

Сервер маски адреса может быть сконфигурирован так, что, даже при отключении от сети на какое-то время, он будет далее передавать широковещательные сообщения *Address Mask Reply*, как только станет активным. Это предоставляет шанс на получение нужной информации системам, которые были запущены в то время, когда сервер был неактивен.

На рис. 7.13 показан формат *запроса маски адреса* и *ответа* на него. Тип 17 применяется для запроса, а тип 18 — для ответа. В общем случае можно игнорировать идентификатор и последовательный номер.

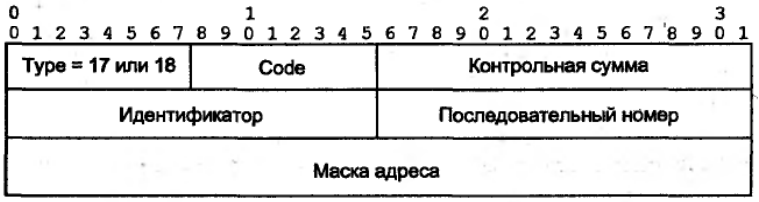

Рис. 7.13. Формат ICMP-сообщений Address Mask

На практике более предпочтительный метод определения маски адреса предоставляют протоколы загрузки, например *Dynamic Host Configuration Protocol* или *BOOTP*. Эти протоколы более эффективны, поскольку обеспечивают полный набор конфигурационных параметров. Кроме того, операции выполняются более точно, в том числе и некорректные.

7.6.3 Временная метка и ответ на Timestamp

Сообщение с ответом на *Timestamp* предоставляет сведения о времени в системе. Оно предназначено для оценки буферизации и обработки датаграммы на удаленной системе. Отметим следующие поля:

Originate timestamp (исходная временная метка)	Время последнего обращения к сообщению в системе-отправителе
Receive timestamp (временная метка получения)	Время первого обращения к сообщению отвечающей системы
Transmit timestamp (временная метка пересылки)	Время последнего обращения к сообщению отвечающей системы

По возможности, возвращаемое время должно измеряться в миллисекундах относительно полуночи по универсальному времени (Universal Time), которое ранее называлось временем по Гринвичу (Greenwich Mean Time). Большинство реализаций реально возвращает одно и то же время в полях *Receive timestamp* и *Transmit timestamp*.

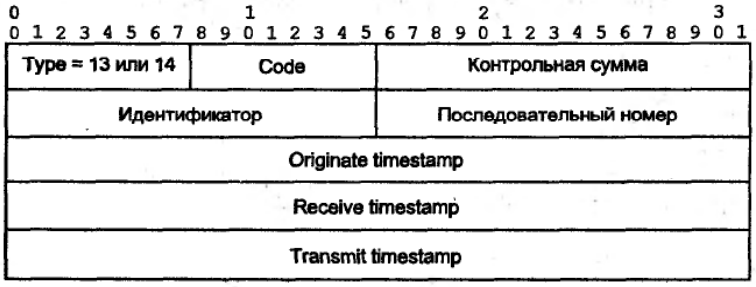

Рис. 7.14. Формат сообщений запросов и ответов о временной метке

Протокол ICMP обеспечивает очень простой способ синхронизации систем по времени. Однако это несколько грубая синхронизация, поскольку на нее влияют задержки в сети. Существует более совершенный *протокол сетевого времени* (Network Time Protocol), который был разработан для синхронизации по времени в Интернете.

Тип 13 используется для *запросов*, а 14 — для *ответов*. Формат сообщения представлен на рис. 7.14.

7.7 Просмотр действий в ICMP

Ниже показана часть отчета о статистике протоколов команды netstat. Приведенный фрагмент посвящен протоколу ICMP. В отчете отражены операции ICMP, выполненные после последней инициализации.

```
> netstat -s

icmp:
  1075 calls to icmp_error

  Output histogram:
      echo reply: 231
      destination unreachable: 1075

  2 messages with bad code fields
  0 messages < minimum length
  21 bad checksums
  0 messages with bad length

  Input histogram:
      echo reply: 26
      destination unreachable: 1269
      source quench: 2
      echo: 231
  231 message responses generated
```

Система отправила 1075 сообщений *Destination Unreachable*. Был получен 231 запрос *Echo Requests*, на каждый из которых был отправлен ответ. Было получено 26 ответов *Echo Replies*.

Локальная система зафиксировала 21 сообщение ICMP, полученное с неверной контрольной суммой *ICMP*.

Системой было получено 1269 сообщений *Destination Unreachable* и 2 сообщения *Source Quenches*.

Следующий далее отчет команды *netstat* содержит сведения о маршрутизации. Видно, что через сообщения *Redirect* были динамически обнаружены новые маршрутизаторы. Было 12 отчетов о *недостижимости точки назначения* (Destination Unreachable). Для выбора маршрута по умолчанию было использовано около 349 *подстановочных символов* (wildcard).

```
> netatat -rs
routing:
  0 bad routing redirects
  0 dynamically created routes
  2 new routers due to redirects
  12 destinations found unreachable
  349 uses of a wildcard route
```

7.8 Обнаружение маршрутов

Хотя многие локальные сети имеют единственный маршрутизатор по умолчанию, существует достаточное количество сетей, имеющих два или более маршрутизаторов.

Что происходит при подключении маршрутизатора к локальной сети? Сообщения о перенаправлении укажут системам на новые маршрутизаторы. Предположим, что произошел крах маршрутизатора по умолчанию.

Протокол *исследования маршрутов* (Router Discovery) предоставляет надежный метод, основанный на сообщениях ICMP, для отслеживания маршрутизаторов локальной сети. Основная идея состоит в периодическом объявлении маршрутизаторами о своем присутствии. Хостам нужно прослушивать такие сообщения.

Предпочтительным способом для объявления маршрутизатора о своем присутствии является отправка *многоадресной рассылки по адресу для всех систем* (224.0.0.1). Однако не все хосты поддерживают прием многоадресных рассылок, поэтому иногда применяется широковещательный адрес (255.255.255.255).

Подключающийся к сети хост может быть не способен к ожиданию при поиске маршрутизаторов локальной сети. Такой хост самостоятельно запрашивает маршрутизаторы об отправке их объявлений о присутствии на собственный адрес. Для этого лучше всего использовать сообщение *Router Solicitation* (настоятельная просьба к маршрутизаторам) в *многоадресной рассылке по адресу "все маршрутизаторы"* (224.0.0.2). Поскольку не все системы поддерживают многоадресные рассылки, иногда применяется широковещательная рассылка (255.255.255.255).

Типичный сценарий для маршрутизатора:

- Каждый интерфейс маршрутизатора конфигурируется с *адресом объявления* (advertisement address) — 224.0.0.1 или 255.255.255.255 — для локальной сети, подключенной к данному интерфейсу.
- При инициализации маршрутизатора и использовании многоадресной рассылки маршрутизатор начинает прослушивание адреса *многоадресной рассылки для всех маршрутизаторов* (224.0.0.2). Кроме того, прослушиваются и широковещательные рассылки.
- Маршрутизатор объявляет о своем присутствии всем подключенным к локальной сети хостам посредством трансляции сообщения *Router Advertisement* на адрес объявления такой локальной сети. В объявлении о присутствии перечисляются все IP-адреса маршрутизатора для данного интерфейса.
- Маршрутизатор напоминает о себе различными периодическими сообщениями Router Advertisement (рекомендуемый период 7 - 10 мин.).
- Маршрутизатор посылает *объявление* о присутствии при запросе об этом от хоста.

Для хоста сценарий выглядит так:

- Каждый интерфейс хоста конфигурируется с *Solicitation Address* — 224.0.0.2 или 255.255.255.255.
- При инициализации хоста начинается прослушивание *Router Advertisement.*
- При запуске хост может послать необязательное сообщение *Router Solicitation* по адресу настоятельных просьб (solicitation address). Маршрутизаторы ответят как по IP-адресу хоста, так и по адресу объявления.

- Когда хост услышит о новом маршрутизаторе, он добавит маршрут *по умолчанию* в свою таблицу маршрутизации. Этому элементу таблицы присваивается значение тайм-аута на время жизни (обычно 30 мин), которое указано в *Router Advertisement*.
- Тайм-аут на время жизни сбрасывается при получении нового объявления от маршрутизатора. Если время жизни завершается по тайм-ауту, элемент для маршрутизатора удаляется из таблицы маршрутизации хоста.
- Для объявления всем о корректном отключении от сети маршрутизатор может послать объявления с нулевым значением для времени жизни.

Если имеется более одного маршрутизатора, то как хост определяет тот, которому следует направить данную датаграмму? Каждое объявление маршрутизатора содержит номер *предпочтительного уровня* (preference level). Если таблица маршрутизации не содержит специальных указаний, выбирается маршрутизатор с *наибольшим* предпочтительным уровнем. Если он не сможет обеспечить наилучший маршрут, то ответит хосту ICMP-сообщением о перенаправлении.

В ICMP-сообщении *Router Advertisement* имеет значение типа 9, а *Router Solicitation* — 10.

7.8.1 Неисправные маршрутизаторы

Исследование маршрутов (маршрутизаторов) помогает хостам определить крах локального маршрутизатора, однако после достаточно длительного периода времени — возможно, через 30 мин. Реализация TCP/IP для хостов предполагает использование встроенного алгоритма для определения неисправности маршрутизатора. Его достоинства очевидны, например:

- Существование активного сеанса TCP/IP через маршрутизатор
- Фиксирование получения от маршрутизатора ICMP-сообщений о перенаправлении.

К числу недостатков можно отнести:

- Невозможность ответа на запросы ARP
- Множество последовательных тайм-аутов при повторной пересылке в TCP

Если есть причины считать маршрутизатор неисправным, окончательная проверка выполняется по запросу *ping*.

IP версии 6 обеспечивает наилучший способ исследования причин приостановки работы локальных хостов или маршрутизаторов.

7.9 Дополнительная литература

ICMP определен в RFC 792. RFC 1122 (требования к хостам) и RFC 1812 (требования к маршрутизаторам) содержат несколько очень полезных разъяснений. Исследованию маршрутов посвящен RFC 1256.

Исследование MTU по пути рассмотрено в RFC 1191, а дополнительные рекомендации представлены в RFC 1435.

Глава

8

Маршрутизация в IP

8.1 Введение

Маршрутизация — наиболее важная функция протокола IP. В больших сетях маршрутизаторы IP обмениваются информацией для сохранения корректного состояния своих таблиц маршрутизации. Каким образом это выполняется?

Единого протокола для изменения информации в таблицах маршрутизации IP не существует.

Поэтому сетевой администратор может выбрать любой протокол для маршрутизации информации, наиболее соответствующий требованиям конкретной сети. За прошедшие годы было разработано и улучшено несколько протоколов, ставших стандартами для групп производителей оборудования. По давней традиции они называются *протоколами внутреннего шлюза* (Interior Gateway Protocol — IGP). Иногда эту аббревиатуру расшифровывают как *Internal* Gateway Protocol, что по-русски означает то же самое.

Отделение способа изменения таблиц маршрутизации от оставшейся части IP — это очень удачная идея. Маршрутизация становится все более совершенной и эффективной, в то время как базовые основы IP остаются неизменными.

На сегодняшний день широко используется несколько протоколов IGP. Остается очень популярным *протокол информации о маршрутизации* (Routing Information Protocol — RIP), выбирающий маршрут на основе оценки простого счетчика попаданий.

Сайты с маршрутизаторами компании Cisco часто выбирают лицензированные протоколы этой компании — *протокол маршрутизации шлюзов Интернета* (Internet Gateway Routing Protocol — IGRP) или *улучшенный протокол IGRP* (Enhanced IGRP — EIGRP), в которых применяется весьма сложный способ измерения стоимости, учитывающий множество факторов, включая нагрузку и надежность.

Более сложный протокол, предлагающий *первым открывать кратчайший путь* (Open Shortest Path First — OSPF), применяется в больших сетях. Некоторые организации используют протокол *от одной промежуточной системы к другой* (Intermediate System to Intermediate System — IS-IS), который может маршрутизировать как трафик IP, так и OSI. Протоколы OSPF и IS-IS формируют подробные карты для сетей и генерируют пути перед выбором маршрута.

Предоставление свободы выбора протокола для организации конечного пользователя реализуется прекрасно. Однако необходим стандарт для маршрутизации в сети провайдера при соединении между собой сетей конечных пользователей. Хотя еще применяют устаревший *протокол внешнего шлюза* (Exterior Gateway Protocol — EGP), большинство провайдеров перешло на *протокол граничного шлюза* (Border Gateway Protocol — BGP).

В этой главе мы познакомимся с каждым из упомянутых выше протоколов и рассмотрим различия в предоставляемых ими возможностях.

8.2 Автономные системы

Как можно предоставить столько различных возможностей при выборе протокола маршрутизации? Модель Интернета разделяет весь мир (как всегда, имеется в виду сетевой мир.— *Прим. пер.*) на элементы, именуемые *автономными системами* (Autonomous System — AS). Грубо говоря, автономной системой является чья-то сеть. Более формальное определение:

> *Подключенный сегмент сетевой топологии, состоящий из набора подсетей (с подключенными к ним хостами) и взаимодействующий через набор маршрутизаторов (RFC 1812, Требования к IP версии 4).*

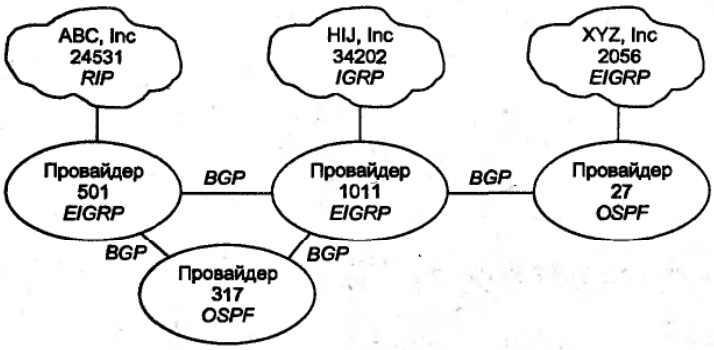

Рис. 8.1. Автономные системы и протоколы маршрутизации

Более важно то, что создающая автономную систему подсеть находится под единым управлением.

Типичной автономной системой является сеть компании или провайдера. Реально никому нет дела до автономной системы, пока не возникает потребность во взаимодействии с ней. В этом случае нужно зарегистрироваться в InterNIC и получить собственный *номер автономной системы* (Autonomous System Number). На рис. 8.1 показаны компании, провайдеры, автономные системы и использование ими протоколов IGP и BGP. Часто нет нужды в обмене информацией о маршрутизации между компанией и провайдером, а необходимые для этого сведения можно ввести вручную.

Администратор сети организации самостоятельно выбирает типы маршрутизаторов для внутреннего использования, как и протокол (протоколы) маршрутизации.

Как же объединяются автономные системы? Способ для этого найден в Интернете уже много лет назад. Как можно догадаться, уникальный номер автономной системы играет в этом основную роль. *Протокол внешнего шлюза* (External Gateway Protocol — EGP) использует такие номера и выполняет всю необходимую работу.

Определение автономных систем и используемых для них номеров было изменено в 1996 г. Провайдерам делегируются полномочия на большие блоки адресов, а далее провайдеры предоставляют своим клиентам подблоки адресов. Трафик к провайдеру можно маршрутизировать с использованием более краткого префикса. Затем провайдер добавляет более длинный префикс для идентификации клиента во внешнем мире.

Для маршрутизации номер автономной системы идентифицирует весь кластер сети, состоящий из сети провайдера и всех сетей его клиентов. Как отмечено в RFC 1930, новое определение для автономной системы такое:

> *Автономной системой является соединенная вместе группа из одного или нескольких префиксов IP для одной или нескольких сетей, имеющих единую и строго определенную политику маршрутизации.*

Многие подключенные к Интернету сети имеют очень простую политику маршрутизации, т.е. один провайдер обеспечивает обмен данными с другими сетями Интернета. Такие сети не имеют отдельного номера автономной системы.

Однако коммерческие организации могут иметь несколько провайдеров или использовать Интернет как недорогое средство для общения с клиентами и поставщиками, или ограничивать коммуникационные возможности своих сайтов. Таким организациям необходим собственный номер автономной системы, который будет использован как индекс при определении и реализации политики маршрутизации.

IANA определила один из блоков IP-адресов для личного (не общедоступного) использования. Для получения личного номера автономной системы можно воспользоваться зарезервированным IANA диапазоном от 64 512 до 65 535.

8.3 Маршрутизация в IP

Датаграмма IP следует по пути, состоящему из участков попаданий, первый из которых формируется при выходе из узла в смежную с ним локальную или региональную сеть. Маршрутизаторы, отстоящие друг от друга на одно попадание, называются *соседями* (neighbor).

В заголовок IP можно поместить заранее определенный список попаданий (маршрутизация от источника). Однако такой способ используется крайне редко (чаще — хакерами, поэтому многие маршрутизаторы конфигурируются на отбрасывание всех датаграмм с маршрутизацией от источника).

Обычно датаграммы маршрутизируются посредством выбора *следующего попадания* для точки назначения в каждом из маршрутизаторов по пути следования.

Маршрутизация по следующему попаданию гибка и надежна. Изменения сетевой топологии обычно проводятся при изменении только в одном или нескольких маршрутизаторах, которые могут информировать друг друга о временных или постоянных изменениях в сети и динамически переключать трафик на альтернативный маршрут.

8.4 Метрики маршрутизации

Для сравнения и выбора лучшего из двух маршрутизаторов используется определенный тип *метрик* (удаленных изменений).

8.4.1 Протоколы вектора расстояния

Самый простой протокол для сравнения маршрутизаторов использует счет попаданий между конечными точками пути. Некоторые улучшенные варианты оценивают *стоимость* или *вес* каждого из участков по пути следования. Например, участок попадания через высокоскоростную локальную сеть имеет вес, равный 1, а участок через низкоскоростной носитель (линия "точка-точка" на 19,2 Кбайт/с) имеет вес 10. Таким образом, путь по скоростному участку предпочтительнее пересылки по низкоскоростной связи. *Протокол RIP* оценивает маршрут по счетчику попаданий.

При вычислении метрики маршрутизации более совершенные протоколы комбинируют характеристики, подобные полосе пропускания, задержке, надежности, текущей загрузке или стоимости оплаты. *Протоколы IGRP* и *EIGRP* используют настраиваемые метрики.

Алгоритмы для принятия решения при маршрутизации, основанные на значениях метрик, называются *векторами расстояния* (distance vector).

8.4.2 Протоколы по состоянию связи

Ранее большое внимание уделялось алгоритмам маршрутизации по *состоянию связи* (link state). Работающие по этому принципу маршрутизаторы создают карту сети и исследуют пути от себя до каждой из точек сети.

Для каждой связи карты формируется метрика стоимости. Общая стоимость для каждого начинающегося от маршрутизатора пути вычисляется как сумма стоимостей каждого участка. Затем можно выбрать наилучший путь для направления трафика.

При изменениях в топологии маршрутизаторы посылают сведения об обновлениях другим маршрутизаторам. После обмена пересчитываются стоимости всех путей. Протоколами по состоянию связи являются *OSPF* и *IS-IS*.

Алгоритмы вычисления состояния связи часто *первым* именуют *кратчайший* путь (Shortest Path First — SPF). Это же название дается компьютерному алгоритму, вычисляющему наиболее короткие пути от одного узла до всех остальных узлов сети.

8.5 Таблицы маршрутизации

При направлении датаграммы в удаленную точку назначения хост или маршрутизатор использует сведения из таблицы маршрутизации. Таблица отражает соответствие между каждой из точек назначения и маршрутизатором следующего попадания на пути к этой точке.

> *Перечисленные в таблице точки назначения могут включать в себя суперсети (бесклассовый блок IP-адресов с единым префиксом), сети, подсети и отдельные системы.*

Точка назначения по умолчанию представляется как 0.0.0.0.

Не существует стандартов на формат таблиц маршрутизации, однако наиболее простая из них должна содержать следующие элементы:

- Адрес сети, подсети или системы назначения
- IP-адрес используемого маршрутизатора следующего попадания
- Сетевой интерфейс для доступа к маршрутизатору следующего попадания
- Маску для точки назначения
- Расстояние до точки назначения (количество попаданий)
- Время в секундах от последнего изменения маршрута

Для сокращения размера таблицы многие или все элементы идентифицируют только суперсети, сети или подсети назначения. Смысл этого в том, что, если известно, как добраться до маршрутизатора сети нужного хоста, а затем до маршрутизатора подсети, то вопрос с маршрутизацией будет решен.

Иногда несколько элементов таблицы содержат полные IP-адреса отдельных систем.

Для изучения работы таблиц маршрутизации рассмотрим два примера.

8.6 Таблица маршрутизации по протоколу RIP

Элементы маршрутизации таблицы 8.1 получены из университетского маршрутизатора, работающего по протоколу RIP. В таблице перечислены точки назначения и перемещающиеся по пути следования к этим точкам маршрутизаторы (на них нужно направить датаграмму при отправке ее в заданную точку назначения). Кроме того, в таблице хранятся метрики (по вектору расстояния), помогающие маршрутизатору выбрать следующее попадание.

Таблица 8.1

Таблица маршрутизации RIP-маршрутизатора

IP-маршрут назначения	Маска IP-маршрута	IP-маршрут следующего попадания	Тип IP-маршрута	Протокол IP-маршрута	Метрика IP-маршрута 1	Метрики IP-маршрутов: 2, 3, 4, 5 (совпадают)	Индекс ЕСЛИ IP-маршрута	Возраст IP-маршрута (секунды)
0.0.0.0	0.0.0.0	128.36.0.2	*	rip	2	–1	1	153,836
128.36.0.0	255.255.255.0	128.36.0.62	**	***	0	–1	1	0
128.36.2.0	255.255.255.0	128.36.0.7	*	rip	1	–1	1	30
128.36.11.0	255.255.255.0	128.36.0.12	*	rip	1	–1	1	13
128.36.12.0	255.255.255.0	128.36.0.21	*	rip	1	–1	1	15
128.36.13.0	255.255.255.0	128.36.0.12	*	rip	1	–1	1	14
128.36.14.0	255.255.255.0	128.36.0.21	*	rip	1	–1	1	16
128.36.15.0	255.255.255.0	128.36.0.21	*	rip	1	–1	1	17
128.36.16.0	255.255.255.0	128.36.0.36	*	rip	12	–1	1	24
128.36.17.0	255.255.255.0	128.36.0.12	*	rip	1	–1	1	16
128.36.19.0	255.255.255.0	128.36.0.10	*	rip	14	–1	1	27
128.36.20.0	255.255.255.0	128.36.0.10	*	rip	1	–1	1	28
128.36.21.0	255.255.255.0	128.36.0.5	*	rip	1	–1	1	5
128.36.22.0	255.255.255.0	128.36.0.5	*	rip	1	–1	1	5
128.36.126.0	255.255.255.0	128.36.0.41	*	rip	1	–1	1	23
130.132.0.0	255.255.0.0	128.36.0.2	*	rip	2	–1	1	25
192.31.2.0	255.255.255.0	128.36.0.1	*	rip	3	–1	1	10
192.31.235.0	255.255.255.0	128.36.0.41	*	rip	1	–1	1	25

* — косвенный
** — прямой
*** — локальный

Таблица маршрутизации содержит элементы для многих различных подсетей сети 128.36.0.0, а также маршруты к сетям 130.132.0.0, 192.31.2.0 и 192.31.235.0 (эти значения извлечены из маршрутизатора приложением *HP Open View for Windows Workgroup Node Manager*). Четыре столбца правой части таблицы не используются в RIP).

8.6.1 Использование маски маршрута

Для поиска совпадения с адресом назначения (например, 128.36.2.25) нужно сравнить 128.36.2.25 с каждым элементом *маршрута назначения* (Route Destination). Элементы *маски маршрута* (Route Mask) указывают, сколько бит из 128.36.2.25 должны совпадать с битами маршрута назначения. Допустим, третья строка таблицы 8.1 имеет маску маршрута 255.255.255.0, означающую, что должны совпадать первые три байта, 128.36.2 (именно так и будет). Более формально можно сказать, что нужно сравнивать маршрут назначения с результатом операции логического умножения адреса назначения и маски маршрута.

Предположим, что совпадение выявлено для двух строк таблицы. Предпочтительный путь будет определять строка с более длинной маской.

8.6.2 Маршрут по умолчанию

Первой строкой в таблице 8.1 стоит маршрут *по умолчанию*. В ней указано, что, не найдя совпадения со строкой таблицы, трафик должен быть направлен на ближайший соседний маршрутизатор с адресом 128.36.0.2.

8.6.3 Использование подсети 0

Администратор данной сети сделал то, что не разрешается стандартами. Он присвоил локальной сети, в которой расположен маршрутизатор, номер подсети 0. Мы уже знаем, что нельзя присваивать 0 в качестве номера подсети. Однако, понимая, что некоторые возможности должны быть у любого доступного номера, разработчики маршрутизаторов позволяют управлять и такими адресами.

8.6.4 Прямые и косвенные назначения

Отметим, что один элемент таблицы указывает на *прямой* (direct) тип локальной сети 128.36.0, что означает непосредственное подключение этой сети к маршрутизатору. Протокол является *локальным* (local), когда маршрут можно изучить, просмотрев конфигурационные параметры самого маршрутизатора.

Оставшиеся элементы перечисляют удаленные подсети и сети, которые достигаются *косвенно* (indirect) при направлении трафика на другие маршрутизаторы. Такие маршруты изучаются средствами протокола RIP.

8.6.5 Метрики маршрутизации

В таблице предусмотрено место для нескольких метрик. RIP использует только одну из них — простой счетчик количества попаданий по пути к точке назначения. Неиспользуемые значения установлены в −1. Отметим, что метрика 0 присвоена подсети 128.36.0, которая подключена непосредственно к маршрутизатору. Многие другие точки назначения доступны за одно попадание. Однако подсеть 128.36.19.0 отстоит от маршрутизатора на 14 попаданий.

Мы рассматривали маршрутизатор модели Shiva *Lanrover*, имеющий множество телефонных номеров для подключения линий к интерфейсу 1.

8.6.6 Возраст маршрута

Столбец *возраста маршрута* (Route Age) отслеживает количество секунд от последнего изменения или проверки каждого из маршрутов. Элементы таблицы, созданные через RIP, будут считаться недействительными по тайм-ауту возраста, если их невозможно реконфигурировать в течение трех минут.

8.7 Таблица маршрутизации IGRP/BGP

Элементы маршрутизации в таблице 8.2 получены из маршрутизатора провайдера Интернета. В ней перечислены назначения и идентифицированы маршрутизаторы для следующего попадания, используемые при доставке датаграмм к каждой точке назначения. Кроме того, здесь содержится информация для помощи маршрутизатору при повторном вычислении участка следующего попадания, когда произойдет изменение топологии сети.

Таблица маршрутизации содержит строки для различных сетей и подсетей (информация из маршрутизатора извлечена через систему управления HP *OpenView*).

Таблица 8.2

Элементы таблицы маршрутизации IGRP и BGP

IP-маршрут назначения	Маска IP-маршрута	IP-маршрут следующего попадания	Тип IP-маршрута	Протокол IP-маршрута	Метрика IP-маршрута					Индекс ЕСЛИ IP-маршрута	Возраст IP-маршрута (секунды)
					1	2	3	4	5		
0.0.0.0	0.0.0.0	130.94.40.250	косвенный	ciscolgrp	10647	1170	21000	0	255	6	12
128.121.50.0	255.255.255.0	128.121.50.50	прямой	локальный	0	–1	–1	–1	–1	1	0
128.121.52.0	255.255.255.0	128.121.50.55	прямой	локальный	0	–1	–1	–1	–1	1	35
128.121.54.0	255.255.255.0	128.121.50.50	прямой	локальный	0	–1	–1	–1	–1	1	0
128.6.0.0	255.255.0.0	130.94.0.49	косвенный	ciscolgrp	12610	1536	61000	2	255	3	11
128.96.0.0	255.255.0.0	130.94.40.250	косвенный	ciscolgrp	14647	1170	61000	2	255	6	16
130.33.0.0	255.255.0.0	130.94.16.2	косвенный	ciscolgrp	8710	1536	22000	1	255	2	18
130.44.0.0	255.255.0.0	130.94.0.49	косвенный	ciscolgrp	16610	1536	101000	4	255	3	37
130.68.0.0	255.255.0.0	130.94.0.49	косвенный	ciscolgrp	12710	1536	62000	3	255	3	39
130.94.1.24	255.255.255.248	130.94.0.49	косвенный	ciscolgrp	82125	128	40000	0	255	3	41
130.94.1.32	255.255.255.248	130.94.0.49	косвенный	ciscolgrp	182571	56	40000	0	255	3	42
130.94.2.8	255.255.255.248	130.94.0.49	косвенный	ciscolgrp	10510	1536	40000	0	255	3	42
130.94.2.16	255.255.255.248	130.94.0.49	косвенный	ciscolgrp	10510	1536	40000	0	255	3	43
130.94.7.0	255.255.255.248	130.94.0.49	косвенный	ciscolgrp	10610	1536	41000	1	255	3	2
130.94.7.8	255.255.255.248	130.94.0.49	косвенный	ciscolgrp	12510	1536	60000	1	255	3	3
44.0.0.0	255.0.0.0	130.94.15.201	косвенный	bgp	0	–1	–1	–1	–1	6	51766
128.3.0.0	255.255.0.0	130.94.40.201	косвенный	bgp	0	–1	–1	–1	–1	6	42049
129.210.0.0	255.255.0.0	130.94.15.201	косвенный	bgp	0	–1	–1	–1	–1	6	586765
13.0.0.0	255.0.0.0	130.94.15.201	косвенный	bgp	0	–1	–1	–1	–1	6	224463

8.7.1 Использование маски маршрута

Для поиска совпадения с назначением 128.121.54.101 нужно применить *маску маршрута* для каждого элемента и сравнить результат с *назначением маршрута* (Route Destination). Применение маски 255.255.255.0 к четвертой строке даст 128.121.54.0, что совпадает с элементом назначения.

IGRP выбирает несколько строк — поскольку может существовать несколько элементов с одинаковым полем назначения и маской. В этом случае используется наилучшая из метрик. Или, если метрики совпадают, IGRP может разделить трафик на два или большее число путей.

8.7.2 Маршрут по умолчанию

Первой строкой в таблице стоит маршрут *по умолчанию*. Если не найдено ни одного совпадения, трафик будет передан на ближайший маршрутизатор с адресом 130.94.40.250.

8.7.3 Прямые и косвенные точки назначения

Три следующие строки имеют *прямой* тип для точки назначения, что означает подсети, подключенные непосредственно к этому маршрутизатору. Их протоколы *локальны*, и маршрутизатор может исследовать эти подсети через конфигурационную информацию, вводимую вручную.

Далее идет несколько строк для удаленных (косвенных) точек назначения, положение которых было определено маршрутизатором посредством лицензированного протокола IGRP компании Cisco.

8.7.4 Малые подсети

Набор точек назначения начинается со строки таблицы, содержащей 130.94.1.24, которая выглядит как адрес хоста. Однако маска маршрута указывает, что все эти элементы являются небольшими подсетями. Они имеют часть адреса для хостов, состоящую только из трех последних бит. Например, двоичное представление 24 — 00011000, и все биты для представления этого числа реально принадлежат части адреса для подсети. Хосты такой подсети будут располагаться в диапазоне адресов от 130.94.1.25 до 130.94.1.30.

8.7.5 Строки для протокола Border Gateway Protocol

Таблица завершается списком удаленных назначений, которые были исследованы с помощью протокола *Border Gateway Protocol*, обеспечившего информацию для маршрутизации между автономными системами и Интернетом.

8.7.6 Метрики маршрутизации

Во второй части таблицы 8.2 видно, что метрика 0 присвоена тем точкам назначения, доступ к которым можно получить в трех непосредственно связанных с маршрутизатором сетях. Как и раньше, значения неиспользуемых метрик равны −1.

Всем пяти метрикам значения присвоены при помощи протокола IGRP компании Cisco. Однако не было попыток обеспечить осмысленные значения для метрик точек назначения Интернета в удаленных автономных системах, которые исследовались через протокол BGP.

Все интерфейсы маршрутизатора пронумерованы, и датаграммы пересылаются через интерфейс, указанный в столбце *Индекс ЕСЛИ*.

8.7.7 Возраст маршрута

Для протокола IGRP столбец *возраста маршрута* (Route Age) означает количество секунд, прошедших со времени последнего изменения или проверки маршрута. Строки таблицы маршрутизации, получаемые через этот протокол, должны время от времени реконфигурироваться. Для протокола BGP возраст маршрута отражает стабильность маршрутов в удаленные точки сети.

8.8 Протоколы обслуживания таблиц маршрутизации

Каким образом маршрутизаторы получают информацию для строк своих таблиц? Как поддерживать корректность строк таблицы маршрутизации? Каким будет лучший способ для выбора маршрутизатора следующего попадания? Все эти вопросы решает протокол маршрутизации, простейший из которых должен:

- Анализировать сетевые датаграммы для определения наилучшего пути. Выбирать следующее попадание для каждого из маршрутов.

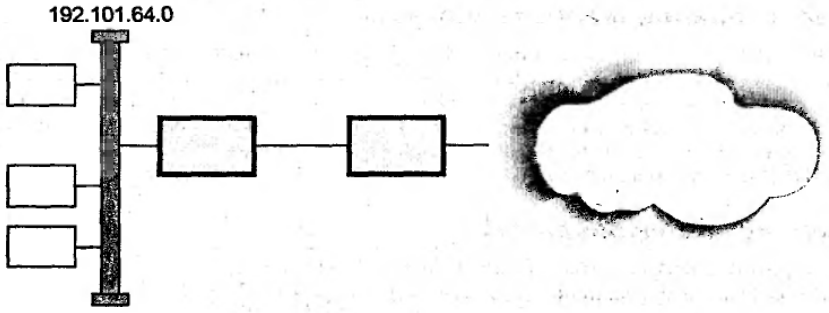

Рис. 8.2. Маршрутизация в подразделении компании

- Обеспечивать ручной ввод данных в таблицу маршрутизации.
- Обеспечивать ручное изменение строк таблицы маршрутизации.

Именно такие операции и выполняет простой маршрутизатор для подразделения компании (см. рис. 8.2). Он может иметь в таблице только две строки — для локальной сети 192.101.64.0 и маршрут по умолчанию для облака (облаками на рисунках принято обозначать сетевые связи через несколько маршрутизаторов.— *Прим. пер.*).

Ручной ввод строк таблицы маршрутизации допустим в небольших сетях, но в сложных, расширяющихся и изменяющихся сетях, имеющих потенциально несколько маршрутов к точке назначения, маршрутизация вручную становится невозможной.

На некотором уровне сложности человек не сможет проанализировать и описать все сетевые условия. Поэтому протокол маршрутизации должен автоматизировать:

- Обмен информацией между маршрутизаторами о текущем состоянии сети
- Повторное вычисление для выбора наилучшего маршрута при каждом изменении в сети

Долгие годы проводились серьезные исследования протоколов маршрутизации. Многие из них были реализованы, а используемые в них метрики породили жаркие дебаты. Приведем характеристики наилучшего протокола:

- Быстрая реакция на изменение в сети
- Вычисление наилучшего маршрута
- Хорошая масштабируемость при расширении сети
- Бережное использование компьютерных ресурсов
- Бережное использование сетевых ресурсов

Однако вычисление наилучшего маршрута в большой сети требует определенных ресурсов центрального процессора и памяти, а быстрая реакция предполагает немедленную пересылку большого объема информации. В хорошем протоколе достигается компромисс между исключающими друг друга требованиями.

Изучение протоколов маршрутизации начинается с наиболее простого из них — RIP.

8.9 Протокол RIP

Наиболее широко используемым протоколом IGP является RIP, заимствованный из протокола маршрутизации сетевой системы компании Xerox (Xerox Network System — XNS). Популярность RIP основана на его простоте и доступности.

RIP был первоначально реализован в TCP/IP операционной системы BSD и продолжает распространяться в операционных системах Unix как программа *routed*.

Программа *routed* стала стандартной частью многих хостов различных разработчиков и пакетов маршрутизации TCP/IP. RIP включен и в бесплатное программное обеспечение Корнельского университета, названное *gated*. RIP получил широкое распространение еще за несколько лет до его стандартизации в документе RFC 1058. Вторая версия протокола была предложена в 1993 г. и улучшена в 1994 г. (после этого исходная версия получила маркировку "историческая", т.е. устаревшая).

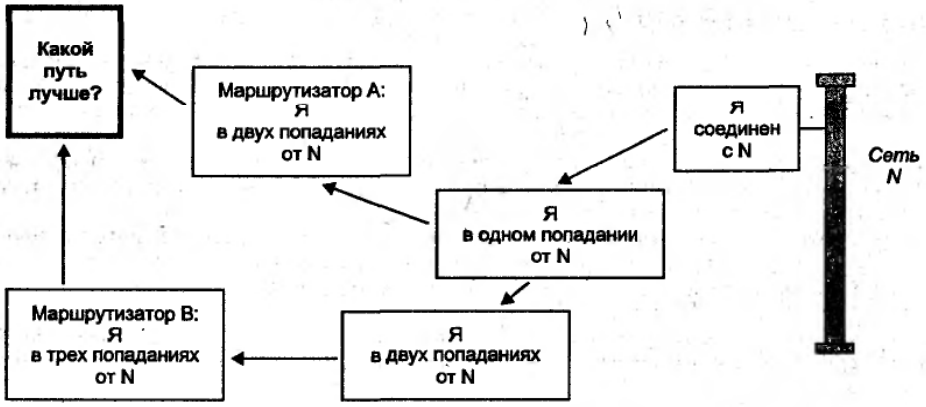

Рис. 8.3. Исследование количества попаданий до точки назначения

RIP анализирует маршрут на основе простого *вектора расстояния*. Каждому попаданию присваивается вес (обычно 1). Общая метрика пути получается как сумма весов всех участков попадания. Выбор лучшего пути для следующего попадания производится по наименьшему значению метрики.

На рис. 8.3 показано распространение в сети процедуры оценки по вектору расстояния. Маршрутизатор из верхнего левого угла рисунка может определить, что датаграмма, направляемая через маршрутизатор A в сеть N, имеет меньше попаданий, чем направляемая в эту сеть через маршрутизатор B.

Для RIP наиболее важны простота и доступность. Часто нет особых причин использовать более совершенные (и более сложные) методы маршрутизации для малых сетей или сетей с простой топологией. Однако при применении в больших и сложных сетях у RIP проявляются серьезные недостатки. Например:

- Максимальное значение метрики для любого пути равно 15.
  Шестнадцать означает *"Точки назначения достичь нельзя!"*. Поскольку в больших сетях можно быстро получить переполнение счетчика попаданий, обычно RIP конфигурируется со значением веса 1 для каждого из участков попадания независимо от того, является этот участок низкоскоростной коммутируемой линией или высокоскоростной волоконно-оптической связью. (Ограничение счетчика позволяет исключить зацикливание датаграмм по круговому маршруту. Другого метода для этого в RIP не существует.— *Прим. пер.*)
- После нарушений в работе сети RIP очень медленно восстанавливает оптимальные маршруты. Реально после нарушения в сети трафик может даже зациклиться по круговому маршруту.
- RIP не реагирует на изменения в задержках или нагрузках линий связи. Он не может распараллеливать трафик для обеспечения баланса нагрузки на связи.

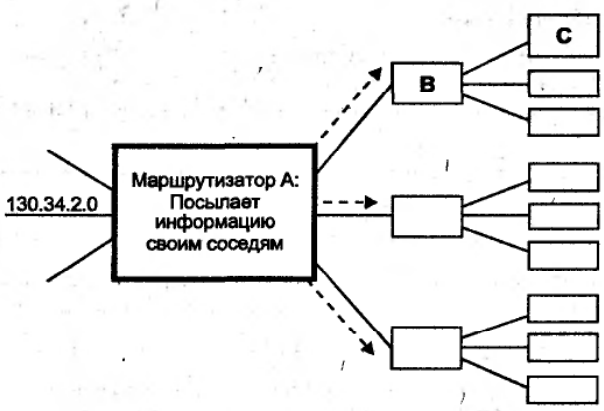

Рис. 8.4. Распространение информации о маршрутизации

8.9.1 Инициализация RIP

При запуске каждый маршрутизатор должен знать только о сети, к которой он подключен. Маршрутизатор RIP отправляет эти сведения широковещательной рассылкой на все соседние с ним в локальной сети маршрутизаторы. Кроме того, эти же сведения посылаются соседям на других концах линий "точка-точка" и виртуальных цепей.

Как показано на рис. 8.4, новости распространяются как сплетни — каждый маршрутизатор пересылает их своему ближайшему соседу. Например, маршрутизатор C очень быстро узнает, что он на расстоянии в два попадания от подсети 130.34.2.0.

Как и все автоматизированные протоколы маршрутизации, RIP посылает информацию об изменениях маршрутов, получает такие сведения от других и пересчитывает пути. Маршрутизатор RIP отсылает информацию своим соседям-маршрутизаторам каждые 30 с. Отправка этих данных называется объявлением о маршруте (advertising route).

Хосты локальной сети могут подслушать объявления в широковещательных рассылках RIP и использовать их для обновления собственных таблиц или, по крайней мере, узнать, что маршрутизатор продолжает работать.

8.9.2 Обновление таблиц RIP

Как видно на рис. 8.5, маршрутизатор A пересылает трафик в сеть 136.10.0.0 через маршрутизатор B. A получил изменения от своего соседа D, который объявил о более коротком маршруте, и A изменил свою таблицу маршрутизации. Отметим, что количество попаданий от A до B добавляется к метрике от D для вычисления расстояния (2) от A до 136.10.0.0.

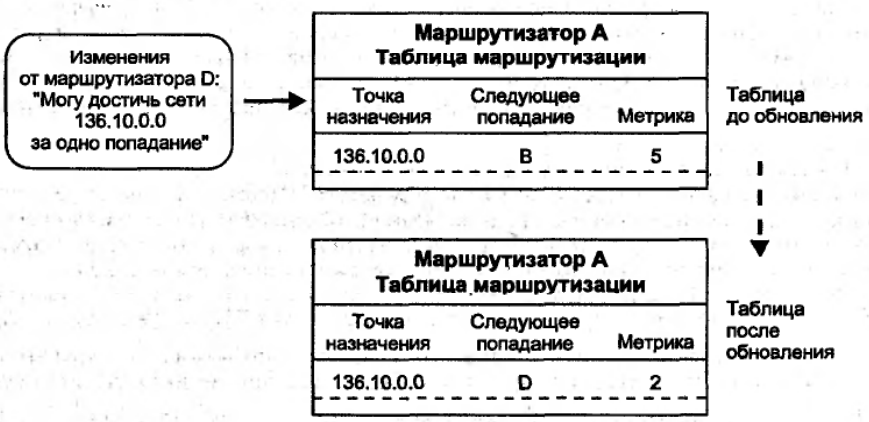

Рис. 8.5. Обновление таблиц маршрутизации в RIP

8.9.3 Механизм RIP версии 1

Рассмотрим формальные этапы маршрутизации в RIP версии 1. Предположим, что в таблице маршрутизации уже есть сведения о нескольких расстояниях. Затем, когда от соседа прибывает информация об изменениях, маршрутизатор перепроверяет свою таблицу и анализирует строки на предмет добавления или улучшения:

1. Присваивается вес для каждой подключенной и пересекаемой при пересылке датаграмм подсети (обычно 1).
2. Маршрутизатор посылает свою таблицу соседям каждые 30 с.
3. Когда маршрутизатор получает таблицу от соседа, он проверяет каждую строку этой таблицы. Присвоенный подсетям вес (в поступивших изменениях) добавляется к каждой из метрик.
4. К локальной таблице маршрутизации добавляются новые точки назначения.
5. Если точка назначения уже присутствует в таблице, но в изменениях указан более короткий путь, то заменяется соответствующая строка локальной таблицы.

Прекрасно, когда маршруты постоянно улучшаются, но иногда, вследствие неисправности связи или маршрутизатора, нужно будет пересылать трафик по более длинному пути. Реакция на неисправности предполагает два пути:

1. Маршрутизатор A пересылал трафик в точку назначения через маршрутизатор X, а X прислал изменения, указывающие на увеличение количества попаданий до точки назначения (или на невозможность достижения точки назначения по данному пути). Маршрутизатор A соответствующим образом изменит строку своей таблицы.
2. Маршрутизатор A пересылал трафик в точку назначения через маршрутизатор X, но не получил изменений от X в течение трех минут. A предполагает неисправность X и маркирует все пути через X как недостижимые (указав для метрики значение 16). Если за 2 мин для таких точек назначения не будет обнаружен новый маршрут, соответствующие строки удаляются (такой процесс образно называют "*сборкой мусора*" — garbage collection). В то же время маршрутизатор A указывает своим соседям через посылаемые изменения, что маршрутизатор X не может обеспечить путь к точкам назначения.

8.9.4 Сообщения об изменениях в RIP версии 1

Как было сказано выше, между маршрутизаторами RIP периодически формируются сообщения об изменениях. Дополнительно можно послать к соседям сообщения с запросами информации о маршрутизации:

- Во время инициализации
- При выполнении операций сетевого мониторинга

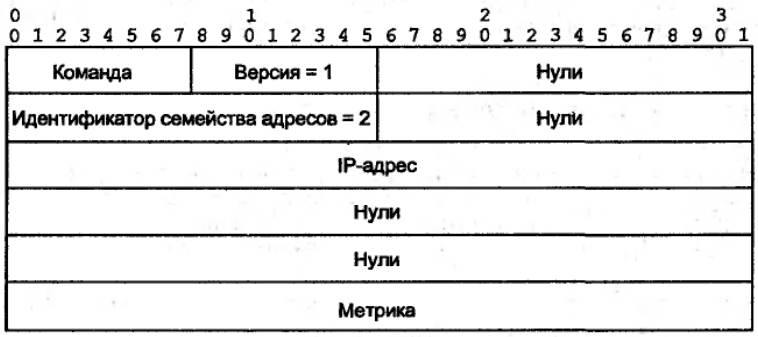

Рис. 8.6. Формат сообщений в RIP версии 1

Формат сообщений RIP версии 1 для запросов или ответов/изменений показан на рис. 8.6. Поле команд со значением 1 указывает на запрос, а идентификатор 2 определяет ответ или самопроизвольное сообщение об изменениях.

8.9.5 Поля сообщения об изменениях в RIP версии 1

Когда создавалась исходная спецификация RFC для RIP, предполагалось, что сообщения о маршрутизации будут использоваться и другими протоколами, а не только IP. Поэтому в сообщении появилось поле *идентификатора семейства адресов* (address family identifier) и место для адреса в 14 октетов.

Семейство адресов, IP-адрес и поле метрики могут повторяться, поэтому сообщение может содержать до 25 адресных элементов. Максимальная длина сообщения составляет 512 октетов. Если нужно переслать сведения о более чем 25 элементах, используется несколько сообщений.

В сообщении об изменениях присутствуют все точки назначения и метрики из таблицы маршрутизации отправителя. В запросе указывают элементы для каждого из адресов, метрику которого нужно получить. Элемент с адресом 0 и метрикой 16 запрашивает полное изменение таблицы маршрутизации.

Регулярные изменения RIP пересылаются через протокол UDP из порта источника 520 в порт 520 маршрутизатора назначения. Однако запросы могут посылаться из любого порта, на который и придет ответ на запрос.

8.9.6 Настройка RIP

Выше мы рассмотрели базовые механизмы протокола RIP. Однако реализации этого протокола имеют некоторые дополнительные возможности для решения следующих проблем:

- При интервале между изменениями, равном 30 с, требуется много времени на распространение изменений по большой сети
- После изменения, особенно при потере соединения, имеется тенденция к зацикливанию трафика по кольцу

Далее рассматриваются пути решения этих проблем.

8.9.7 Триггерные изменения и хранение

Триггерные изменения (triggered updates) ускоряют процесс исследования изменений. Маршрутизатор, изменив метрику пути, посылает объявление о таком изменении.

Отметим, что новое изменение приводит к переключению следующего изменения, и процесс продолжается далее (это напоминает работу триггера). Такой кратковременный поток сообщений позволит множеству пользователей не применять заведомо неисправные пути.

Поскольку формируются причины для одновременной пересылки большого числа изменений, каждая из систем будет ожидать произвольный период времени, прежде чем начать дальнейшую пересылку. Кроме того, полоса пропускания для трансляции таких изменений может быть сокращена за счет пересылки только реально изменившихся строк, а не всей таблицы маршрутизации.

В процессе распространения согласований таблиц маршрутизатор может обнаружить восстановление неисправности и послать сообщение об отмене изменений, поскольку плохой маршрутизатор снова стал хорошим. В таком случае никто не должен менять в своих таблицах хорошую информацию на плохую и вносить изменения, чтобы не распространять далее неверные сведения.

По этой причине многие разработчики реализуют в своих устройствах операцию *хранения* (hold down), когда в течение определенного периода времени игнорируются точки назначения, маркированные как недостижимые.

8.9.8 Деление горизонта и опасный реверс

Почему иногда происходит зацикливание трафика в RIP? Причина в том, что после изменения проходит некоторое время, пока все маршрутизаторы не обновят информацию. На рис. 8.7 показан простой пример (он взят из RFC 1058). Маршрутизатор D имеет два пути к *сети N*. Один из них короткий (в одно попадание), а другой — длинный (в 10 попаданий). Если оборвется связь по короткому пути, маршрутизатор D заменит его на альтернативный (длинный) путь с метрикой 10.

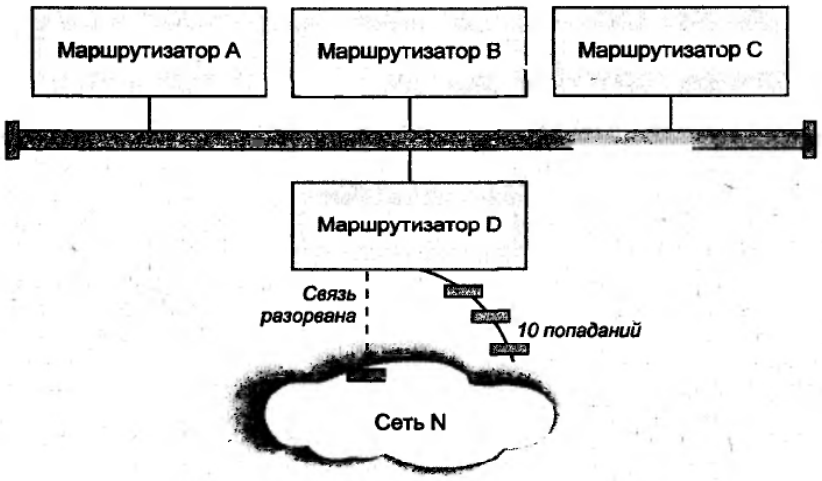

Рис. 8.7. Маршрутизация после неисправности в сети

Однако в сообщениях RIP об изменении, посланных маршрутизаторам A, B и C, будут только следующие сведения:

Сеть N Метрика = 2

Нет никакого способа указать в сообщении, что путь проходит через маршрутизатор D. Что же произойдет, когда маршрутизатор D получит изменения от A до того, как укажет A на собственные изменения?

- D изменит строку своей таблицы на:

Назначение	Следующее попадание	Метрика
Сеть N	A	3

- D попытается переслать трафик к *сети N* через A (последний отправит трафик обратно).
- D отправит объявления об изменении своей таблицы на A, B и C (что он может достичь *сети N* за три попадания).
- Маршрутизаторы ответят, что они теперь смогут попасть в *сеть N* за четыре попадания. Маршрутизаторы B и C столкнутся с неоднозначностью и, в зависимости от времени поступления изменений, могут пытаться отправлять свои трафики к *сети N* друг через друга, через A или D.
- Изменения RIP будут распространяться дальше и глубже.

Хорошо то, что метрики для сети N в A, B и C будут постоянно увеличиваться с приходом каждого нового изменения, пока не достигнут значения 11 и не будет определен правильный маршрут. Два простых механизма позволяют избежать путаницы в сети, которая может возникнуть во время устранения неисправности.

Деление горизонта (split horizon) требует, чтобы маршрутизаторы не посылали своих объявлений о пути к системам со следующим попаданием по этому пути. В примере на рис. 8.7 маршрутизаторы A, B и C не будут указывать D, что могут достичь *сети N*, поскольку путь к N проходит через сам маршрутизатор D.

Опасный реверс (poisoned reverse) идет еще дальше. По этому алгоритму маршрутизаторы A, B и C (см. рис. 8.7) предотвращают распространение неверных сведений с помощью специального сообщения, означающего "Не пытайтесь передавать через меня!". Более точно — изменения будут включать элемент:

Сеть N Метрика = 16

Это исключает проблемы в небольших сетях, но для сетей с большим диаметром колец зацикливания они остаются, даже когда реально нельзя достичь точки назначения. Метрики все равно когда-нибудь увеличатся до 16, и будет восстановлен правильный маршрут. Этот процесс называется *подсчетом до бесконечности* (counting to infinity).

Способы, идентичные рассмотренным выше алгоритмам, можно обнаружить в любом из маршрутизаторов RIP. Однако существует десяток версий RIP, написанных для слишком простых устройств (возможно, для кухонных тостеров). Если нет достоверных данных о способе работы конкретной модели маршрутизатора, его лучше переместить в небольшую сеть и конфигурировать вручную.

Несколько очевидных недостатков сообщений протокола RIP версии 1 мы рассмотрим в следующих разделах.

8.9.9 Нет маски подсети

В сообщения прокола RIP версии 1 не включаются маски (см. рис. 8.6), следовательно, нельзя определить, что представляет собой адрес: подсеть или хост.

Долгое время разработчики маршрутизаторов решали эту проблему, требуя, чтобы пользователи выбирали одну маску подсети для всей сети. Подключенный к сети маршрутизатор определял эту маску с помощью анализа конфигурации интерфейсов сети.

Маршрутизаторы, не подключенные непосредственно к сети, не имели возможности определить маску подсети. Сведения о подсети удаленной сети были для них бесполезны. По этой причине маршрутизаторы RIP версии 1 не посылали сведений о подсетях и хостах на маршрутизаторы, которые не были подключены непосредственно к сети, содержащей эти подсети и хосты. Внешний маршрутизатор посылал единственный элемент для всей сети (например, 145.102.0.0).

Данный способ мог привести как к *избытку* строк в таблице маршрутизации, так и к их недостатку. Если в сети использовалась адресация CIDR, следовало обеспечить отдельные строки для каждого из адресов класса C такой связки. В то время как один элемент с маской подсети мог бы представлять всю сеть CIDR.

8.9.10 Широковещательные рассылки в локальной сети

Версия 1 выполняет широковещательные рассылки в локальной сети. Следовательно, каждый из сетевых интерфейсов должен принимать и анализировать такие сообщения. Больший смысл имеет использование *многоадресных рассылок*.

8.9.11 Отсутствие аутентификации

Еще одним неприятным свойством версии 1 является отсутствие аутентификации для сообщений RIP. Если некто получил доступ к сети и сформировал сообщение с заведомо ложной информацией (фальсифицировав адрес источника), то это может сделать недоступным большинство точек назначения и привести к серьезному нарушению работы сети.

8.9.12 Отсутствие распознавания медленных и быстрых связей

Сетевой администратор может вручную присвоить для связи значение счетчика попаданий. Следовательно, для связи "точка-точка" со скоростью 9,6 Кбайт/с можно установить значение счетчика 5, что укажет на ее меньшие возможности по сравнению со связью 10 Мбайт/с.

К сожалению, когда счетчик достигнет значения 15, точка назначения станет недоступной, а значит, администратору лучше использовать для всех связей одно и то же значение веса, равное 1.

Небольшое максимальное значение счетчика имеет одно преимущество. Вспомним, что недоступная точка назначения иногда приводит к временному зацикливанию пути. Метрики из сообщений об изменениях быстро доведут значение счетчика до 16, и такое кольцо зацикливания будет удалено. Больший предел счетчика привел бы к увеличению времени на уничтожение колец зацикливания.

8.9.13 Избыточный трафик

В больших сетях размер таблиц маршрутизации быстро увеличивается. Пересылка всего содержимого таблицы приведет к существенной дополнительной нагрузке на сеть. Кроме того, замедляется работа маршрутизаторов, которым требуется постоянно анализировать десятки и сотни строк из сообщений об изменениях, большинство из которых вовсе не нужно обновлять.

Небольшой по времени период обновления таблиц приводит к проблемам коммутации на дальние расстояния. Коммутируемые линии или цепи X.25 могут использоваться случайно, создавая отдельные всплески сетевого трафика. Для экономии такие линии и цепи часто закрывают после завершения пересылки данных. По возможности используется ручная конфигурация для связей с удаленными сетями.

Новые протоколы маршрутизации решают такие проблемы с помощью посылки изменений только после их внесения и включают в сообщение только сведения о реально измененных путях. Периодически маршрутизаторы обмениваются сообщениями Hello! (Привет!), позволяющими выяснить работоспособность друг друга, за исключением коммутируемых связей, для которых всегда предполагается нормальное состояние у соседа, пока попытка реальной пересылки данных не завершится неудачей.

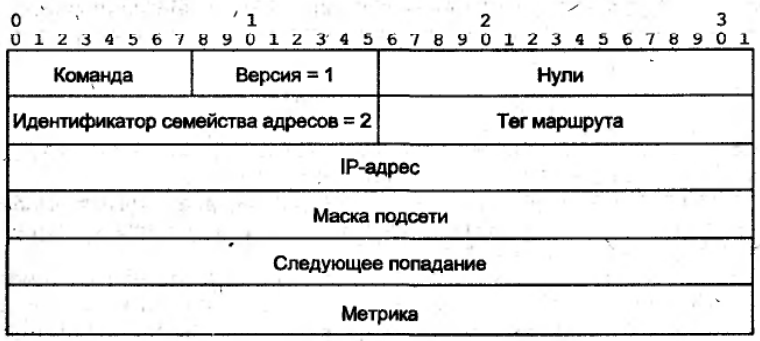

Рис. 8.8. Формат сообщения RIP версии 2

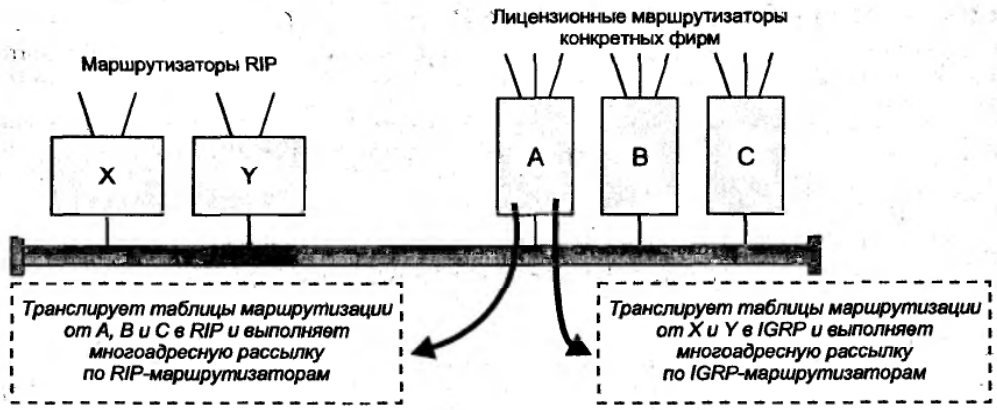

Рис. 8.9. Использование поля *"Следующее попадание"* в отчете маршрутизатора

8.10 Протокол RIP версии 2

Хотя стандарт RFC 1058, в котором была определена версия 1, был опубликован еще в 1983 г., версия 2 протокола RIP появилась только в 1993 г. К этому времени была проведена большая работа по созданию более сложного протокола, способного решить проблемы старой версии. Однако многим организациям нравится простота в инсталляции и использовании RIP старой версии.

Версия 1 была декларирована "исторической", и пользователям нужно было перейти на версию 2. RIP версии 2 предлагает простые решения большинства проблем первой версии. Однако для совместимости с версией 1 изменения были ограничены. Максимальное значение счетчика попаданий осталось равным 15, а все содержимое таблицы маршрутизации по-прежнему обновляется каждые 30 с. Но для передачи изменений стали использоваться *многоадресные*, а не широковещательные рассылки.

Большинство доработок в версии 2 связано с размещением дополнительной информации в сообщении об изменениях. Формат сообщения версии 2 показан на рис. 8.8.

Маска подсети (subnet mask)	Помещена в сообщение
Следующее попадание (next hop)	Используется для отчета маршрутизатора через *другие* маршрутизаторы. Например, "нужно идти к *сети N* через маршрутизатор B". На рис. 8.9 показано, как один многопротокольный маршрутизатор проводит трансляцию между протоколами RIP и IGRP, а также пересылку информации о следующем попадании между двумя наборами маршрутизаторов.
Тег маршрута (route tag)	Это поле содержит информацию для внешнего протокола (например, для BGP). Наиболее популярно использование этого тега для указания номера автономной системы во внешней сети.

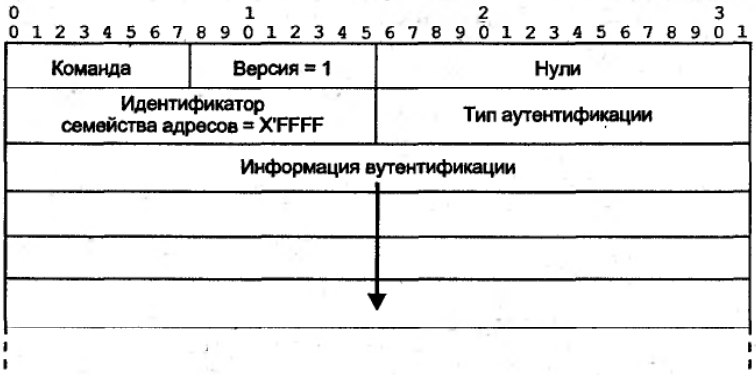

Рис. 8.10. Сообщение версии 2 RIP, начинающееся с аутентификации

8.10.1 Аутентификация в RIP версии 2

Как один из вариантов, место для первого изменения может быть использовано для аутентификации. Оно указывается как поле аутентификации при значении X'FFFF в поле идентификатора семейства адресов. Используемый тип аутентификации описывается в следующем поле.

Оставшиеся 16 бит содержат саму информацию об аутентификации. Хотя для версии 2 определен только один тип аутентификации (с идентификатором 2), использующий простой пароль, разработчики маршрутизаторов понемногу переходят на аутентификацию MD5. На рис. 8.10 показан формат сообщения с аутентификационной информацией.

8.11 Переход на более интеллектуальные протоколы

Для перехода на более интеллектуальные протоколы были сделаны два улучшения. Как и RIP, лицензированный протокол IGRP компании Cisco использует вектор расстояния, однако в нем устранены недостатки RIP. OSPF и IS-IS являются протоколами по состоянию связи. В них создаются карты сети и исследуются все маршруты к точке назначения, а затем полученные метрики путей сравниваются друг с другом.

В этих протоколах поддерживаются дополнительные возможности, например способность разделять трафик по нескольким эквивалентным путям.

Кроме того, произошел переход на поддержку маршрутизации на основе *типов обслуживания* (TOS). Например, один из низкоскоростных маршрутов можно зарезервировать для интерактивного трафика, а путь с большей производительностью (но не слишком малой задержкой) использовать для пересылки больших массивов данных.

8.12 Протоколы IGRP и EIGRP

Хотя IGRP основан на векторе расстояния, его метрики вычисляются по формуле, учитывающей множество факторов, включая полосу пропускания и задержку сети. Дополнительно IGRP учитывает текущий уровень загрузки каждой связи, а также уровень ошибок при пересылке данных из одного конца в другой.

IGRP может разделять трафик по эквивалентным или почти эквивалентным путям. Когда существует несколько путей к точке назначения, большая часть трафика пересылается по пути с большей полосой пропускания.

Граничный маршрутизатор провайдера, использующий протокол IGRP, может собирать сведения от нескольких внешних автономных систем. Следовательно, в этом протоколе поддерживается маршрутизация между различными автономными системами.

EIGRP использует те же метрики и формулы маршрутизации, что и IGRP, но имеет несколько важных улучшений: существенно снижает дополнительный трафик, пересылая сообщения об изменениях только после их внесения в свою таблицу и передает при этом только сведения о реальных изменениях. В EIGRP реализован алгоритм исключения колец зацикливания.

В следующих разделах мы рассмотрим возможности IGRP и улучшения, вносимые EIGRP.

8.12.1 Маршрутизация в IGRP

Как и в RIP, маршрутизатор IGRP периодически распространяет среди соседей содержимое своей таблицы через широковещательные рассылки. Однако в отличие от RIP маршрутизатор IGRP начинает работу с уже сформированной таблицей маршрутизации для подключенных к нему подсетей. Эта таблица расширяется далее благодаря сведениям от ближайших соседей-маршрутизаторов. В сообщениях об изменениях протокола IGRP не содержится сведений о маске подсети. Вместо простого счетчика попаданий RIP применяются различные типы информации о метриках, а именно:

Delay Задержка	Описывает (в десятках мкс) время на достижение точки назначения при отсутствии нагрузки в сети.
Bandwidth Полоса пропускания	Равна 10 000 000, деленным на наименьшую полосу пропускания по заданному маршруту (измеряется в Кбит/с). Например, наименьшая полоса пропускания в 10 Кбит/с соответствует метрике в 1 000 000 Кбит/с.
Load Нагрузка	Измеряется как доля полосы пропускания по заданному маршруту, используемая в текущий момент времени. Кодируется числами от 0 до 255 (255 соответствует нагрузке в 100%).
Reliability Надежность	Часть датаграмм, пришедшая без повреждения. Кодируется числами от 0 до 255 (255 соответствует 100-процентному отсутствию повреждений в датаграммах).

Hop count Счетчик попаданий	Определяет число попаданий до точек назначения.
Path MTU MTU пути	Наибольшее значение Maximum Transmission Unit (MTU) для датаграмм, которые можно переслать по любой связи общего пути.

Значения для задержки, полосы пропускания и MTU берутся из конфигурационной информации маршрутизатора, а значения для нагрузки и надежности вычисляются динамически на основе информации, которой обмениваются маршрутизаторы. В таблице 8.3 дано несколько примеров для кодов задержки и полосы пропускания.

В таблице 8.2 приведены метрики, возвращаемые протоколом *Simple Network Management Protocol* (SNMP) из пула маршрутизаторов Cisco. Например:

IP-маршрут назначения	Метрика IP-маршрута 1	Метрика IP-маршрута 2	Метрика IP-маршрута 3	Метрика IP-маршрута 4	Метрика IP-маршрута 5	Индекс ЕСЛИ IP-маршрута	Возраст IP-маршрута (с)
128.6.0.0	12610	1536	61000	2	255	3	11
128.96.0.0	14647	1170	61000	2	255	6	16
128.112.0.0	10667	1170	21200	1	255	6	23

Для IGRP/EIGRP значения метрик имеют следующий смысл:

Метрика 1	Обобщенная метрика маршрута
Метрика 2	Метрика полосы пропускания
Метрика 3	Сумма задержек интерфейса
Метрика 4	Счетчик попаданий маршрута
Метрика 5	Надежность интерфейса (255 означает 100%)

Таблица 8.3

Измерение задержки и полосы пропускания в IGRP

Носитель	Значение задержки по умолчанию (в десятках мкс)	Метрика полосы пропускания (10 000 000 разделить на полосу пропускания в Кбит/с)
Спутниковая связь (500 Мбит)	200 000 (2 с)	20
Ethernet (10 Мбит)	100 (1 мс)	1 000
1.544 Мбит	2 000 (20 мс)	6 480
64 Кбит	2 000	156 250
56 Кбит	2 000	178 570
10 Кбит	2 000	1 000 000
1 Кбит	2 000	10 000 000

8.12.2 Другие конфигурируемые значения IGRP

Конфигурировать маршрутизаторы IGRP несложно. Кроме IP-адреса, маски подсети, MTU, полосы пропускания и задержки связи, можно специфицировать:

- Фактор изменения (variance factor) V. Если M является наименьшей метрикой пути, используется путь с метрикой M × V.
- Разрешить или запретить хранение (hold down).

Можно конфигурировать и таймеры, хотя чаще используют следующие значения по умолчанию:

- Широковещательная рассылка изменений каждые 90 с.
- Если в течение 270 с не приходит сообщение об изменениях от соседнего маршрутизатора, то соответствующие элементы удаляются по тайм-ауту. Если нет альтернативных маршрутов, точка назначения маркируется как недостижимая.
- Выполняется хранение, во время которого не учитываются новые пути к недостижимой точке назначения (в течение не менее 280 с).
- Если в течение 540 с (время существования потока обновления — flush time), не приходит сведений об изменениях точки назначения, то удаляется соответствующая строка.

8.12.3 Механизм протокола IGRP

Как и в RIP, маршрутизатор IGRP периодически посылает своим соседям сведения об изменениях. К ним относится полное содержимое текущей таблицы маршрутизации со всеми метриками.

Промежуток хранения предотвращает воссоздание разорванного маршрута по сведениям из устаревших сообщений. Ни один новый маршрут к точке назначения не учитывается, пока не завершится период его хранения (хотя можно отключить этот механизм).

Метод расширения горизонтов служит для предотвращения объявления о пути тем маршрутизатором, который расположен ниже по цепочке следования на таком маршруте. Кроме того, IGRP предоставляет собственную версию метода опасного реверса. Если метрика маршрута увеличивается более чем в 1,1 раза, вероятно, будет сформировано зацикливание, и такой маршрут игнорируется.

Триггерные изменения пересылаются только после внесения этих изменений в собственную таблицу маршрутизации (например, при удалении маршрута). Маршрут удаляется в следующих случаях:

- По тайм-ауту коммуникации с ближайшим соседом — удаляется маршрут к этому соседу
- Маршрутизатор следующего попадания указывает на недоступность маршрута
- Метрика увеличивается настолько существенно, что возможно возникновение опасного реверса

8.12.4 Внешняя маршрутизация

Причина популярности IGRP среди провайдеров заключается в возможности управления маршрутизацией между автономными системами. Распространяемые в IGRP изменения включают в себя несколько путей к внешним сетям, из которых можно выбрать один путь для использования по умолчанию.

8.12.5 Возможности EIGRP

Улучшения в EIGRP основаны на тех же метриках и вычислении расстояния, что и обычные свойства этого протокола. Однако расширение свойств существенно улучшает возможности EIGRP за счет поддержки маски подсети и исключения периодических изменений. Пересылаются только реальные изменения, а EIGRP обеспечивает проверку их получения путем анализа обратного сообщения о подтверждении приема. Простые периодические сообщения *Hello!* (Привет!) позволяют узнать об активности своих ближайших соседей. Еще одним важным усовершенствованием стало применение *диффузионного алгоритма для изменений* (Diffusing Update ALgorithm — DUAL), гарантирующего маршрутизацию без зацикливания.

8.12.6 DUAL в EIGRP

Основная идея DUAL проста и основана на следующем наблюдении:

> *Если путь постоянно приближает к точке назначения, то он не может сформировать зацикливание.*

С другой стороны, если путь зациклен (т.е. образует кольцо), он будет содержать маршрутизатор, расстояние которого до точки назначения больше, чем у предшествующего маршрутизатора (см. рис. 8.11).

Метод DUAL разработан для поиска таких путей, на которых каждый маршрутизатор при движении к точке назначения стоит ближе каждого своего предшественника. Маршрутизатор E на рис. 8.11 порождает *серьезные подозрения*, поскольку сведения от ближайшего маршрутизатора, следующего по пути движения (Z), в сообщениях будут иметь большую метрику, чем в собственной таблице E.

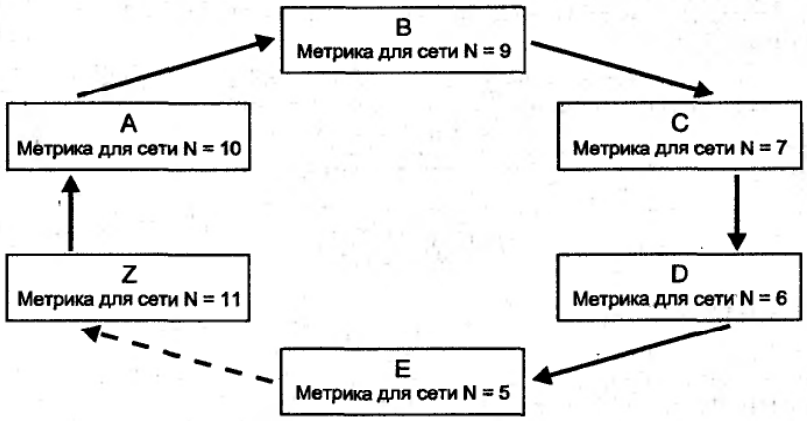

Рис. 8.11. Маршрут с формированием зацикливания

8.12.7 Таблицы топологии в DUAL

Для реализации DUAL протокол EIGRP сохраняет информацию, которой не пользуется IGRP. EIGRP хранит информацию о маршрутах для каждого соседнего маршрутизатора, извлекая ее из сообщений об изменениях от этих маршрутизаторов (IGRP игнорирует любую информацию о неоптимальных маршрутах). Эта информация хранится в дополнительной *таблице топологии* (topology table), содержащей следующие сведения:

Точка назначения	Ближайший сосед	Метрика ближайшего соседа	Собственная текущая метрика

8.12.8 Пригодный преемник в DUAL

Наиболее интересными в таблице топологии являются сведения о *пригодном преемнике* (feasible successor), которым для маршрутизатора является его ближайший сосед, находящийся в текущий момент ближе к точке назначения, чем он сам.

Когда существует, по крайней мере, один пригодный преемник, то можно достичь точки назначения, и для данного пути текущим является *пассивное* (passive) состояние DUAL. Однако когда поступившие изменения меняют картину и пригодный преемник теряется, маршрутизатор начинает опрос ближайших соседей, чтобы определить, нельзя ли переключиться на более длинный маршрут и не будет ли при этом сформировано зацикливание.

Рассмотрим этот процесс с более формальной точки зрения:

1. Предположим, что я могу достичь точки, где будет только один пригодный преемник на пути к точке назначения, через маршрутизатор Z.
2. Поступившие от Z изменения увеличат метрику Z. Более того, новое расстояние от Z до точки назначения *больше*, чем текущее расстояние. Это верный признак формирования зацикливания.
3. Я перехожу в *активное* (active) состояние и начинаю процесс *пересчета маршрута* (route recomputation).
4. Во время пересчета я продолжаю маршрутизировать данные через Z.
5. Я посылаю сообщение об изменениях (называемое query — запрос) всем ближайшим соседям, за исключением Z. В сообщении объявляется о моей новой, большей метрике расстояния до точки назначения.
6. Если сосед имеет один или более пригодных маршрутов, он посылает ответ и объявляет собственный верный путь к точке назначения.
7. Сосед, не имеющий пригодного пути, переходит в *активное* состояние (если только он уже не находится в нем) и посылает запросы *своим* соседям (может немедленно сообщить о том, что он в *активном* состоянии и выполняет пересчет).

8. Запросы распространяются в сети, пока не будут найдены все пригодные маршруты или запрос не дойдет до маршрутизатора, который точно знает, что данная точка назначения недостижима.
9. Когда маршрутизатор определяет для себя пригодный путь или недоступность точки назначения, он отсылает обратно ответ на полученный им запрос.
10. Когда придут ответы на все собственные запросы (не вторичные от других маршрутизаторов.— *Прим. пер.*), маршрутизатор переходит в пассивное состояние.

EIGRP показал, что вектор расстояния еще долго может использоваться при маршрутизации в сетях. В следующих разделах мы рассмотрим альтернативный способ — метод по состоянию связи.

8.13 Протокол OSPF

В 1988 г. комитет IETF начал работу над стандартом нового протокола для замены RIP. В результате была создана спецификация одного из протоколов IGP, призванная *сначала открывать самый короткий путь* (Open Shortest Path First — OSPF). OSPF был разработан как протокол маршрутизации для использования внутри всех автономных систем любых сайтов. В 1990 г. OSPF был рекомендован в качестве стандарта. Это нелицензированный протокол для общедоступного использования.

Вспомним, что протоколы по состоянию связи исследуют пути посредством построения карты сети для формирования дерева пути, корнем которого является маршрутизатор. Метрики вычисляются для каждого пути, а затем оптимальный путь (пути) определяется для каждого типа обслуживания IP (Type Of Service — TOS).

В OSPF используется как метод вектора расстояния, так и состояние связи. Этот протокол разрабатывался для обеспечения хорошей масштабируемости и быстрого распространения по сети сведений о точных маршрутах. Кроме того, в OSPF поддерживается:

- Быстрое определение изменений в топологии и очень эффективное восстановление маршрутов без зацикливания
- Небольшая нагрузка, что связано с распространением в сети только сведений об изменениях, а не обо всех маршрутах
- Разделение трафика между несколькими эквивалентными путями
- Маршрутизация на основе типа обслуживания
- Использование в локальных сетях многоадресных рассылок
- Маски для подсетей и суперсетей
- Аутентификация

В апреле 1990 г., когда очень большая сеть NASA Science (Космического агентства США.— *Прим. пер.*) была переведена на протокол OSPF, обнаружилось существенное снижение трафика в этой сети. После изменения или нарушения в работе сети глобальная корректировка информации о маршрутизации стала выполняться необычайно быстро — в пределах нескольких секунд (по сравнению с минутами для некоторых старых протоколов).

В середине 1991 г. была опубликована вторая версия OSPF, а в марте 1994 г. появилась доработанная вторая версия. Последний вариант описывается в 216-страничном документе, поэтому приведенные ниже сведения можно рассматривать только как общее описание этого протокола.

8.13.1 Автономные системы, области и сети

В стандарте OSPF термин "*сеть*" (network) означает сеть IP, подсеть или суперсеть CIDR. Точно так же *маска сети* (network mask) определяет сеть, подсеть или суперсеть CIDR. *Область* (area) рассматривается как набор непрерывных сетей или хостов вместе со всеми маршрутизаторами, имеющими интерфейсы в этих сетях.

Автономная система, использующая OSPF, создается из одной или нескольких областей. Каждой области присвоен номер. Область 0 представляет собой *магистраль* (backbone), которая соединяет все другие области и объединяет вместе автономные системы. Рассматриваемую топологию иллюстрирует рис. 8.12.



Рис. 8.12. Области и магистрали OSPF

8.13.2 Маршрутизация в области OSPF

Маршрутизация внутри области основана на подробной карте состояний связи в этой области. OSPF хорошо масштабируется, поскольку маршрутизатору нужно подробно знать топологию и метрики *только* об области, которой он принадлежит.

Каждый маршрутизатор OSPF в заданной области хранит идентичную *базу данных маршрутизации* (routing database), описывающую топологию и статус всех элементов этой области. База данных используется для создания карты области и содержит сведения о состоянии каждого маршрутизатора, каждого используемого интерфейса маршрутизатора, подключенной к нему сети и смежных с ним маршрутизаторах.

Как только происходит изменение (например, обрыв связи), информация об этом распространяется по всей сети. Именно этим обеспечивается точность маршрутизации и быстрая реакция на неисправность. Например, если для показанной на рис. 8.13 структуры используется OSPF, то маршрутизатор A будет быстро информирован об обрыве связи с маршрутизатором B и узнает о невозможности доступа к *сети N*.

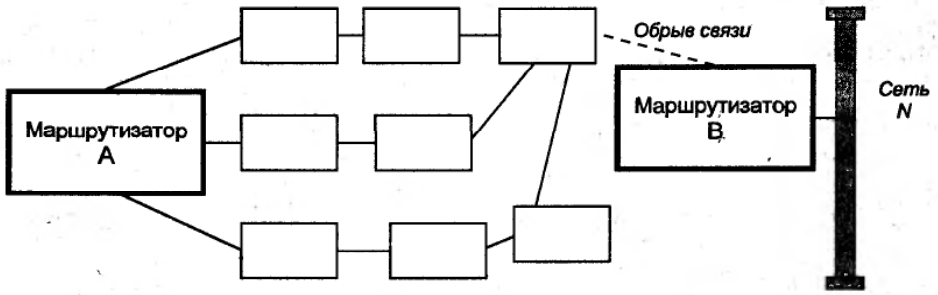

Рис. 8.13. Использование полной информации о маршрутизации

Маршрутизатор инициирует получение копии текущего состояния базы данных от смежного с ним соседа. После этого происходит обмен только изменениями, которые быстро становятся известными в OSPF, поскольку для распространения информации об изменениях по всей области используется потоковый алгоритм.

8.13.3 Кратчайшие пути для области OSPF

Маршрутизатор использует базу данных области для создания дерева кратчайших путей, рассматривая себя как корень этого дерева. На основе дерева формируется таблица маршрутизации. Если в области поддерживается тип обслуживания (TOS), то для значений каждого из типов обслуживания формируются отдельное дерево и набор маршрутов.

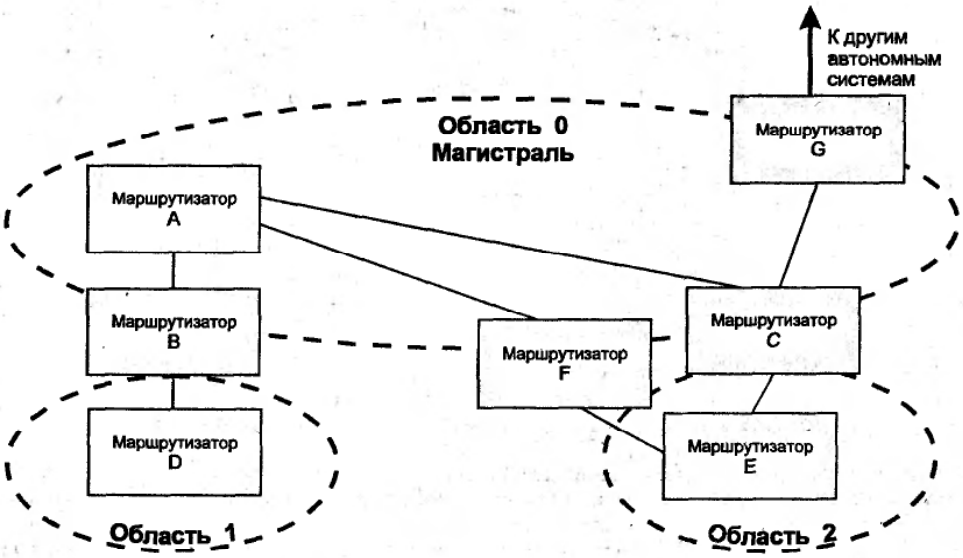

Рис. 8.14. Маршрутизаторы и области в автономных системах

8.13.4 Магистрали, грани и границы OSPF

Области объединяются магистралями. Магистраль содержит все маршрутизаторы, принадлежащие разным областям, а также любые сети и маршрутизаторы, не включенные в другие области. Области нумерованы, а магистраль имеет номер 0.

Маршрутизатор *грани* (border) принадлежит одной или нескольким областям и магистрали. Если автономная система соединена с внешним миром, то маршрутизатор *границы* (boundary) содержит сведения о маршрутизаторах сети, являющейся внешней для автономной системы.

На рис. 8.14 магистраль (область 0) включает маршрутизаторы A, B, C, F и G. К области 1 относятся маршрутизаторы B и D. Область 2 содержит маршрутизаторы C, E и F. Маршрутизаторы B, C и F являются маршрутизаторами грани, а G — маршрутизатором границы. Маршрутизатор B знает все о топологии области 1 и магистрали. Аналогично маршрутизаторы C и F имеют сведения об области 2 и магистрали.

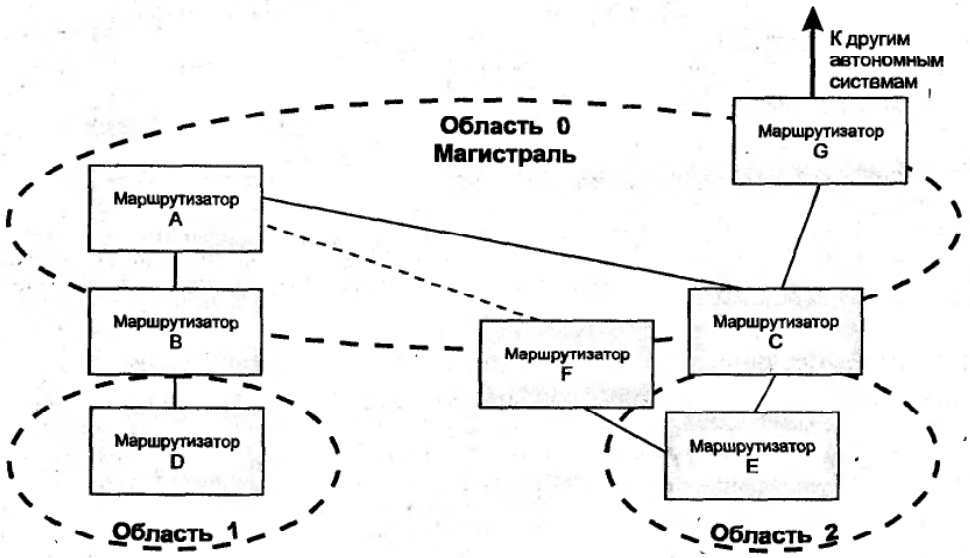

Рис. 8.15. Определение виртуальной связи

Магистраль должна быть непрерывной. Что произойдет при разрыве магистрали из-за расформирования сети или неисправности оборудования? Иногда для объединения отдельных элементов в магистраль используют *виртуальные связи*.

Виртуальную связь (virtual link) можно установить между двумя маршрутизаторами магистрали, имеющими интерфейсы в одной и той же области. Виртуальная связь трактуется как нечисловая связь "точка-точка". Мера стоимости виртуальной связи определяется общей стоимостью пути между двумя маршрутизаторами.

Как показано на рис. 8.15, когда потеряна связь между A и F, маршрутизатор F не будет более соединен с другим маршрутизатором посредством магистральной связи. Для восстановления целостности магистрали придется воспользоваться виртуальной связью F-E-C.

8.13.5 Маршрутизация через грань области OSPF

Маршрутизатор грани имеет все данные о топологии каждой из подключенных к нему областей. Кроме того, он знает и всю топологию магистрали, поскольку подключен к ней непосредственно.

8.13.6 Использование итоговой информации внутри области OSPF

Каждый маршрутизатор грани создает итоговую информацию об области и указывает другим маршрутизаторам магистрали, насколько далеко они расположены относительно сети его области. Это позволяет каждому маршрутизатору грани вычислять расстояние до точки назначения вне его собственной области и пересылать эти сведения внутрь собственной области.

Итоговая информация содержит сведения о сети, подсети или идентификатор суперсети, а также маску сети и расстояние от маршрутизатора до внешней сети.

Например, на рис. 8.16 маршрутизатору E нужно выбрать путь к *сети M*. Маршрутизатор E использует базу данных своей области для поиска расстояния d_c и d_f до маршрутизаторов граней C и F. Каждый из них сообщает сведения о своем расстоянии m_c и m_f до *сети M*. Маршрутизатор E может сравнить $d_c + m_c$ и $d_f + m_f$ и выбрать кратчайший маршрут.

Отметим, что маршрутизатор B может не беспокоиться о пересылке итоговых сведений о расстоянии в область 1. Существует только один путь из этой области и можно использовать единственный элемент, описывающий путь по умолчанию, который применим для всех внешних точек назначения. Если область имеет единственный маршрутизатор грани или если неважно, какой из нескольких маршрутизаторов будет использован, то такая область именуется *тупиковой* (stub), и для доступа из нее к внешней точке назначения должен использоваться один или несколько маршрутизаторов по умолчанию.

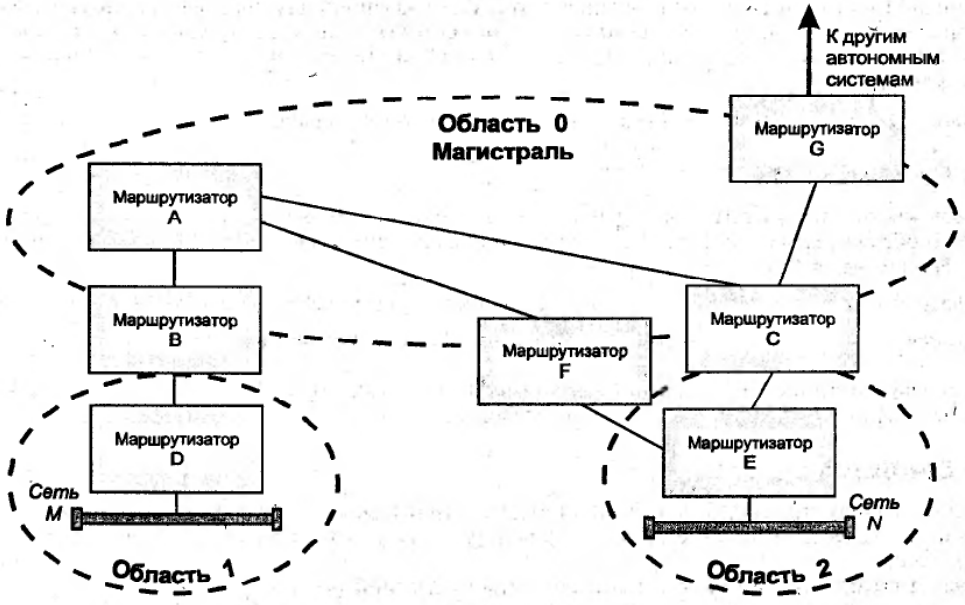

Рис. 8.16. Маршрутизация между областями

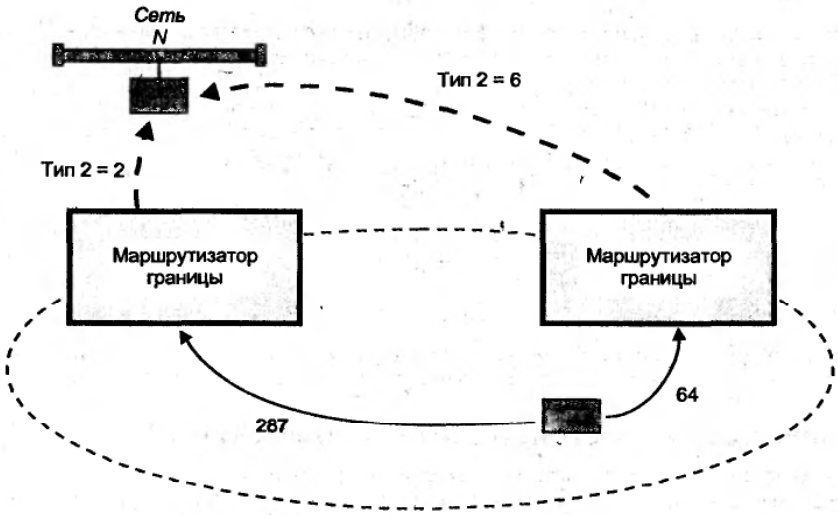

Рис. 8.17. Выбор маршрута по метрике типа 2

8.13.7 Точка назначения вне автономной области OSPF

Многие автономные системы соединены с Интернетом или другими автономными системами. Маршрутизаторы границ (boundary, не путать с гранями.— *Прим. пер.*) предоставляют информацию о расстоянии до сети, расположенной вне автономной системы.

В OSPF существует два типа метрик для внешнего расстояния. Тип 1 эквивалентен метрике состояния локальной связи. Метрика типа 2 служит для длинных расстояний — она измеряет величины в большом диапазоне. Используя аналогию, можно уподобить метрику типа 2 километражу по общенациональной карте автодорог, на которой расстояния измеряются в сотнях км, а метрику типа 1 — километражу по карте отдельной области, где расстояния измеряются в км.

На рис. 8.17 показаны два маршрута к внешней *сети N*. На таком расстоянии игнорируется метрика типа 1, а вычисления производятся по метрике типа 2 (будет выбран маршрут со значением этой метрики, равным 2).

Еще одной возможностью OSPF (специально предназначенной для провайдеров) является возможность маршрутизатора границы автономной системы работать в качестве *сервера маршрутизации* (route server) и предоставлять сведения, идентифицирующие другие маршрутизаторы границ. Такие сведения должны включать:

Точку назначения, Метрику, Используемый маршрутизатор границы

8.13.8 Протокол OSPF

Теперь мы готовы описать некоторые внутренние свойства протокола OSPF. Каждый маршрутизатор OSPF обслуживает подробную базу данных с информацией для создания дерева маршрутизации области. Например, в базе данных отражены:

- Каждый интерфейс маршрутизатора, соединения и связанные с ними метрики
- Каждая сеть с множественным доступом и список всех маршрутизаторов такой сети

Как маршрутизатор получает эту информацию? Он начинает исследование с поиска своих ближайших соседей, используя для этого сообщения Hello.

8.13.9 Сообщения Hello

Каждый маршрутизатор OSPF конфигурируется с уникальным идентификатором, использующимся в сообщениях. Обычно в качестве идентификатора применяют наименьшую часть IP-адреса этого маршрутизатора.

Маршрутизатор периодически отправляет в многоадресной рассылке сообщение Hello! (Привет!) в сети с множественным доступом (например, локальные сети Ethernet, Token-Ring или FDDI), чтобы другие маршрутизаторы смогли узнать о его активности. Это же сообщение посылается на другие

концы подключенных линий "точка-точка" или виртуальных цепей, чтобы партнеры по этим связям смогли узнать о рабочем состоянии маршрутизатора.

Причина эффективности сообщения Hello кроется в передаваемом в нем списке идентификаторов ближайших соседей, от которых отправитель уже получил аналогичные сообщения. Таким способом каждый маршрутизатор узнает, через кого прошло сообщение.

8.13.10 Назначенный маршрутизатор

В сетях с множественным доступом сообщение Hello используется, кроме прочего, для выбора и идентификации *назначенного маршрутизатора* (designated router), который выполняет две задачи:

- Несет ответственность за надежность изменений в базах данных своих смежных соседей в соответствии с последними изменениями в топологии
- Служит источником *объявления о сетевых связях* (network link advertisement), в которых перечисляются все маршрутизаторы, подключенные к сети с множественным доступом

На рис. 8.18 назначенный маршрутизатор A обменивается сведениями с маршрутизаторами B, C и D своей сети, а также с маршрутизатором E, подключенным по связи "точка-точка".

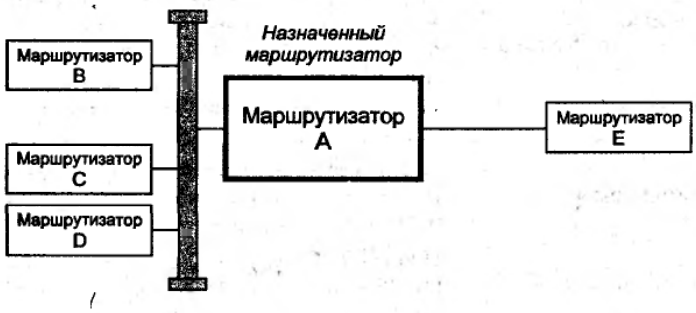

Рис. 8.18. Назначенный маршрутизатор обновляет сведения о сети у своих соседей

8.13.11 Смежность маршрутизаторов

Назначенный маршрутизатор A является главным хранителем текущих сведений о сетевой топологии, предоставляя их для смежных с ним маршрутизаторов.

Маршрутизаторы B, C и D синхронизируют содержимое своих баз данных с маршрутизатором A. Они не обмениваются этими сведениями друг с другом. Два маршрутизатора, которые синхронизируют свои базы данных, называются *смежными* (adjacent). Маршрутизаторы B и C являются *соседями*, но не являются смежными.

Ясно, что назначенный маршрутизатор обеспечивает эффективный метод согласования содержимого баз данных маршрутизаторов локальной сети. Этот же способ применяется в сетях Frame Relay и X.25. Маршрутизаторы могут обмениваться сообщениями Hello по виртуальным цепям, выбирать назначенный маршрутизатор и синхронизовать с ним свои базы данных. Все это позволяет ускорить процесс синхронизации сведений о сети и снизить сетевой трафик.

Потеря назначенного маршрутизатора приведет к серьезному нарушению работы в сети. Поэтому следует всегда выполнять резервное копирование информации из назначенного маршрутизатора и быть готовым к восстановлению этих данных.

8.13.12 Инициализация базы данных маршрутизации

Предположим, что выполняется запуск маршрутизатора B после завершения его профилактического обслуживания с выключением питания. Прежде всего B начинает прослушивать сообщения Hello, исследуя с их помощью своих ближайших соседей и определяя назначенный маршрутизатор (A). Далее B обновляет свои сведения при обмене с A.

Если говорить более строго, то A и B обмениваются сообщениями *Database Description* (описание базы данных). В этих сообщениях находится список содержимого базы данных каждого маршрутизатора. Все элементы таблицы имеют порядковый номер, служащий для определения того, какой из маршрутизаторов содержит более свежие сведения (последовательный номер элемента увеличивается при каждом обновлении этого элемента, и обычно отсчет начинается с нуля).

После завершения обмена каждый маршрутизатор будет знать:

- Какой элемент еще не находится в локальной базе данных
- Какой элемент имеется, но не содержит информации

Сообщения *Link State Request* (запрос о состоянии связи) применяются для элементов, требующих обновления. Сообщение *Link State Update* (изменение состояния связи) приходит в ответ на Link State Request. После полного (и подтвержденного) обмена информацией базы данных считаются синхронизированными. Сообщения Link State Update применяются и для формирования отчета об изменениях в сетевой топологии. С их помощью такие изменения становятся известными по всей сети, и все базы данных синхронизируются.

8.13.13 Типы сообщений в OSPF

Протокол OSPF использует сообщения пяти типов:

Hello "Привет!"	Используется для идентификации соседей, выбора и поиска в сети назначенного маршрутизатора, а также в качестве сигнала "Я присутствую в сети!".
Database Description Описание базы данных	Используется во время инициализации для обмена информацией, чтобы маршрутизатор мог найти сведения, отсутствующие в его базе данных.
Link State Request Запрос состояния связи	Служит для запроса данных, которых маршрутизатор не обнаружил в своей базе данных, либо когда его данные уже устарели.
Link State Update Изменение состояния связи	Применяется в ответ на Link State Request и для динамического обмена сведениями о сетевой топологии.
Link State ACK Подтверждение состояния связи	Используется для подтверждения приема Link State Update. Отправитель повторно отсылает данные, пока не получит это сообщение.

8.13.14 Сообщения OSPF

Сообщения OSPF пересылаются непосредственно в датаграммах IP с типом протокола, равным 89.

Все сообщения OSPF начинаются 24-октетным заголовком (см. рис. 8.19). Номер текущей версии равен 2. *Поле типа* содержит номер соответствующего типа сообщения. Длина определяет общую длину сообщения, включая заголовок.

Тип аутентификации регистрируется через IANA. Безопасность и аутентификация пересылки информации маршрутизации особенно важны для надежности работы сети.

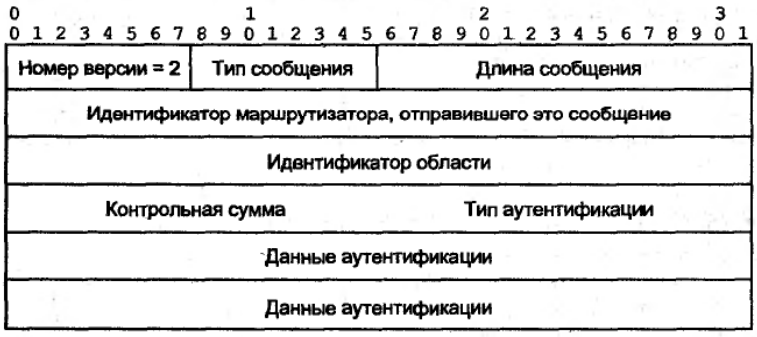

Рис. 8.19. Стандартный 24-октетный заголовок сообщения OSPF

8.13.15 Содержание сообщения Link State Update протокола OSPF

В сообщениях Link State Update пересылается критическая для протокола OSPF информация. Изменения передаются между смежными маршрутизаторами. Назначенный маршрутизатор, получая сообщение об изменениях в сети с широковещательными рассылками, передает их в многоадресных рассылках другим маршрутизаторам этой сети. Изменения распространяются по области необычайно быстро. Прием каждого объявления о новом состоянии связи должен быть подтвержден.

Сообщения Link State Update содержат элементы, называемые *объявлениями* (advertisement). Каждое сообщение может включать следующие типы объявлений:

Router Links Связи маршрутизатора	Состояние каждого из интерфейсов маршрутизатора.
Network Links Сетевые связи	Список маршрутизаторов, подключенных к сети с множественным доступом. Предоставляется назначенным маршрутизатором этой сети.
Summary Link to a Network Список связей с сетью	Маршруты к сети вне локальной области, но внутри автономной системы. Предоставляются маршрутизатором грани.
Summary Link to a Boundary Router Список связей с маршрутизатором границы	Маршруты через автономную систему к ее границе. Предоставляются маршрутизатором грани.
AS External Link Внешние связи автономной системы	Маршруты к точкам назначения в других автономных системах. Предоставляются маршрутизатором границы.

Сообщение Link State Update начинается стандартным 24-октетным заголовком. Оставшаяся часть сообщения содержит объявления о различных типах связей (перечислены выше).

8.13.16 Улучшения в OSPF

Протокол OSPF был значительно улучшен. Например, для снижения стоимости выгодно отключать коммутируемые линии и виртуальные цепи, когда по ним не пересылается трафик. Теперь в протоколе для таких линий формируются периодические сообщения Hello, что позволяет отключать линии, не участвующие в работе. Кроме того, OSPF доработан для поддержки многоадресных рассылок IP. В настоящее время OSPF активно используется, и можно ожидать дальнейших улучшений и пересмотров требований.

8.14 Маршрутизация в OSI

В OSI вместо маршрутизаторов или шлюзов используются *промежуточные системы* (intermediate system). Протокол маршрутизации OSI (IS-IS) был первоначально разработан для OSI, но позднее расширен на IP.

Как и OSPF, IS-IS является протоколом по состоянию связи и поддерживает иерархическую маршрутизацию, типы обслуживания (TOS), разделение трафика по нескольким путям и аутентификацию.

В IS-IS определены: маршрутизаторы двух типов: уровня 1 для маршрутизации внутри области и уровня 2 для точек назначения вне области (последние можно рассматривать как аналоги магистральных маршрутизаторов OSPF). Маршрутизатор уровня 1 для промежуточных систем пересылает трафик, направленный вне границ области, на ближайший маршрутизатор уровня 2. Трафик маршрутизируется далее на маршрутизатор уровня 2 области назначения.

Многие механизмы OSPF основаны на подобных (но не идентичных) механизмах IS-IS, например объявления о состоянии связи, поток сообщений и последовательные номера.

Некоторые сторонники IS-IS считают, что этот протокол лучше IP и для OSI более выгодно применение единого интегрированного протокола, чем отдельных протоколов, для взаимодействия между маршрутизаторами.

8.15 Протоколы EGP

По определению протокол EGP используется внутри автономной системы. Различные автономные системы свободны в выборе конкретного протокола и метрик, наиболее подходящих для каждого конкретного случая. Однако как сделать правильный выбор для маршрутизации трафика между различными автономными системами?

8.16 EGP

Многие годы в Интернете широко использовался простой протокол внешнего шлюза (Exterior Gateway Protocol — EGP) для обеспечения автономных систем маршрутизацией информации во внешнюю сеть. Он характеризуется очень простой структурой. Маршрутизаторы EGP соседних автономных систем обмениваются сведениями о достижимых через них сетях.

EGP был разработан еще в начале 80-х годов, когда Интернет имел очень простую топологию, состоящую из магистрали и набора сетей, непосредственно подключенных к этой магистрали. Когда Интернет достиг современного размера и стал представлять собой топологию в виде сети сетей, EGP используется для пересылки сведений о доступе через цепочки автономных систем (см. рис. 8.20).

EGP не раскрывает, через какие маршрутизаторы будет проходить датаграмма на пути следования к внешней точке назначения. Он скрывает и сведения о пересекаемых на этом пути автономных системах. Простейшие сведения о достижимости, предоставляемые EGP, не соответствуют используемому современному оборудованию. Применение EGP сокращается, поэтому мы рассмотрим его очень кратко.

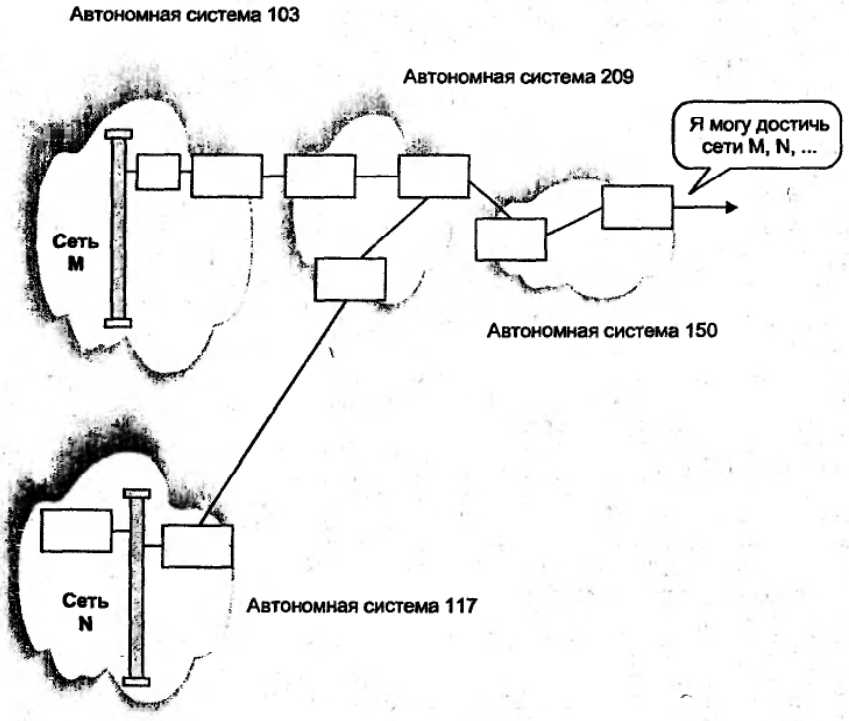

Рис. 8.20. Простое сообщение EGP в сложной сети

8.16.1 Модель EGP

Маршрутизатор EGP конфигурируется с адресом IP для одного или нескольких внешних соседних маршрутизаторов. Обычно внешние соседи соединены с общей сетью с множественным доступом или объединены одной линией "точка-точка".



Рис. 8.21. Сообщения Network Reachability

EGP позволяет маршрутизатору определить, какие из сетей доступны через его внешнего соседа. В EGP используются следующие понятия:

Neighbor Acquisition Обнаружение ближайшего соседа	Маршрутизатор посылает запрос Neighbor Acquisition Request. Получатель запроса возвращает ответ Neighbor Acquisition Response и собственное сообщение Neighbor Acquisition Request.
Neighbor Release Освобождение соседа	Для прекращения связи с соседом маршрутизатор посылает сообщение Neighbor Cease (прекратить связь с соседом), на что получатель отвечает собственным сообщением Neighbor Cease.
Neighbor Reachability Достижимость соседа	Отношения между обнаруженными соседями поддерживаются за счет периодического обмена сообщениями Hello (Привет!) и I Heard You (Я получил ваше сообщение).
Network Reachability Достижимость сети	Маршрутизатор посылает блок своих запросов внешнему соседу, запрашивая информацию о достижимости сетей. Сосед отвечает сообщениями Network Reachability.

Содержание сообщений Network Reachability требует несколько большего обсуждения. Если внешний сосед соединен с линией "точка-точка", то сообщение должно идентифицировать сети, которых можно достичь через отправителя сообщения. Обеспечиваются сведения о счетчике попаданий для каждой точки назначения. На рис. 8.21 показана такая конфигурация — маршрутизатор A отчитывается о достижимости сетей перед маршрутизатором X.

Как показано на рис. 8.22, иногда несколько маршрутизаторов различных автономных систем совместно используют сеть с множественным доступом. В этом случае маршрутизатор A по протоколу EGP будет информировать маршрутизатор X о достижимых через A, B и C сетях, предоставляя для каждой из них значения счетчика попаданий. Точно так же EGP-маршрутизатор X будет информировать маршрутизатор A о сетях, достижимых через X, Y и Z.

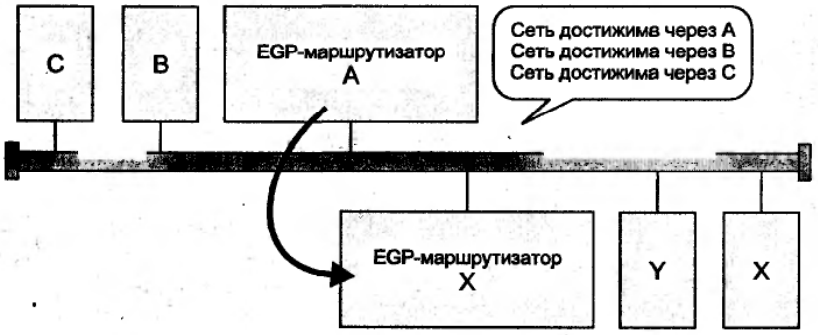

Рис. 8.22. Эффективный обмен информацией EGP

Маршрутизаторы A и X являются *прямыми* соседями (direct neighbor), а B и C — *косвенными* (indirect) для маршрутизатора X.

Если откажет маршрутизатор A, то X должен попытаться использовать одного из своих косвенных соседей (B или C) как прямого соседа для протокола EGP.

Сообщения EGP пересылаются непосредственно в датаграммах IP, имеющих в поле протокола значение 8.

8.17 Протокол BGP

В Интернете широко используется протокол граничного шлюза (Border Gateway Protocol — BGP). Текущей версией протокола является BGP-4.

В современном Интернете существует множество провайдеров, объединенных между собой на манер сети межсоединений. При движении к точке своего назначения трафик часто пересекает сети различных провайдеров. Например, показанный ниже путь начинается в JVNC, пересекает MCI, SPRINT и маршрутизатор NYSERNET, а затем достигает точки своего назначения.

```
> traceroute nyu.edu
traceroute to CMCL2.NYU.EDU          (128.122.128.2), 30 hops max, 40 byte packets
1 nomad-gateway.jvnc.net             (128.121.50.50)   3 ms  3 ms  2 ms
2 liberty-gateway.jvnc.net           (130.94.40.250)  49 ms 10 ms 21 ms
3 border2-hssi2-0.NewYork.mci.net    (204.70.45.9)    13 ms 12 ms 19 ms
4 sprint-nap.NewYork.mci.net         (204.70.45.6)    33 ms 25 ms 19 ms
5 sl-pen-2-F4/0.sprintlink.net       (192.157.69.9)   24 ms 21 ms 21 ms
6 ny-nyc-2-H1/0-T3.nysernet.net      (144.228.62.6)   31 ms 29 ms 24 ms
7 ny-nyc-3-F0/0.nysernet.net         (169.130.10.3)   31 ms 23 ms 20 ms
8 ny-nyu-1-h1/0-T3.nysernet.net      (169.130.13.18)  21 ms 34 ms 19 ms
9 NYU.EDU                            (128.122.128.2)  19 ms 22 ms 21 ms
```

Целью BGP является поддержка маршрутизации через цепочку автономных систем и предотвращение формирования зацикливания. Для этого системы BGP обмениваются информацией о путях к сетям, которых они могут достичь. В отличие от EGP, BGP показывает всю цепочку автономных систем, которые нужно пройти по пути к заданной сети.

Например (см. рис. 8.23), система BGP в автономной системе 34 сообщает автономной системе (АС) 205, что *сети M* и *N* находятся в этой АС. АС 205 отчитывается о пути к *M* и *N* через себя *и* через АС 34. Затем АС 654 указывает на путь к *M* и *N* через себя *и* АС 205 *и* 34. В этом процессе происходит увеличение длины маршрута, но для каждой следующей системы в отчете приводится описание полного пути. Таким образом, информация о доступности в BGP включает полную цепочку автономных систем, которые пересекаются по пути следования к точке назначения.

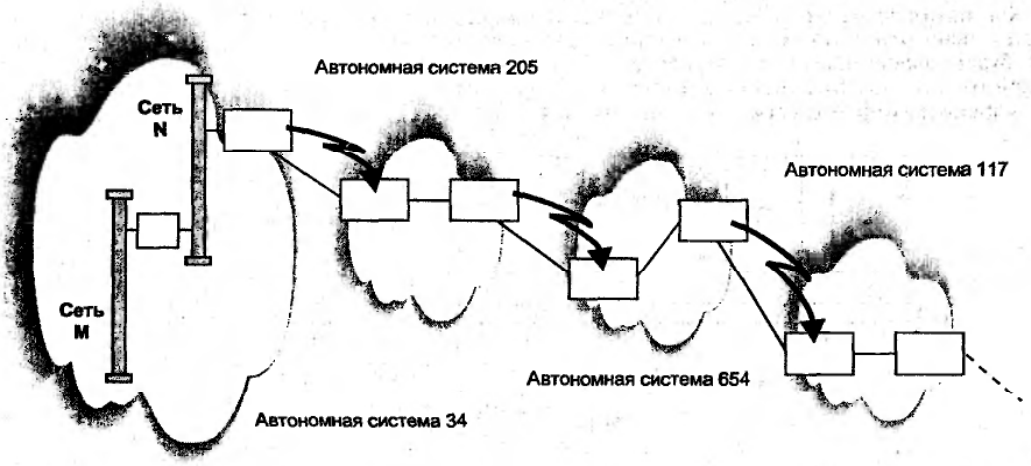

Рис. 8.23. Цепочка BGP из автономных систем

Путь приводится в том порядке, в котором будут пересекаться автономные системы по пути следования к точке назначения:

654, 205, 34

Когда эти сведения будет передавать АС 117, она добавит себя в начало:

117, 654, 205, 34

Отметим, насколько просто выявляются и устраняются кольца зацикливания. Когда АС получает объявление, в котором видит собственный идентификатор, она просто игнорирует такое объявление.

Кроме отчета о маршруте к отдельной сети, BGP способен распознать объединенный набор сетей, используя для этого префикс CIDR.

8.17.1 Объединение маршрутов в BGP

Маршрут в Интернете состоит из сети назначения и инструкций по ее достижению. Наблюдается огромное увеличение числа маршрутов вследствие увеличения числа сетей.

Необходимы меры по управлению маршрутами. Текущим методом сокращения количества маршрутов является присваивание блока адресов с общим префиксом каждому провайдеру, который выделяет из этого блока подблоки своим клиентам.

Длина префикса провайдера определяется числом, указывающим в битах размер префикса в IP-адресе. Трафик может направляться из внешней автономной системы к провайдеру и его клиентам, предполагая использование одного маршрута, соответствующего префиксу. Затем провайдер самостоятельно использует длинный префикс для направления трафика каждой из автономных систем своих клиентов.

Это несложно сделать для входящего трафика, но приходится выполнять обратные действия, когда провайдеру требуется обрабатывать выходящий трафик на основе внешних объявлений. Клиентская автономная система будет информировать провайдера о маршруте к своей внутренней сети. Далее провайдер *объединит* (aggregate) маршруты с общим префиксом в единый элемент описания маршрута, перед тем как об этом маршруте будет объявлено во внешнем мире.

8.17.2 Механизмы BGP

Системы BGP открывают соединение TCP с общеизвестным (well-known) портом 179 соседа по BGP. Каждое сообщение об открытии определяет автономную систему отправителя и имеет идентификатор BGP, а также может содержать дополнительные сведения.

После открытия соединения равные между собой соседи обмениваются информацией о маршрутах. Соединение остается открытым и используется при необходимости для пересылки сведений об изменениях. Для проверки продолжения контакта системы периодически (обычно каждые 30 с) обмениваются сообщениями Keep-alive (продолжаю работать).

Сеть провайдера переносит трафик между автономными системами, и очень неплохо, когда многие системы могут общаться через BGP. Такие системы способны взаимодействовать друг с другом через *внутренние* соединения BGP. *Внешние* соединения BGP используются для коммуникации между равными друг другу системами, находящимися в различных автономных системах (такие соединения называются связями, даже если это *соединения* TCP, которые, возможно, проходят через промежуточные маршрутизаторы).

Существенным отличием BGP от других протоколов маршрутизации является способность обмена информацией о маршрутизации с хостами, а не только с маршрутизаторами. Возможна конфигурация, в которой хост возьмет на себя всю работу по общению с внешними системами BGP в соседних автономных системах. Хост может использоваться как сервер маршрутизации, пересылая информацию граничному серверу собственной автономной системы.

8.17.3 Содержание сообщения об изменениях в BGP

Сообщение об изменениях в BGP может содержать сведения только об одном пригодном маршруте. Однако в нем может присутствовать список из одного или нескольких *изолированных* (withdrawn) маршрутов, которые не следует более использовать.

Описание маршрута состоит из нескольких *атрибутов маршрута*, которые включают:

Origin of Path Information	Источник информации о пути: IGP исходной автономной системы, EGP или иной источник сведений.
AS Path Путь к автономной системе	Маршрут, по которому поступило сообщение об изменениях.

Next Hop Следующее попадание	IP-адрес граничного маршрутизатора, который нужно применять для следующего попадания на пути к точке назначения. Это может быть локальный маршрутизатор автономной системы или внешний маршрутизатор, который напрямую подключен к отправителю и получателю сообщения об изменениях.
Multi-exit Discriminator Многовыходной дискриминатор	Если существует несколько точек выхода для соединения с соседней автономной системой, сосед может присвоить им номера, чтобы указать, какой выход будет лучшим. *Меньший* номер определяет лучший маршрут.
Local Preference Локальное предпочтение	Чистая внешняя информация используется для пересылки изменений BGP элементам локальной автономной системы. Когда есть несколько BGP-маршрутизаторов на пути к точке назначения, более предпочтительный из них имеет *больший* номер.
Atomic Aggregate Атомарное объединение	Указывает, что автономная система объединила несколько точек назначения в единый маршрут.
Aggregator Объединитель	IP-адрес и номер автономной системы для последней из систем, которые объединяли несколько маршрутов в один.
Reachable Nets Достижимые сети	Список префиксов сетей, которых можно достичь через данный маршрутизатор.

8.17.4 Проблема выбора варианта

Рис. 8.24 показывает различия между Multi-exit Discriminator и Local Preference. Системы в АС 117 хотят достичь *сети N* автономной системы (АС) 433. АС 654 имеет два маршрута к точке назначения, и она объявила, что лучший из них — через маршрутизатор E. Однако АС 117 имеет внутренне назначенное локальное предпочтение для доступа к *сети N* через АС 119.

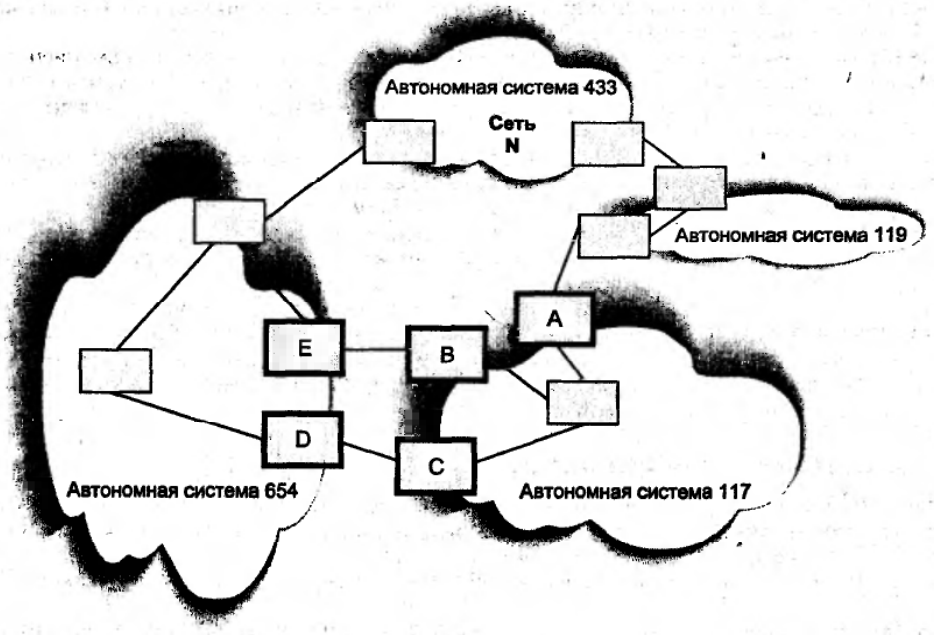

Рис. 8.24. Предпочтительные маршруты

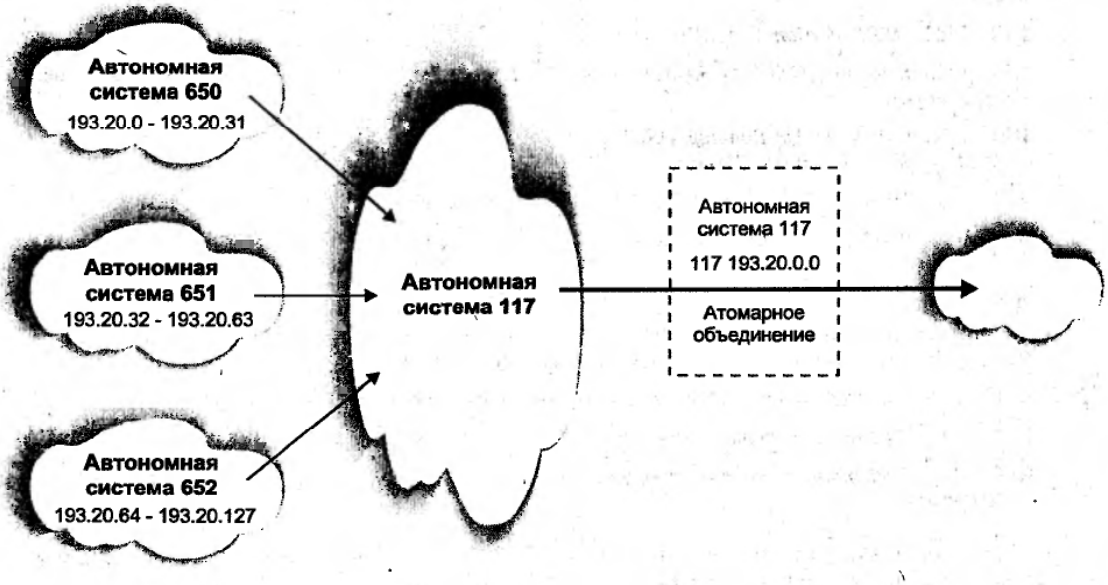

Рис. 8.25. Объединение маршрутов

8.17.5 Применение объединения маршрутов

Целью объединения маршрутов является исключение ненужной информации из удаленных таблиц маршрутизации. Провайдер может объединить маршруты, сведения о которых получены от его клиентской автономной системы.

Как показано на рис. 8.25, маршрутизаторы BGP в автономных системах 650, 651 и 652 могут отчитаться о своих маршрутах, однако провайдер автономной системы 117 объединил их в один маршрут (элемент таблицы маршрутизации). Этот факт отражается атрибутом *Atomic Aggregate*.

Отметим, что автономная система 652 может быть локальным провайдером и объединять маршруты своих клиентов, т.е. от удаленной системы может быть скрыто более одного маршрута. Каждый из объединяющих маршрутизаторов автономной системы будет пересылать трафик к точкам назначения своих клиентов на основе собственной таблицы маршрутизации.

8.17.6 Изолированные маршруты BGP

Маршрут исключается, если:

- Он присутствует в списке изолированных маршрутов из сообщения об изменениях.
- В изменениях приведен заменяющий маршрут.
- Система BGP завершает такое соединение. Все маршруты через эту систему становятся недействительными.

8.18 Дополнительная литература

Маршрутизация настолько важна, что ей посвящены многие RFC. Несколько наиболее существенных и широко используемых документов перечислены ниже. Следует проверить индекс RFC на наличие более поздних версий.

RIP:

RFC 1058 *Routing Information Protocol* (протокол информации о маршрутизации)

RFC 1723 *RIP Version 2 Carrying Additional Information* (RIP, версия 2: перенос дополнительной информации)

RFC 1582 *Extensions to RIP to Support Demand Circuits* (расширение RIP для поддержки цепей по требованию)

OSPF:

RFC 1583 *OSPF Version 2* (OSPF, версия 2)

RFC 1793 *Extending OSPF to Support Demand Circuits* (расширение OSPF для поддержки цепей по требованию)

RFC 1586 *Guidelines for Running OSPF Over Frame Relay Networks* (рекомендации по работе с OSPF через сети Frame Relay)

RFC 1584 *Multicast Extensions to OSPF* (расширение OSPF для многоадресных рассылок)

RFC 1403 *BGP OSPF Interaction* (взаимодействие BGP и OSPF)

BGP
(в будущем предполагается вытеснение BGP протоколом IDRP для OSI — Inter-Domain Routing Protocol, протокол междоменной маршрутизации):

RFC 1771 *A Border Gateway Protocol 4 (BGP-4)* (протокол граничного шлюза, версия 4)

RFC 1773 *Experience with the BGP-4 Protocol* (исследование протокола BGP-4)

RFC 1772 *Application of the Border Gateway Protocol in the Internet* (Приложения для BGP в Интернете)

Кроме того, можно обратиться к интерактивной документации компании Cisco по адресу *www.cisco.com* для получения технических данных о протоколах IGRP и EIGRP.

Глава
9

Протокол UDP

9.1 Введение

После знакомства с физическим перемещением битов в носителе и маршрутизацией датаграмм в Интернете, настало время рассмотреть службы для приложений, связанные с пересылкой данных. Начнем с *протокола пользовательских датаграмм* (User Datagram Protocol — UDP). Это достаточно простой протокол, позволяющий приложениям обмениваться отдельными сообщениями.

Для каких целей используются эти службы? Существует множество приложений, построенных совершенно естественным способом поверх UDP. Так можно, например, реализовать простую систему просмотра базы данных. Кроме того, мы уже упоминали о системе *DNS*, сформированной на основе UDP (см. рис. 9.1).

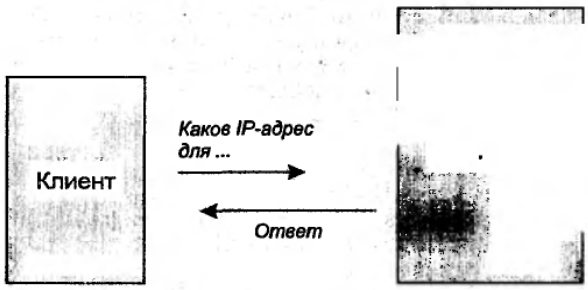

Рис. 9.1. Запрос и ответ DNS

Нагрузки по открытию и закрытию соединений при пересылке большого объема сообщений могут быть исключены благодаря передаче простых запросов и ответов. Кроме того, UDP служит прекрасной основой для конструирования средств мониторинга, отладки, обслуживания или тестирования.

UDP является первичной службой, пересылающей простые отдельные сообщения в IP для последующей передачи по сети. Поскольку IP не обеспечивает надежности пересылки, то нет и гарантий доставки сообщения. Если приложение пытается пересылать свои запросы в датаграммах UDP, но не получает ответов за разумный интервал времени, приложению следует повторно переслать данные.

Иногда это приводит к дублированию запросов на сервере. Если приложение включит в свое сообщение идентификатор транзакции, сервер сможет распознать дублирование и исключить дополнительную копию сообщения. За эти действия ответственно само приложение, а не UDP.

9.1.1 Широковещательные и многоадресные рассылки

Одним из преимуществ UDP является использование этого протокола для широковещательных и многоадресных рассылок из приложений. Например, широковещательная рассылка клиента BOOTP запрашивает инициализационные параметры.

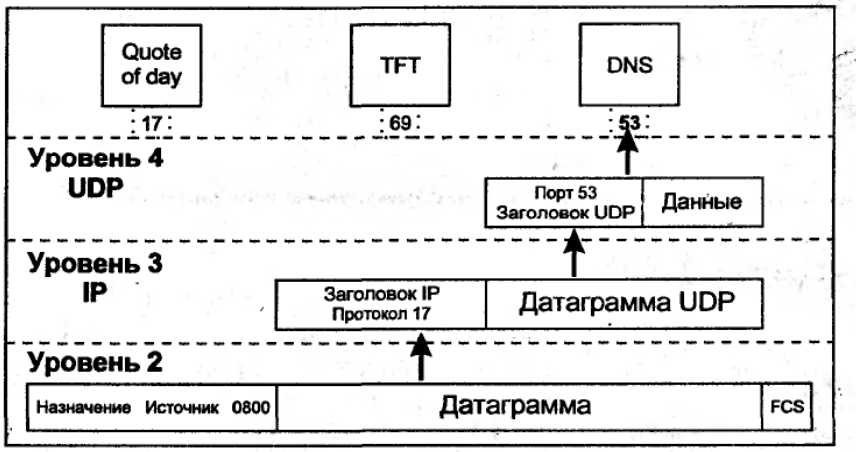

Рис. 9.2. Пересылка данных до уровня приложений

9.2 Порты приложений

Что происходит после прибытия данных в хост назначения? Как выполняется их доставка в нужное приложение (процесс)?

На рис. 9.2 видно, что для каждого уровня существует идентификатор протокола, указывающий операции, выполняемые над данными. На втором уровне тип Ethernet X'08-00 в заголовке кадра показывает, что кадр нужно передать в IP. На третьем уровне поле *протокола* в заголовке IP указывает протокол четвертого уровня, куда нужно переслать данные (например, 6 для TCP или 17 для UDP).

Хост может участвовать одновременно в нескольких коммуникациях. Так как же из общего потока выделяется датаграмма UDP и доставляется на нужный уровень приложения? Такой процесс пересылки данных в требуемый процесс часто называют *демультиплексированием*. Ответ состоит в том, что каждой конечной коммуникационной точке UDP присвоен 16-разрядный идентификатор, называемый номером *порта*. Термин "*порт*" не очень удачен для данного идентификатора. Порт для клиентской и серверной частей приложения *не имеет никакого отношения* к портам оборудования и физическому пути пересылки данных).

Порты с номерами от 0 до 1023 зарезервированы для стандартных служб. Такие порты называются *общеизвестными* (well-known). Их использование позволяет клиенту идентифицировать службу, к которой он хочет получить доступ. Например, доступ к DNS (которая основана на UDP) производится через общеизвестный порт 53.

Кто назначает общеизвестные порты? Как не трудно догадаться, этим занимается IANA. Номера портов для определенных приложений регистрируются этой организацией и публикуются в документе RFC *Assigned Numbers* (присвоенные номера). Сокращенный список портов UDP из текущего документа RFC *Assigned Numbers* показан в таблице 9.1.

Таблица 9.1

Примеры общеизвестных портов UDP

Служба	Порт/протокол	Описание
Echo	7/udp	Посылка отправителю эхо-ответа на пользовательскую датаграмму
Discard	9/udp	Отмена пользовательской датаграммы
Daytime	13/udp	Отчет о времени дня в понятном формате
Quote	17/udp	Возврат сообщения quote of the day — цитата дня
Chargen	19/udp	Генератор символов
Nameserver	53/udp	Сервер имен доменов
Bootps	67/udp	Порт сервера для загрузки конфигурационной информации
Bootpc	68/udp	Порт клиента для получения конфигурационной информации

Таблица 9.1 (*продолжение*)

Примеры общеизвестных портов UDP

Служба	Порт/протокол	Описание
TFTP	69/udp	Порт протокола Trivial File Transfer Protocol
SunRPC	111/udp	Вызов удаленных процедур (Remote Procedure Call) компании Sun
NTP	123/udp	Протокол Network Time Protocol
SNMP	161/udp	Используется для получения сетевых запросов обслуживания
SNMP-trap	162/udp	Служит для получения отчетов о проблемах в сети

Несколько общеизвестных служб обеспечивает модули для тестирования, отладки и измерений. Например, *echo* (эхо) с портом 7, соответствуя своему имени, возвращает любую посланную на этот порт датаграмму. Служба *Discard* (отмена) порта 9, наоборот, удаляет из сети любую посланную на этот порт датаграмму. *Character generator* (генератор символов) отвечает на любое сообщение датаграммой, содержащей от 0 до 512 байт. Количество байт выбирается случайным образом.

Служба *quote of the day* (цитата дня) отвечает на любую датаграмму определенным сообщением, например, в некоторых системах программа *fortune* выводит при регистрации "мудрые" советы (в данном примере приведена фраза Уинстона Черчилля: "Человек может случайно споткнуться об истину, но в большинстве случаев не замечает ее и сосредоточенно продолжает дальнейший поиск".):

```
-> fortune
Churchill's Commentary on Man:
  Man will occasionally stumble over the truth, but most of the
  time he will pick himself up and continue on.
```

Служба *daytime* (время дня) отвечает на любые датаграммы сообщением, содержащим текущую дату и время в формате ASCII. Такой формат можно прочитать на экране без дополнительных преобразований. Иначе ведет себя служба *Network Time Protocol* (NTP), обеспечивающая надежный метод синхронизации компьютеров сети.

Сервер BOOTP и клиент этой службы используются для неконфигурируемых устройств. Рабочая станция может получить для себя IP-адрес, свою маску адреса, узнать местоположение маршрутизатора по умолчанию, адреса наиболее важных серверов сети и, при необходимости, имя и местоположение на сервере boot загружаемого программного файла конфигурации. Программное обеспечение в рабочую станцию поступает через протокол *Trivial File Transfer Protocol* (см. главу 14).

Мы уже знаем, что *сервер имен* доступен через порт 53 и команду *nslookup*. Порты 161 и 162 используются протоколом *Simple Network Management Protocol*.

Кроме официально назначенных номеров, любая система с TCP/IP может резервировать диапазон номеров для важных сетевых служб и приложений.

Оставшиеся номера портов (выше 1023) предоставляются клиентам от программного обеспечения хоста по мере необходимости. Выделение предусматривает следующие шаги:

1. Пользователь запускает клиентскую программу (например, *nslookup*).
2. Клиентский процесс исполняет системную подпрограмму, имеющую смысл: "Я хочу выполнить коммуникацию UDP. Предоставьте мне порт".
3. Системная подпрограмма выбирает неиспользованный порт из пула доступных портов и предоставляет его клиентскому процессу.

Можно видеть, что TCP также идентифицирует источник и назначение своим 16-разрядным идентификатором порта. Например, порт 21 применяется для доступа к службе *пересылки файлов*, а порт 23 — для службы регистрации *telnet*.

Номера TCP и UDP независимы друг от друга. Один процесс может посылать сообщения из порта UDP с номером 1700, в то время как другой продолжает сеанс TCP через порт 1700. Существует несколько служб, доступных как через TCP, так и через UDP. В этом случае IANA предусматривает присвоение одинакового номера порта для службы UDP и TCP. Однако конечные точки коммуникации остаются в разных местах.

9.3 Адреса socket

Используемая для коммуникации комбинация IP-адреса и порта называется *адресом socket* (дословно — гнездо, разъем). Отметим, что адрес socket обеспечивает для сервера или клиента всю информацию, необходимую для идентификации партнера по коммуникации.

Заголовок IP содержит IP-адреса источника и назначения. Заголовки UDP и TCP содержат номера портов источника и назначения. Следовательно, каждое сообщение UDP или TCP несет в себе адрес socket для источника и назначения.

Ниже приведен результат выполнения команды *netstat -na*, выводящей локальные и удаленные адреса socket для текущих активных коммуникаций с системой *tigger*. Адреса socket записаны в форме *IP-адрес.номер_порта*.

```
> netstat -na
Active Internet connections (including servers)
Proto  Recv-Q  Send-Q  Local Address        Foreign Address          (state)
Tcp         0       0  127.0.0.1.1340       127.0.0.1.111    TIME_WAIT
Tcp         0       0  128.121.50.145.25    128.252.223.5.1526       SYN_RCVD
Tcp         0       0  128.121.50.145.25    148.79.160.65.3368       ESTABLISHED
Tcp         0     438  128.121.50.145.23    130.132.57.246.2219      ESTABLISHED
Tcp         0       0  128.121.50.145.25    192.5.5.1.4022   TIME_WAIT
Tcp         0       0  128.121.50.145.25    141.218.1.100.3968       TIME_WAIT
Tcp         0       0  128.121.50.145.25    35.8.2.2.3722    TIME_WAIT
Tcp         0       0  128.121.50.145.1338  165.247.48.4.25          ESTABLISHED
Tcp         0       0  128.121.50.145.25    128.173.4.8.3826         ESTABLISHED
Tcp         0       0  128.121.50.145.25    192.48.96.14.3270        ESTABLISHED
. . .
Udp         0       0  *.7                  *.*
Udp         0       0  *.9                  *.*
Udp         0       0  *.37                 *.*
Udp         0       0  *.19                 *.*
Udp         0       0  *.111                *.*
. . .
```

Например, выделенный рамкой элемент показывает сеанс регистрации TCP из порта клиента 2219 с IP-адресом 130.132.57.246 на стандартный порт *telnet* с номером 23 и адресом 128.121.50.145. Строки, подобные *.7 и *.9, представляют службы UDP на *tigger*, ожидающие запросов от клиентов.

9.4 Механизмы протокола UDP

Какой механизм необходим для запуска протокола User Datagram Protocol? Прежде всего, UDP должен быть присвоен уникальный идентификатор протокола (17). Это значение будет помещаться в поле *протокола* IP с названием Protocol во всех исходящих сообщениях UDP. Входящие сообщения со значением 17 в поле *протокола* IP доставляются в UDP. Протокол UDP формирует сообщение, добавляя простой заголовок к данным от приложения. В этом заголовке указываются номера портов источника и назначения.

9.4.1 Заголовок UDP

На рис. 9.3 представлен формат заголовка UDP. Заголовок содержит 16-разрядные номера портов источника и назначения, определяющие конечные точки коммуникации. Поле длины определяет общее количество октетов в заголовке и части для данных сообщения UDP. Поле контрольной суммы позволяет проверить корректность содержимого сообщения.

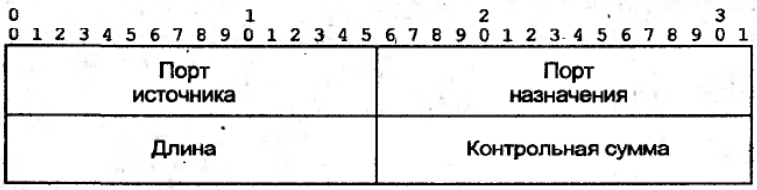

Рис. 9.3. Заголовок UDP

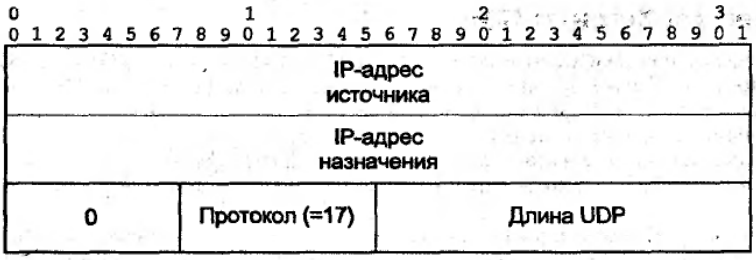

Рис. 9.4. Поля псевдозаголовка для контрольной суммы UDP

9.4.2 Контрольная сумма

Вспомним, что заголовок IP содержит контрольную сумму для проверки корректности своих полей. Назначением контрольной суммы UDP является проверка *содержимого* сообщения UDP.

В UDP контрольная сумма вычисляется как комбинация специально сформированного *псевдозаголовка* (pseudo header), содержащего некоторую информацию IP, заголовка UDP и данных из сообщения.

Формат псевдозаголовка и его участие в вычислении контрольной суммы показаны на рис. 9.4. Отметим, что адрес источника, адрес назначения и поле протокола заимствуются из заголовка IP.

Использование контрольной суммы в коммуникации не является обязательным. Когда она не применяется, поле имеет нулевое значение. Если же контрольная сумма была вычислена и равна нулю, такое значение представляется последовательностью единиц.

9.4.3 Другие функции UDP

Кроме отправки и получения датаграмм, UDP должен руководствоваться здравым смыслом при пересылке данных вниз, от приложения к IP, и обеспечивать указание на ошибки от IP к приложению.

```
IP:  ---- IP Header ----
IP:
IP:  Version = 4, header length = 20 bytes
IP:  Type of service = 00
IP:        000. .... = routine
IP:        ...0 .... = normal delay
IP:        .... 0... = normal throughput
IP:        .... .0.. = normal reliability
IP:  Total length = 131 bytes
IP:  Identification = 21066
IP:  Flags = 0X
IP:  .0.. .... = may fragment
IP:  ..0. .... = last fragment
IP:  Fragment offset = 0 bytes
IP:  Time to live = 60 seconds/hops
IP:  Protocol = 17 (UDP)
IP:  Header checksum = 2A13 (correct)
IP:  Source address = [128.1.1.1]
IP:  Destination address = [128.1.1.10]
IP:  No options
IP:
UDP:  ---- UDP Header ----
UDP:
UDP:  Source port = 1227 (SNMP)
UDP:  Destination port = 161
UDP:  Length = 111
UDP:  No checksum
UDP:
```

```
IP:  ---- IP Header ----
IP:
IP:  Version = 4, header length = 20 bytes
IP:  Type of service = 00
IP:        000. .... = routine
IP:        ...0 .... = normal delay
IP:        .... 0... = normal throughput
IP:        .... .0.. = normal reliability
IP:  Total length = 160 bytes
IP:  Identification = 2015
IP:  Flags = 0X
IP:  .0.. .... = may fragment
IP:  ..0. .... = last fragment
IP:  Fragment offset = 0 bytes
IP:  Time to live = 64 seconds/hops
IP:  Protocol = 17 (UDP)
IP:  Header checksum = 7061 (correct)
IP:  Source address = [128.1.1.10]
IP:  Destination address = [128.1.1.1]
IP:  No options
IP:
UDP:  ---- UDP Header ----
UDP:
UDP:  Source port = 161 (SNMP)
UDP:  Destination port = 1227
UDP:  Length = 140
UDP:  Checksum = 4D4F (correct)
UDP:
```

Рис. 9.5. Заголовки IP и UDP для запроса и ответа

9.4.4 Пример сообщения UDP

Рис. 9.5 содержит совмещенный вывод IP и UDP частей запроса и соответствующих им ответов. Этот результат получен в мониторе локальной сети *Sniffer* компании Network General. Запрос содержал требование вывода статуса информации и был послан хостом на сетевую станцию управления. Часть для данных в сообщениях запроса и ответа не приведена.

Запрос был послан из IP-адреса 128.1.1.1 и порта UDP с номером 1227 на IP-адрес назначения 128.1.1.10 и 161-й порт UDP (запросы сетевого обслуживания всегда направляются на порт UDP с номером 161).

В обоих заголовках IP поле *протокола* имеет значение 17, что указывает на использование протокола UDP. Контрольная сумма UDP не вычисляется для запроса, но присутствует в ответе.

Анализатор *Sniffer* распознает, что порт 161 назначен для сетевого обслуживания.

9.5 Нагрузки в UDP

Когда приложение получает порт UDP, сетевое программное обеспечение протокола резервирует несколько буферов для хранения очереди поступающих на этот порт пользовательских датаграмм. Службы на основе UDP не могут предвидеть количество одновременно поступающих датаграмм и управлять ими.

Если на службу приходит больше датаграмм, чем она может обработать, то дополнительные сообщения просто отбрасываются. Этот факт можно обнаружить с помощью секции UDP Socket Overflows (переполнение в socket протокола UDP) отчета сетевой статистики. Например, приведенный ниже отчет создан командой *netstat*:

```
> netstat -s
. . .
udp:
  0 incomplete headers
  0 bad data length fields
  0 bad checksums
  17 socket overflows
```

9.6 Дополнительная литература

Протокол User Datagram Protocol определен в RFC 768. RFC от 862 до 865 обсуждают UDP-службы *echo*, *discard*, *character generator* и *quote of the day*. RFC 867 описывает утилиту *daytime*, а RFC 1119 представляет вторую версию *службы network time*. Протокол BOOTP рассматривается в главе 11, а о дополнительных службах UDP будет упомянуто в следующих главах.

Глава
10

Протокол TCP

10.1 Введение

Протокол IP слишком прост для того, чтобы в его рамках сконцентрироваться на основной цели этого протокола: маршрутизации данных от источника к назначению. Поэтому работу по обеспечению для трафика датаграмм надежности соединения между приложениями выполняет протокол TCP, который реализуется на каждом из конечных хостов. Поверх протокола TCP реализованы службы WWW, регистрации с терминала, пересылки файлов и обработки электронной почты.

10.1.1 Основные службы TCP

TCP можно рассматривать как средство обеспечения *запросов данных* (data call) по аналогии с обычными телефонными звонками. Вызывающая сторона указывает точку назначения, а на другом конце слушающее приложение реагирует на поступающие вызовы и устанавливает соединение. Производится обмен данными между двумя концами соединения, а по завершении обмена оба партнера говорят "До свидания" и вешают трубки.

IP пытается доставлять датаграммы, прилагая максимальные усилия, однако по пути следования данные могут разрушиться или прибыть в точку назначения в ином порядке, чем были отправлены. Датаграмма может путешествовать по сети достаточно долго и прибывать в произвольные моменты времени. Именно в TCP *обеспечивается надежность, порядок следования и исключаются неисправности и ошибки.*

Приложение быстрого и мощного хоста может перегрузить данными медленного получателя. В TCP реализовано *управление потоком* (flow control), позволяющее *получателю* (receiver) регулировать скорость пересылки данных отправителем. Кроме того, в TCP встроен механизм реакции на текущее состояние сети, подстраивающий поведение протокола для получения оптимальной производительности.

10.1.2 TCP и модель клиент/сервер

TCP естественным образом интегрируется в окружение клиент/сервер (см. рис. 10.1). Серверное приложение *прослушивает* (listen) поступающие запросы на соединение. Например, службы WWW, пересылки файлов или доступа с терминала прослушивают запросы, поступающие от клиентов. Коммуникации в TCP запускаются соответствующими подпрограммами, которые и инициализируют соединение с сервером (см. главу 21 о программном интерфейсе socket).

Реально клиент может быть другим сервером. Например, почтовые серверы могут соединяться с другими почтовыми серверами для пересылки сообщений электронной почты между компьютерами.



Рис. 10.1. Клиент вызывает сервер.

10.2 Концепции TCP

В какой форме приложения должны пересылать данные в TCP? В каком виде TCP передает данные в IP? Каким образом передающий и принимающий протоколы TCP идентифицируют соединение между приложениями и необходимые для его реализации элементы данных? На все эти вопросы даны ответы в следующих разделах, описывающих основные концепции TCP.

10.2.1 Входной и выходной потоки данных

Концептуальная модель соединения предполагает пересылку приложением потока данных равному приложению. В то же время оно способно принимать поток данных от своего партнера по соединению. TCP предоставляет *полнодуплексный* (full duplex) режим работы, при котором одновременно обслуживаются *два потока* данных (см. рис. 10.2).

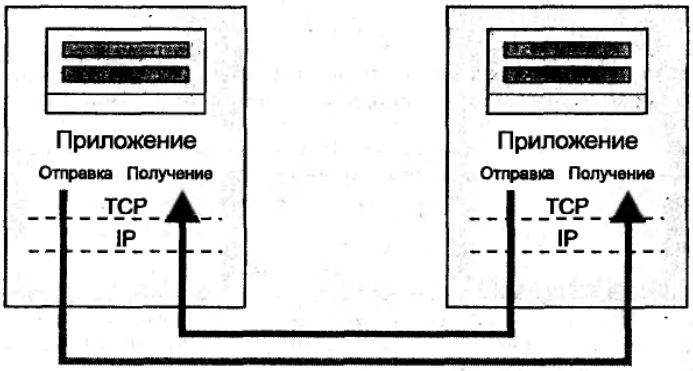

Рис. 10.2. Приложения обмениваются потоками данных.

10.2.2 Сегменты

TCP может преобразовывать выходящий из приложения поток данных в форму, пригодную для размещения в датаграммах. Каким образом?

Приложение передает данные в TCP, а этот протокол помещает их в *выходной буфер* (send buffer). Далее TCP вырезает куски данных из буфера и отправляет их, добавляя заголовок (при этом формируются *сегменты* — segment). На рис. 10.3 показано, как данные из *выходного буфера* TCP пакетируются в сегменты. TCP передает сегмент в IP для доставки в виде отдельной датаграммы. Пакетирование данных в куски правильной длины обеспечивает эффективность их пересылки, поэтому до создания сегмента TCP будет ожидать, пока в выходном буфере не появится соответствующее количество данных.

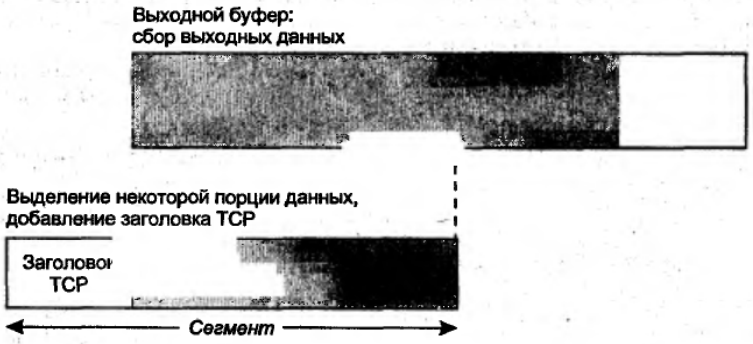

Рис. 10.3 Создание сегмента TCP

10.2.3 Выталкивание

Однако большие объемы данных часто невозможно применить для реальных приложений. Например, когда клиентская программа конечного пользователя инициирует интерактивный сеанс с удаленным сервером, далее пользователь только вводит команды (с последующим нажатием на клавишу *Return*).

Клиентской программе пользователя нужно, чтобы TCP знал о пересылке данных на удаленный хост и выполнил эту операцию немедленно. В этом случае используется *выталкивание* (push).

Если посмотреть на операции в интерактивном сеансе, можно обнаружить много сегментов с небольшим количеством данных, и, более того, выталкивание можно встретить практически в каждом сегменте данных. Однако выталкивание не должно применяться во время пересылки файлов (за исключением самого последнего сегмента), и TCP сможет наиболее эффективно паковать данные в сегменты.

10.2.4 Срочные данные

Модель пересылки данных приложением предполагает применение упорядоченного потока байтов, следующего к точке назначения. Снова обратившись к примеру интерактивного сеанса, предположим, что пользователь нажал клавишу *attention* (внимание) или *break* (прерывание). Удаленное приложение должно быть способно пропустить мешающие байты и отреагировать на нажатие клавиши как можно скорее.

Механизм *срочных данных* (urgent data) маркирует специальную информацию в сегменте как *срочную*. Этим TCP сообщает своему партнеру, что сегмент содержит срочные данные, и может указать, где они находятся. Партнер должен переслать эту информацию в приложение назначения как можно скорее.

10.2.5 Порты приложения

Клиент должен идентифицировать службу, к которой он хочет получить доступ. Это выполняется через спецификацию IP-адреса службы хоста и его номера порта TCP. Как и для UDP, номера портов TCP находятся в диапазоне от 0 до 65 535. Порты в диапазоне от 0 до 1023 называются общеизвестными (well-known) и используются для доступа к стандартным службам.

Несколько примеров общеизвестных портов и соответствующих им приложений показано в таблице 10.1. Службы *Discard* (порт 9) и *chargen* (порт 19) являются TCP-версиями уже известных нам по UDP служб. Нужно помнить, что трафик на порт 9 протокола TCP полностью изолирован от трафика на порт 9 протокола UDP.

Таблица 10.1

Общеизвестные порты TCP и соответствующие им приложения

Порт	Приложение	Описание
9	Discard	Отмена всех входящих данных
19	Chargen	Генератор символов. Обмен потоком символов
20	FTP-Data	Порт пересылки данных FTP
21	FTP	Порт для диалога FTP
23	TELNET	Порт для удаленной регистрации по *Telnet*
25	SMTP	Порт протокола SMTP
110	POP3	Служба выборки почтовых сообщений для персональных компьютеров
119	NNTP	Доступ к сетевым новостям

Что можно сказать о портах, используемых клиентами? В редких случаях клиент работает не через общеизвестный порт. Но в таких ситуациях, желая открыть соединение, он часто запрашивает у операционной системы присвоения ему неиспользуемого и незарезервированного порта. В конце соединения клиент обязан возвратить этот порт обратно, после чего порт может быть использован повторно другим клиентом. Поскольку в пуле нерезервированных номеров существует более 63 000 портов TCP, ограничения на порты для клиентов можно не учитывать.



Рис. 10.4. Адреса socket

10.2.6 Адреса socket

Как мы уже знаем, комбинация IP-адреса и порта для коммуникации называется *адресом socket*. Соединение TCP полностью идентифицируется адресом socket на каждом конце данного соединения. На рис. 10.4 показано соединение между клиентом с адресом socket (128.36.1.24, порт = 3358) и сервером с адресом socket (130.42.88.22, порт = 21).

Заголовок каждой датаграммы содержит IP-адреса источника и назначения. В дальнейшем будет видно, что номера портов источника и назначения указываются в заголовке сегмента TCP.

Обычно сервер способен одновременно управлять несколькими клиентами. Уникальные адреса socket сервера присваиваются одновременно всем его клиентам (см. рис. 10.5).

Поскольку датаграмма содержит сегмент соединения TCP, идентифицирующийся IP-адресами и портами, серверу очень просто отслеживать несколько соединений с клиентами.

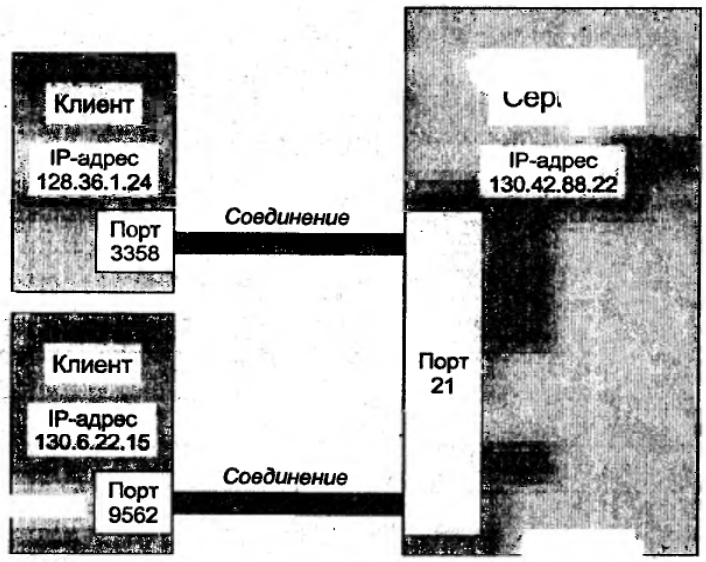

Рис. 10.5. Несколько клиентов соединены с адресами socket сервера

10.3 Механизм обеспечения надежности TCP

В этом разделе мы рассмотрим механизм TCP, используемый для надежной доставки данных при сохранении порядка пересылки и исключения потерь либо дублирования.

10.3.1 Нумерация и подтверждение

Для обеспечения надежной пересылки данных в TCP используются нумерация (numbering) и подтверждение (acknowledgment — ACK). Схема нумерации TCP несколько необычна: *каждый* пересылаемый по соединению *октет* рассматривается как имеющий порядковый номер. Заголовок сегмента TCP содержит порядковый номер *первого октета данных этого сегмента.*

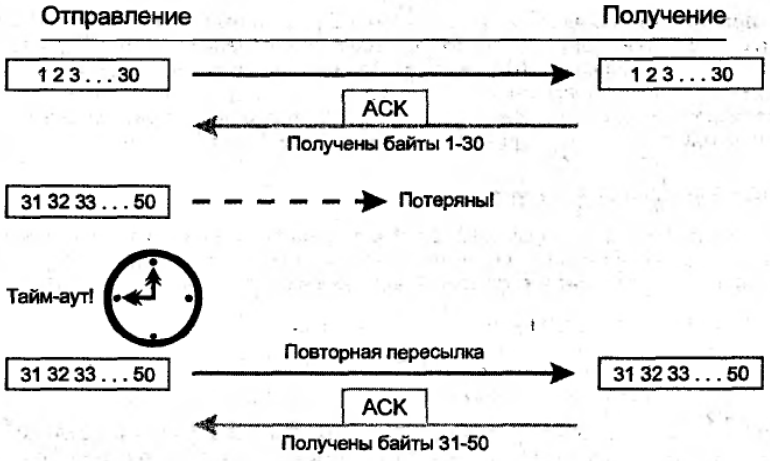

Рис. 10.6. Тайм-аут и повторная пересылка в TCP

От приемника требуется подтверждение получения данных. Если ACK не приходит за интервал тайм-аута, данные передаются повторно. Этот способ называется *позитивным подтверждением с ретрансляцией* (positive acknowledgment with retransmission).

Получатель данных TCP проводит строгий контроль входящих порядковых номеров, чтобы проверить последовательность получения данных и отсутствие потерянных частей. Поскольку ACK случайным образом может быть потерян или задержан, к получателю могут поступить дублированные сегменты. Порядковые номера позволяют определить дублирование данных, которые далее отбрасываются.

На рис. 10.6 показан упрощенный взгляд на тайм-аут и повторную пересылку в TCP.

10.3.2 Поля портов, последовательности и ACK в заголовке TCP

Как показано на рис. 10.7, первые несколько полей заголовка TCP предоставляют место для значений портов источника и назначения, порядкового номера первого байта вложенных данных и ACK, равного порядковому номеру *следующего* байта, ожидаемого на другом конце. Другими словами, если TCP от своего партнера получит все байты до 30-го, в этом поле будет значение 31, указывающее сегмент, который следует переслать далее.

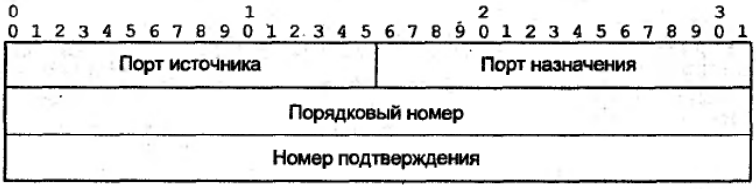

Рис. 10.7. Начальные значения в полях заголовка TCP

Нельзя не отметить одну маленькую деталь. Предположим, что TCP переслал байты от 1 до 50 и более уже нет данных для отправки. Если от партнера поступают данные, TCP обязан подтвердить их получение, для чего пошлет заголовок без подключенных к нему данных. Естественно, в этом заголовке присутствует значение ACK. В поле последовательности — значение 51, т.е. номер следующего байта, который *намеревается* послать TCP. Когда TCP пошлет следующие данные, новый заголовок TCP также будет иметь в поле последовательности значение 51.

10.4 Установка соединения

Каким образом два приложения соединяются между собой? Перед коммуникацией каждое из них вызывает подпрограмму для формирования блока памяти, который будет использован для хранения параметров TCP и IP данного соединения, например адресов socket, текущего порядкового номера, начального значения времени жизни и т.д.

Серверное приложение ожидает появления клиента, который, желая получить доступ к серверу, выдает запрос на *соединение* (connect), идентифицирующий IP-адрес и порт сервера.

Существует одна техническая особенность. Каждая сторона начинает нумерацию каждого байта не с единицы, а со *случайного порядкового номера* (далее мы узнаем, для чего это делается). Исходная спецификация дает совет: начальный порядковый номер генерировать на основе 32-разрядного внешнего таймера, увеличивающего значения примерно каждые 4 мкс.

10.4.1 Сценарий соединения

Процедуру соединения часто называют тройным рукопожатием (three-way handshake), поскольку для установки соединения производится обмен тремя сообщениями — SYN, SYN и ACK.

Во время установки соединения партнеры обмениваются тремя важными порциями информации:

1. Объем буферного пространства для приема данных
2. Максимальное количество данных, переносимое во входящем сегменте
3. Начальный порядковый номер, используемый для исходящих данных

Отметим, что каждая из сторон применяет операции 1 и 2 для указания *пределов, в которых будет действовать другая сторона.* Персональный компьютер может иметь небольшой приемный буфер, а суперкомпьютер — огромный буфер. Структура памяти персонального компьютера может ограничивать поступающие порции данных 1 Кбайт, а суперкомпьютер управляется с большими сегментами.

> *Способность управлять тем, как посылает данные другая сторона, является важным свойством, обеспечивающим масштабируемость TCP/IP.*

На рис. 10.8 показан пример сценария соединения. Представлены очень простые начальные порядковые номера, чтобы не перегружать рисунок. Отметим, что на данном рисунке клиент способен получать большие по размеру сегменты, чем сервер.

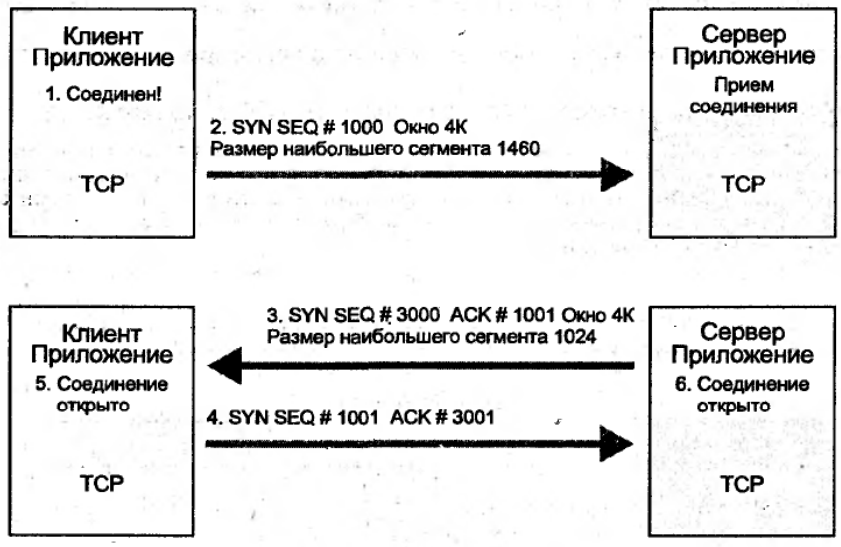

Рис. 10.8. Установление соединения

Выполняются следующие операции:

1. Сервер инициализируется и становится готовым к соединению с клиентами (это состояние называется пассивным открытием — passive open).
2. Клиент запрашивает у TCP открытие соединения с сервером по указанному IP-адресу и порту (это состояние называется активным открытием — active open).
3. Клиентская TCP получает начальный порядковый номер (в данном примере — 1000) и посылает *сегмент синхронизации* (synchronize segment — SYN). В этом сегменте пересылается порядковый номер, размер приемного окна (4 K) и размер наибольшего сегмента, который может принять клиент (1460 байт).

4. Когда поступает SYN, серверная TCP получает *свой* начальный порядковый номер (3000). Она посылает сегмент SYN, содержащий начальный порядковый номер (3000), ACK 1001 (что означает нумерацию первого посланного клиентом байта как 1001), размер приемного окна (4 K) и размер наибольшего сегмента, который сможет получить сервер (1024 байта).
5. Клиентская TCP, получив от сервера сообщение SYN/ACK, отсылает обратно ACK 3001 (первый байт посланных сервером данных должен нумероваться как 3001).
6. Клиентская TCP указывает своему приложению на открытие соединения.
7. Серверная TCP, получив от клиентской TCP сообщение ACK, информирует свое приложение об открытии соединения.

Клиент и сервер анонсируют свои правила для принимаемых данных, синхронизируют свои порядковые номера и становятся готовыми к обмену данными. Спецификация TCP разрешает и другой сценарий (не слишком удачный), когда равные между собой приложения одновременно выполняют активное открытие друг друга.

10.4.2 Установка значений параметров IP

Запрос приложения на установку соединения может заодно указать параметры для датаграмм IP, которые будут переносить данные этого соединения. Если не указывается определенное значение параметра, используется величина, заданная по умолчанию.

Например, приложение может выбрать требуемое значение для приоритета IP или типа обслуживания. Поскольку каждая из соединяемых сторон независимо друг от друга устанавливает собственный приоритет и тип обслуживания, теоретически эти значения могут отличаться для различных направлений потоков данных. Как правило, на практике применяются одинаковые значения для каждого направления обмена.

Когда в приложении задействованы варианты безопасности для правительственных или военных учреждений, каждая из конечных точек соединения должна использовать одинаковые уровни безопасности, иначе такое соединение не будет установлено.

10.5 Пересылка данных

Пересылка данных начинается после завершения трехшагового подтверждения создания соединения (см. рис. 10.9). Стандарт TCP позволяет включать в сегменты подтверждения обычные данные, но они не будут доставляться приложению, пока создание соединения не завершится. Для упрощения нумерации применяются 1000-байтные сообщения. *Каждый сегмент заголовка TCP имеет поле ACK, идентифицирующее порядковый номер байта, который предполагается получить от партнера по соединению.*

Первый посланный клиентом сегмент содержит байты от 1001 до 2000. В его поле ACK должно находиться значение 3001, что указывает порядковый номер байта, который предполагается получить от сервера.

Сервер отвечает клиенту сегментом, содержащим 1000 байт данных (начинающихся с номера 3001). В его поле ACK заголовка TCP будет указано, что байты с 1001 по 2000 уже успешно получены, поэтому следующий ожидающийся от клиента порядковый номер сегмента должен быть 2001.

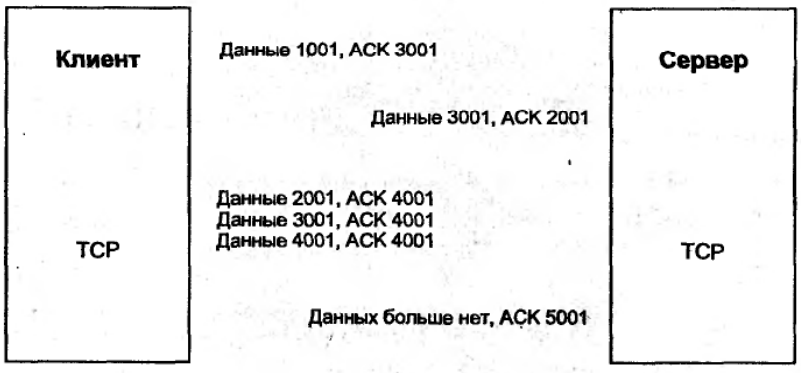

Рис. 10.9. Простой поток обмена данными и ACK

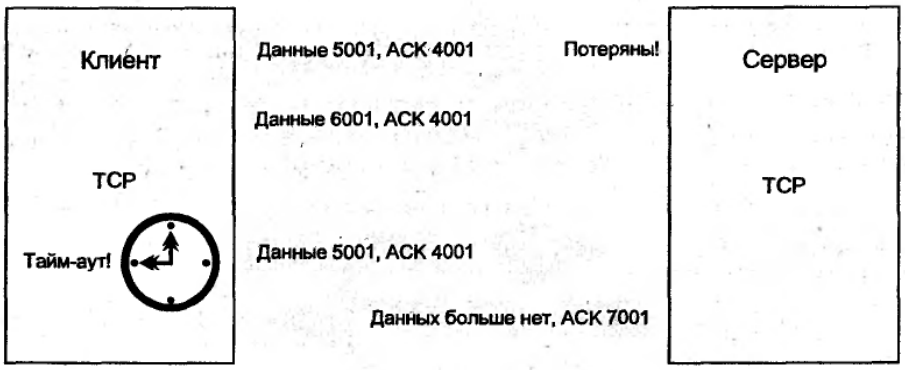

Рис. 10.10. Потеря данных и повторная трансляция

Далее клиент посылает сегменты, начинающиеся с байтов 2001, 3001 и 4001 в указанной последовательности. Отметим, что клиент не ожидает ACK после каждого из посланных сегментов. Данные пересылаются партнеру до заполнения его буферного пространства (ниже мы увидим, что получатель может очень точно указать объем пересылаемых ему данных).

Сервер экономит пропускную способность соединения, используя единственный ACK для указания успешности пересылки всех сегментов.

На рис. 10.10 показана пересылка данных при потере первого сегмента. По завершении тайм-аута пересылка сегмента повторяется. Отметим, что, получив потерянный сегмент, приемник отправляет один ACK, подтверждающий пересылку обоих сегментов.

10.6 Закрытие соединения

Нормальное завершение соединения выполняется с помощью той же процедуры тройного рукопожатия, что и при открытии соединения. Каждая из сторон может начать закрытие соединения по следующему сценарию:

A: "Я закончил работу. Данных для пересылки больше нет".

B: "Хорошо".

B: "Я тоже завершил работу".

A: "Хорошо".

Допустим и такой сценарий (хотя он используется крайне редко):

A: "Я закончил работу. Данных для пересылки больше нет".

B: "Хорошо. Однако есть какие-то данные..."

B: "Я тоже завершил работу".

A: "Хорошо".

В рассмотренном ниже примере соединение закрывает сервер, как это часто происходит для связей клиент/сервер. В данном случае после ввода пользователем в сеансе *telnet* команды logout (выйти из системы) сервер инициирует запрос на закрытие соединения. В ситуации, показанной на рис. 10.11, выполняются следующие действия:

1. Приложение на сервере указывает TCP на закрытие соединения.
2. TCP сервера посылает заключительный сегмент (Final Segment — FIN), информируя своего партнера о том, что данных для отправки больше нет.
3. TCP клиента посылает ACK в сегменте FIN.
4. TCP клиента сообщает своему приложению, что сервер хочет закрыть соединение.
5. Клиентское приложение сообщает своему TCP о закрытии соединения.

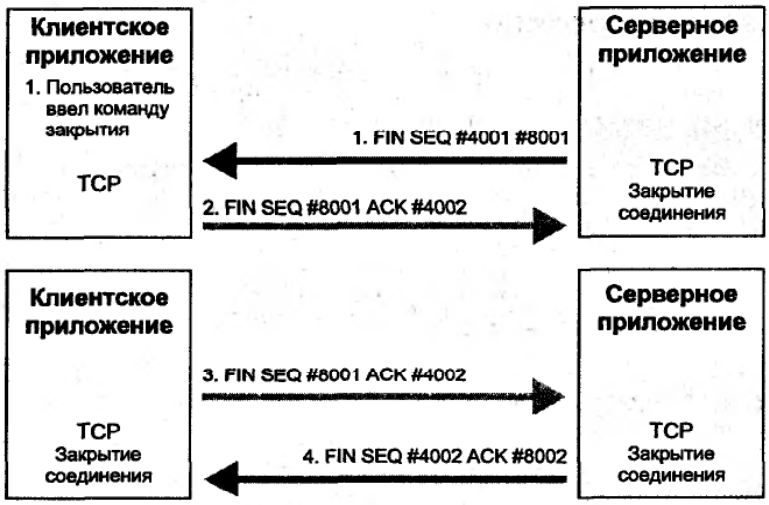

Рис. 10.11. Закрытие соединения

6. TCP клиента посылает сообщение FIN.
7. TCP сервера получает FIN от клиента и отвечает на него сообщением ACK.
8. TCP сервера указывает своему приложению на закрытие соединения.

Обе стороны могут одновременно начать закрытие. В этом случае обычное закрытие соединения завершается после отправки каждым из партнеров сообщения ACK.

10.6.1 Внезапное завершение

Каждая из сторон может запросить внезапное завершение (abrupt close) соединения. Это допустимо, когда приложение желает завершить соединение или когда TCP обнаруживает серьезную коммуникационную проблему, которую не может разрешить собственными средствами. Внезапное завершение запрашивается посылкой партнеру одного или нескольких сообщений reset (сброс), что указывается определенным флажком в заголовке TCP.

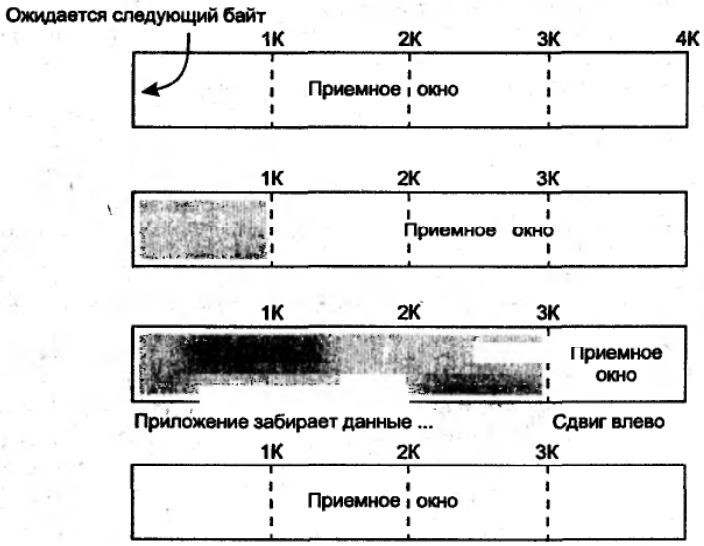

Рис. 10.12. Приемное окно входного буфера

10.7 Управление потоком

Получатель TCP загружается поступающим потоком данных и определяет, какой объем информации он сможет принять. Это ограничение воздействует на отправителя TCP. Представленное ниже объяснение данного механизма является концептуальным, и разработчики могут по-разному реализовать его в своих продуктах.

Во время установки соединения каждый из партнеров выделяет пространство для входного буфера соединения и уведомляет об этом противоположную сторону. Обычно объем буфера выражается целым числом максимальных размеров сегмента.

Поток данных поступает во входной буфер и сохраняется в нем до пересылки в приложение (определяемое портом TCP). На рис. 10.12 показан входной буфер, способный принять 4 Кбайт.

Буферное пространство заполняется по мере поступления данных. Когда приложение-получатель забирает данные из буфера, освободившееся место становится доступным для новых поступающих данных.

10.7.1 Приемное окно

Приемное окно (receive window) — любое пространство во входном буфере, еще не занятое данными. Данные остаются во входном буфере, пока не будут задействованы целевым приложением. Почему приложение не забирает данные сразу?

Ответить на этот вопрос поможет простой сценарий. Предположим, что клиент переслал файл на сервер FTP, работающий на очень загруженном многопользовательском компьютере. Программа FTP далее должна прочитать данные из буфера и записать их на диск. Когда сервер выполняет операции ввода/вывода на диск, программа ожидает завершения этих операций. В это время может запуститься другая программа (например, по расписанию) и, пока программа FTP запустится снова, в буфер уже поступят следующие данные.

Приемное окно расширяется от последнего подтвержденного байта до конца буфера. На рис. 10.12 сначала доступен весь буфер и, следовательно, доступно приемное окно в 4 Кбайт. Когда поступит первый Кбайт, приемное окно сократится до 3 Кбайт (для простоты мы будем считать, что каждый сегмент имеет размер в 1 Кбайт, хотя на практике это значение меняется в зависимости от потребностей приложения). Поступление следующих двух сегментов по 1 Кбайту приведет к сокращению приемного окна до 1 Кбайта.

Далее приложение примет 3 Кбайт данных из буфера, освобождая место для приема следующей информации. Мысленно это можно представить как *сдвиг* окна влево. А в буфере доступными станут уже 4 Кбайт.

Каждый посланный приемником ACK содержит сведения о текущем состоянии приемного окна, в зависимости от которого регулируется поток данных от источника.

По большей части размер входного буфера устанавливается во время запуска соединения, хотя стандарт TCP и не оговаривает реализации управления этим буфером. Входной буфер может увеличиваться или уменьшаться, осуществляя обратную связь с отправителем.

Что произойдет, если поступивший сегмент можно разместить в приемном окне, но он поступил не по порядку? Обычно считается, что все реализации хранят поступившие данные в приемном окне и посылают подтверждение (ACK) только для целого непрерывного блока из нескольких сегментов. Это правильный способ, поскольку иначе при отбрасывании данных, пришедших не по порядку, существенно снизится производительность.

10.7.2 Окно отправки

Система, передающая данные, должна отслеживать две характеристики: сколько данных уже было отправлено и подтверждено, а также текущий размер приемного окна получателя. Активное *пространство отправки* (send space) расширяется от первого неподтвержденного октета к левому краю текущего приемного окна. Часть *окна*, используемая *для отправки*, указывает, сколько еще дополнительных данных можно послать партнеру.

Начальный порядковый номер и начальный размер приемного окна задаются во время установки соединения. Рис. 10.13 иллюстрирует некоторые особенности механизма пересылки данных.

1. Отправитель начинает работу с окном отправки в 4 Кбайт.
2. Отправитель пересылает 1 Кбайт. Копия этих данных сохраняется до получения подтверждения (ACK), поскольку может потребоваться их повторная передача.
3. Прибывает сообщение ACK для первого Кбайта, и отправляются следующие 2 Кбайт данных. Результат показан в третьей сверху части рис. 10.13. Хранение 2 Кбайт продолжается.

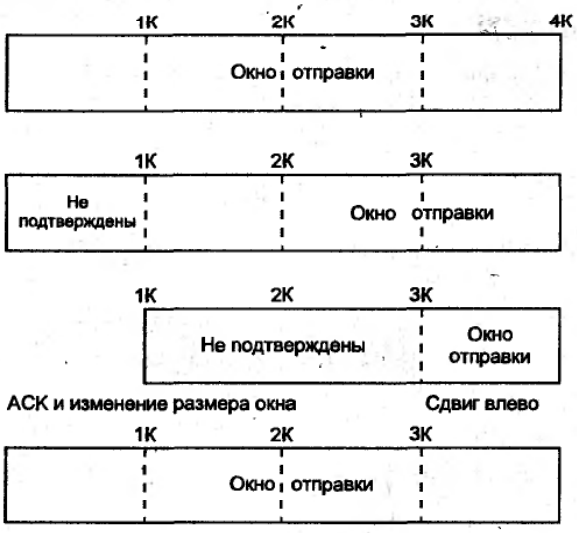

Рис. 10.13. Окно отправки

4. Наконец поступает ACK для всех переданных данных (т.е. все они получены приемником). ACK восстанавливает размер окна отправки в 4 Кбайт.

Следует указать на несколько интересных особенностей:

- Отправитель не дожидается ACK для каждого из посылаемых сегментов данных. Единственным ограничением на пересылку является размер приемного окна (например, отправитель должен пересылать только 4 К однобайтовых сегментов).
- Предположим, что отправитель посылает данные в нескольких очень коротких сегментах (например, по 80 байт). В этом случае данные могут быть переформатированы для более эффективной передачи (например, в единый сегмент).

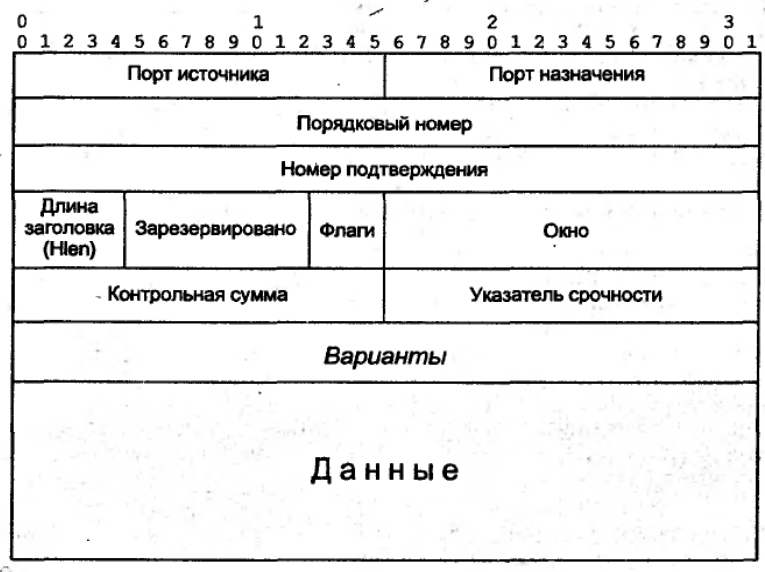

Рис. 10.14. Сегмент TCP

10.8 Заголовок TCP

На рис. 10.14 показан формат сегмента (заголовок TCP и данные). Заголовок начинается с идентификаторов портов источника и назначения. Следующее далее поле *порядкового номера* (sequence number) указывает позицию в исходящем потоке данных, которую занимает данный сегмент. Поле *ACK* (подтверждения) содержит сведения о предполагаемом следующем сегменте, который должен появиться во входном потоке данных.

Существуют шесть флагов:

URG Равен 1 для срочных данных

ACK Равен 1 для всех сегментов, кроме начального

PSH Указывает на необходимость своевременной доставки данных

RST Индикатор ошибки, используется и для завершения сеанса

SYN Равен 1 во время установки соединения

FIN Равен 1 при нормальном закрытии

Поле *смещения данных* (Data Offset) содержит размер заголовка TCP в 32-разрядных словах. Заголовок TCP должен заканчиваться на 32-битной границе.

10.8.1 Вариант максимального размера сегмента

Параметр "*максимальный размер сегмента*" (maximum segment size — MSS) применяется для объявления о наибольшем куске данных, который может быть принят и обработан системой. Однако название несколько неточно. Обычно в TCP *сегмент* рассматривается как заголовок плюс данные. Однако *максимальный размер сегмента* определяется как:

Размер наибольшей датаграммы, которую можно принять, — 40

Другими словами, MSS отражает наибольшую *полезную нагрузку* в приемнике при длине заголовков TCP и IP по 20 байт. Если имеются дополнительные параметры, их длину следует вычесть из общего размера. Следовательно, количество данных, которые можно переслать в сегменте, определяется как:

Заявленное значение MSS + 40 – (сумма длин заголовков TCP и IP)

Обычно партнеры обмениваются значениями MSS в начальных сообщениях SYN при открытии соединения. Если система не анонсирует величину максимального размера сегмента, используется значение по умолчанию в 536 байт.

Размер максимального сегмента кодируется 2-байтовой вводной частью со следующим далее 2-байтовым значением, т.е. наибольшая величина будет составлять $2^{16} - 1$ (65 535 байт).

MSS накладывает жесткое ограничение на пересылаемые в TCP данные: приемник не сможет обработать большие значения. Однако отправитель использует сегменты *меньшего размера*, поскольку для соединения определяется еще размер MTU по пути следования.

10.8.2 Использование полей заголовка в запросе на соединение

Первый сегмент, посылаемый для открытия соединения, имеет значение флага SYN, равное 1, и флага ACK — 0. Начальный SYN является *единственным* сегментом, имеющим поле ACK со значением 0. Отметим, что средства защиты пользуются этим признаком для выявления входных запросов на сеанс TCP.

Поле *порядкового номера* содержит *начальный порядковый номер* (initial sequence number), поле *окна* — начальный размер *приемного окна*. Единственным определенным на сегодняшний момент параметром TCP является максимальный размер сегмента (когда он не указан, используется значение по умолчанию в 536 байт), который предполагает получать TCP. Это значение занимает 32 бита и обычно присутствует в запросе на соединение в поле *вариантов* (Option). Длина заголовка TCP, содержащего значение MSS, составляет 24 байта.

10.8.3 Использование полей заголовка в ответе на запрос соединения

В разрешающем ответе на запрос соединения оба флага (SYN и ACK) равны 1. Отвечающая система указывает начальный порядковый номер в соответствующем поле, а размер приемного окна — в поле *Window*. Максимальный размер сегмента, который желает использовать получатель, обычно находится

в ответе на запрос о соединении (в поле *вариантов*). Это значение может отличаться от значения запрашивающей соединение стороны, т.е. могут применяться две разные величины.

Запрос на соединение может быть отклонен посредством указания в ответе флага сброса (RST) со значением 1.

10.8.4 Выбор начального порядкового номера

Спецификация TCP предполагает, что во время установки соединения каждая из сторон выбирает *начальный порядковый номер* (по текущему значению 32-разрядного внутреннего таймера). Как это выполняется?

Представим, что произойдет при крахе системы. Предположим, что пользователь открыл соединение как раз перед крахом и отправил небольшое количество данных. После восстановления система уже не помнит ничего из того, что делалось перед крахом, включая уже запущенные соединения и присвоенные номера портов. Пользователь повторно устанавливает соединение. Номера портов не совпадают с первоначальными присваиваниями, и некоторые из них, возможно, уже используются другими соединениями, установленными за несколько секунд до краха.

Поэтому другая сторона в самом конце соединения может не знать о том, что ее партнер прошел через крах и его работа затем была восстановлена. Все это приведет к серьезным сбоям в работе, особенно когда проходит долгое время, пока старые данные не пройдут по сети и не смешаются с данными от вновь созданного соединения. Выбор по таймеру старта с обновлением (fresh start) позволяет исключить подобные проблемы. Старые данные будут иметь иную нумерацию, чем диапазон порядковых номеров нового соединения. Хакеры при фальсификации IP-адреса источника для доверительного хоста пытаются получить доступ к компьютерам с помощью указания в сообщении предсказуемого начального порядкового номера. Криптографическая функция хеширования на основе внутренних ключей служит лучшим способом для выбора защищенных начальных номеров.

10.8.5 Общепринятое использование полей

При подготовке заголовка TCP к пересылке порядковый номер первого октета передаваемых данных указывается в поле *последовательного номера* (Sequence Number).

Номер следующего октета, ожидаемого от партнера по соединению, заносится в поле *подтверждения* (Acknowledgment Number) при установке бита ACK в 1. Поле *окна* (Window) предназначено для текущего размера приемного окна. В этом поле содержится *количество байтов от номера подтверждения, которое можно принять*. Отметим, что это значение позволяет точно управлять потоком данных. С помощью этого значения партнер указывает реальное состояние приемного окна в течение сеанса обмена.

Если приложение указывает на операцию выталкивания в TCP, то флаг PUSH устанавливается в 1. Принимающая TCP должна отреагировать на этот флаг быстрой доставкой данных в приложение, как только отправитель захочет их переслать.

Флаг URGENT (срочность) при значении 1 предполагает срочную пересылку данных, а соответствующий указатель должен ссылаться на последний октет срочных данных. Типичным использованием срочных данных является пересылка из терминала сигналов на отмену или прерывание.

Срочные данные часто называют *информацией вне полосы пропускания* (out-of-band). Однако этот термин неточен. Срочные данные пересылаются в обычном потоке TCP, хотя отдельные реализации могут иметь специальные механизмы для указания приложению на поступление срочных данных, а приложение должно проверить содержимое срочных данных, прежде чем поступят все байты сообщения.

Флаг RESET (сброс) имеет значение 1, когда следует аварийно завершить соединение. Этот же флаг устанавливается в ответе при поступлении сегмента, не связанного ни с одним из текущих соединений TCP.

Флаг FIN устанавливается в 1 для сообщений о закрытии соединения.

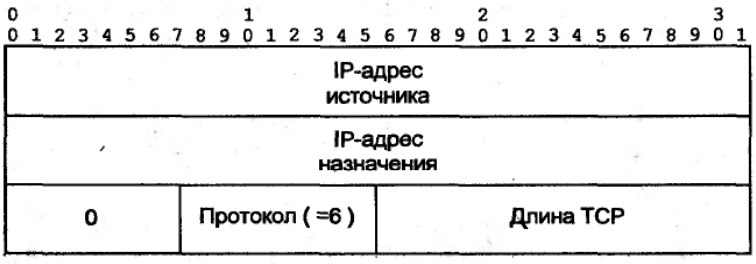

Рис. 10.15. Поле псевдозаголовка включается в контрольную сумму TCP

10.8.6 Контрольная сумма

Контрольная сумма IP предназначена только для заголовка IP, а контрольная сумма TCP вычисляется для всего сегмента, а также для псевдозаголовка, созданного из заголовка IP. Во время вычисления контрольной суммы TCP соответствующее поле имеет значение 0. На рис. 10.15 показан псевдозаголовок, очень напоминающий тот, что используется в контрольной сумме UDP.

Длина TCP вычисляется сложением длины заголовка TCP с длиной данных. Контрольная сумма TCP является *обязательной*, не как в UDP. Контрольная сумма поступившего сегмента сначала вычисляется приемником, а затем сравнивается с содержимым поля контрольной суммы заголовка TCP. Если значения не совпадут, сегмент отбрасывается.

10.9 Пример сегмента TCP

Рис. 10.16, протокол работы анализатора *Sniffer* компании Network General, представляет собой последовательность сегментов TCP. Первые три сегмента устанавливают соединение между клиентом и сервером *Telnet*. В последнем сегменте переносится 12 байт данных.

```
1. SYN
TCP:  -- TCP header --
TCP:
TCP:  Source port = 2345
TCP:  Dest port = 23 (Telnet)
TCP:  Initial sequence number = 16 421 121

TCP:  Data offset = 24 bytes
TCP:  Flags = 02
TCP:  ..0. .... = (No urg ptr)
TCP:  ...0 .... = (No ack)
TCP:  .... 0... = (No push)
TCP:  .... .0.. = (No reset)
TCP:  .... ..1. = SYN
TCP:  .... ...0 = (No FIN)
TCP:  Window = 2048
TCP:  Checksum = F2DA (correct)
TCP:
TCP:  Options follow
TCP:  Max segment size = 1460
```

```
2. SYN
TCP:  -- TCP header --
TCP:
TCP:  Source port = 23 (Telnet)
TCP:  Dest port = 2345
TCP:  Initial sequence number = 390 272 001
TCP:  Acknowledgment number = 16 421 122
TCP:  Data offset = 24 bytes
TCP:  Flags = 12
TCP:  ..0. .... = (No urg ptr)
TCP:  ...1 .... = Acknowledgment
TCP:  .... 0... = (No push)
TCP:  .... .0.. = (No reset)
TCP:  .... ..1. = SYN
TCP:  .... ...0 = (No FIN)
TCP:  Window = 4096
TCP:  Checksum = C13A (correct)
TCP:
TCP:  Options follow
TCP:  Max segment size = 1024
```

```
3. ACK
TCP:  -- TCP header --
TCP:
TCP:  Source port = 2345
TCP:  Dest port = 23 (Telnet)
TCP:  Sequence number = 16 421 122
TCP:  Acknowledgment number = 390 272 002
TCP:  Data offset = 20 bytes
TCP:  Flags = 10
TCP:  ..0. .... = (No urg ptr)
TCP:  ...1 .... = Acknowledgment
TCP:  .... 0... = (No push)
TCP:  .... .0.. = (No reset)
TCP:  .... ..0. = (No SYN)
TCP:  .... ...0 = (No FIN)
TCP:  Window = 2048
TCP:  Checksum = DF43 (correct)
TCP:  No TCP options
```

```
4. DATA
TCP:  -- TCP header --
TCP:
TCP:  Source port = 23 (Telnet)
TCP:  Dest port = 2345
TCP:  Sequence number = 390 272 002
TCP:  Acknowledgment number = 16 421 122
TCP:  Data offset = 20 bytes
TCP:  Flags = 18
TCP:  ..0. .... = (No urg ptr)
TCP:  ...1 .... = Acknowledgment
TCP:  .... 1... = Push
TCP:  .... .0.. = (No reset)
TCP:  .... ..0. = (No SYN)
TCP:  .... ...0 = (No FIN)
TCP:  Window = 4096
TCP:  Checksum = 9FEF (correct)
TCP:  No TCP options
TCP:  [12 byte(s) of data]
```

Рис. 10.16. Отображение заголовка TCP анализатором Sniffer

Анализатор *Sniffer* транслирует большинство значений в десятичный вид. Однако значения флагов выводятся как шестнадцатеричные. Флаг со значением 12 представляет собой 010010. Контрольная сумма выводится тоже в шестнадцатеричном виде.

10.10 Поддержка работы сеанса

10.10.1 Зондирование окна

Скоростной отправитель и медленный получатель могут сформировать приемное окно размером в 0 байт. Этот результат называется *закрытием окна* (close window). Когда появляется свободное место для обновления размера приемного окна, используется ACK. Однако, если такое сообщение будет потеряно, обе стороны должны будут ожидать до бесконечности.

Для избежания такой ситуации отправитель устанавливает *сохраняемый таймер* (persist timer) при закрытии премного окна. Значением таймера берется тайм-аут повторной пересылки. По завершении работы таймера партнеру отсылается сегмент *зондирования окна* (window probe; в некоторых реализациях в него включаются и данные). Зондирование заставляет партнера посылать назад ACK, который сообщает о текущем статусе окна.

Если окно все еще остается нулевого размера, удваивается значение сохраняемого таймера. Этот процесс повторяется, пока значение таймера не достигнет максимума в 60 с. TCP продолжит посылать сообщения зондирования каждые 60 с — до открытия окна, до завершения процесса пользователем или до завершения по тайм-ауту приложения.

10.11 Завершение сеанса

10.11.1 Тайм-аут

Работа партнера по соединению может завершиться крахом либо полностью прерваться вследствие неисправности шлюза или связи. Чтобы предотвратить повторную пересылку данных в TCP, существует несколько механизмов.

Достигнув первого порогового значения для повторной пересылки (ретрансляции), TCP указывает IP на необходимость проверки отказавшего маршрутизатора и одновременно информирует приложение о возникшей проблеме. TCP продолжает пересылку данных, пока не будет достигнуто второе граничное значение, и только после этого разрывает соединение.

Разумеется, перед тем как это произойдет, может поступить сообщение ICMP о недостижимости точки назначения по каким-то причинам. В некоторых реализациях даже после этого TCP продолжит попытки доступа к точке назначения до завершения интервала тайм-аута (после чего проблема может быть зафиксирована). Далее приложению сообщается о недостижимости точки назначения.

Приложение может установить собственный тайм-аут на доставку данных и проводить собственные операции при завершении этого интервала. Обычно производится разрыв соединения.

10.11.2 Поддержание соединения

Когда незавершенное соединение долгое время имеет данные для пересылки, оно получает статус неактивного. Во время периода неактивности может произойти крах сети или обрыв физических линий связи. Как только сеть снова станет работоспособной, партнеры продолжат обмен данными, не прерывая сеанса связи. Данная стратегия соответствовала требованиям Министерства обороны.

Однако любое соединение — активное или неактивное — занимает много памяти компьютера. Некоторым администраторам нужно возвратить в системы неиспользованные ресурсы. Поэтому многие реализации TCP способны посылать сообщение о *поддержании соединения* (keep-alive), тестирующее неактивные соединения. Такие сообщения периодически отправляются партнеру для проверки его существования в сети. В ответ должны поступать сообщения ACK. Использование сообщений о поддержании соединения не является обязательным. Если в системе имеется такая возможность, приложение может отменить ее собственными средствами. Предполагаемый период *по умолчанию* для тайм-аута поддержания соединения составляет целых два часа!

Вспомним, что приложение может установить собственный таймер, по которому на своем уровне и будет принимать решение о завершении соединения.

10.12 Производительность

Насколько эффективна работа TCP? На производительность ресурсов влияют многие факторы, из которых основными являются память и полоса пропускания (см. рис. 10.17).

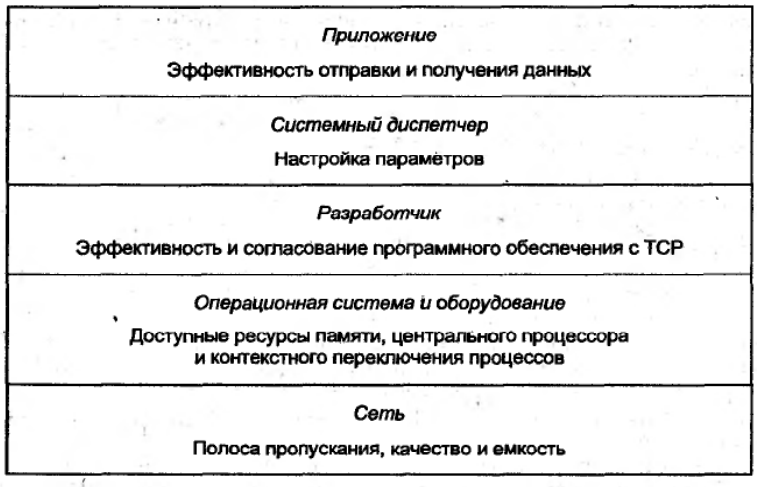

Рис. 10.17. Факторы производительности TCP

Полоса пропускания и задержки в используемой физической сети существенно ограничивают пропускную способность. Плохое качество пересылки данных приводит к большому объему отброшенных датаграмм, что вызывает повторную пересылку и, как следствие, снижает эффективность полосы пропускания.

Приемная сторона должна обеспечить достаточное буферное пространство, позволяющее отправителю выполнять пересылку данных без пауз в работе. Это особенно важно для сетей с большими задержками, в которых между отправкой данных и получением ACK (а также при согласовании размера окна) проходит большой интервал времени. Для поддержания устойчивого потока данных от источника принимающая сторона должна иметь окно размером не менее чем произведение полосы пропускания на задержку.

Например, если источник может отсылать данные со скоростью 10 000 байт/с, а на возврат ACK тратится 2 с, то на другой стороне нужно обеспечить приемное окно размером не менее 20 000 байт, иначе поток данных не будет непрерывным. Приемный буфер в 10 000 байт вполовину снизит пропускную способность.

Еще одним важным фактором для производительности является способность хоста реагировать на высокоприоритетные события и быстро выполнять *контекстное переключение*, т.е. завершать одни операции и переключаться на другие. Хост может интерактивно поддерживать множество локальных пользователей, пакетные фоновые процессы и десятки одновременных коммуникационных соединений. Контекстное переключение позволяет обслуживать все эти операции, скрывая нагрузки на систему. Реализации, в которых проведена интеграция TCP/IP с ядром операционной системы, могут существенно снизить нагрузки от использования контекстного переключения.

Ресурсы центрального процессора компьютера необходимы для операций по обработке заголовков TCP. Если процессор не может быстро вычислять контрольные суммы, это приводит к снижению скорости пересылки данных по сети.

Кроме того, разработчикам следует обращать внимание на упрощение конфигурирования параметров TCP, чтобы сетевой администратор мог настроить их в соответствии со своими локальными требованиями. Например, возможность настройки размера буфера на полосу пропускания и задержку сети существенно улучшит производительность. К сожалению, многие реализации не уделяют этому вопросу должного внимания и жестко программируют коммуникационные параметры.

Предположим, что сетевое окружение совершенно: имеются достаточные ресурсы и контекстное переключение выполняется быстрее, чем ковбои выхватывают свои револьверы. Будет ли получена прекрасная производительность?

Не всегда. Имеет значение и качество разработки программного обеспечения TCP. За долгие годы в различных реализациях TCP были диагностированы и решены многие проблемы производительности. Можно считать, что наилучшим будет программное обеспечение, соответствующее документу RFC 1122, в котором определены требования к коммуникационному уровню хостов Интернета.

Не менее важно исключение *синдрома "бестолкового окна"* и применение алгоритмов Джекобсона, Керна и Партриджа (эти интересные алгоритмы будут рассмотрены ниже).

Разработчики программного обеспечения могут получить существенные выгоды, создавая программы, исключающие ненужные пересылки небольших объемов данных и имеющие встроенные таймеры для освобождения сетевых ресурсов, не используемых в текущий момент.

10.13 Алгоритмы повышения производительности

Переходя к знакомству с достаточно сложной частью TCP, мы рассмотрим механизмы повышения производительности и решения проблем снижений пропускной способности. В этом разделе обсуждаются следующие проблемы:

- *Медленный старт* (slow start) мешает использованию большой доли сетевого трафика для нового сеанса, что может привести к непроизводительным потерям.
- Излечение от *синдрома "бестолкового окна"* (silly window syndrom) предохраняет плохо разработанные приложения от перегрузки сети сообщениями.
- *Задержанный ACK* (delayed ACK) снижает перегрузку посредством сокращения количества независимых сообщений подтверждения пересылки данных.
- *Вычисляемый тайм-аут повторной пересылки* (computing retransmission timeout) основывается на согласовании реального времени сеанса, уменьшая объем ненужных повторных пересылок, но при этом не вызывает больших задержек для реально необходимых обменов данными.
- Торможение пересылки TCP при *перегрузках* в сети позволяет маршрутизаторам вернуться в исходный режим и совместно использовать сетевые ресурсы для всех сеансов.
- Отправка *дублированных ACK* (duplicate ACK) при получении сегмента вне последовательности отправки позволяет партнерам выполнить повторную пересылку до наступления тайм-аута.

10.13.1 Медленный старт

Если дома одновременно включить все бытовые электроприборы, произойдет перегрузка электрической сети. В компьютерных сетях *медленный старт* не позволяет перегореть сетевым предохранителям.

Новое соединение, мгновенно запускающее пересылку большого объема данных в уже и так нагруженной сети, может привести к проблемам. Идея медленного старта заключается в обеспечении новому соединению успешного запуска с медленным увеличением скорости пересылки данных в соответствии с реальной нагрузкой на сеть. Отправитель ограничивается размером нагрузочного окна, а не большим по размеру приемным окном.

Нагрузочное окно (congestion window) начинается с размера в 1 сегмент. Для каждого сегмента с успешно полученным ACK размер нагрузочного окна увеличивается на 1 сегмент, пока оно остается меньше, чем приемное окно. Если сеть не перегружена, нагрузочное окно постепенно достигнет размера приемного окна. При нормальном состоянии пересылки размеры этих окон будут совпадать.

Отметим, что медленный старт — не такой уж и медленный. После первого ACK размер нагрузочного окна равен 2-м сегментам, а после успешного получения ACK для двух сегментов размер может увеличиться до 8 сегментов. Другими словами, размер окна увеличивается экспоненциально.

Предположим, что вместо получения ACK возникла ситуация тайм-аута. Поведение нагрузочного окна в таком случае рассматривается ниже.

10.13.2 Синдром "бестолкового окна"

В первых реализациях TCP/IP разработчики столкнулись с феноменом *синдрома "бестолкового окна"* (Silly Window Syndrome — SWS), который проявлялся достаточно часто. Для понимания происходящих событий рассмотрим следующий сценарий, приводящий к нежелательным последствиям, но вполне возможный:

1. Передающее приложение быстро отсылает данные.
2. Принимающее приложение читает из входного буфера по 1 байту данных (т.е. медленно).
3. Входной буфер после чтения быстро заполняется.
4. Принимающее приложение читает 1 байт, и TCP отправляет ACK, означающий "Я имею свободное место для 1 байта данных".
5. Передающее приложение отправляет по сети пакет TCP из 1 байта.
6. Принимающий TCP посылает ACK, означающий "Спасибо. Я получил пакет и не имею больше свободного места".
7. Принимающее приложение опять читает 1 байт и отправляет ACK, и весь процесс повторяется.

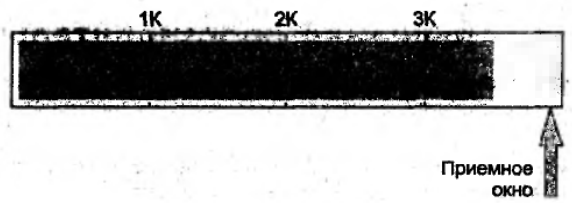

Рис. 10.18. Буфер приемного окна с очень малым размером свободного пространства

Медленное принимающее приложение долго ожидает поступления данных и постоянно подталкивает полученную информацию к левому краю окна, выполняя совершенно бесполезную операцию, порождающую дополнительный трафик в сети.

Реальные ситуации, конечно, не столь экстремальны. Быстрый отправитель и медленный получатель будут обмениваться небольшими (относительно максимального размера сегмента) кусками данных и переключаться по почти заполненному приемному окну. На рис. 10.18 показаны условия для появления синдрома "бестолкового окна".

Решить эту проблему не сложно. Как только приемное окно сокращается на длину, меньшую чем данный целевой размер, TCP начинает обманывать отправителя. В этой ситуации TCP не должен указывать отправителю на *дополнительное* пространство в окне, когда принимающее приложение читает данные из буфера небольшими порциями. Вместо этого нужно держать освобождающиеся ресурсы в секрете от отправителя до тех пор, пока их не будет достаточное количество. Рекомендуется размер в один сегмент, кроме случаев, когда весь входной буфер хранит единственный сегмент (в последнем случае используется размер, равный половине буфера). Целевой размер, о котором должен сообщать TCP, можно выразить как:

minimum (1/2 входного буфера, Максимальный размер сегмента)

TCP начинает обманывать, когда размер окна станет меньше этого размера, и скажет правду, когда размер окна не меньше, чем получаемая по формуле величина. Отметим, что для отправителя нет никакого ущерба, поскольку принимающее приложение все равно не смогло бы обработать большую часть данных, которых оно ожидает.

Предложенное решение легко проверить в рассмотренном выше случае с выводом ACK для каждого из полученных байтов. Этот же способ пригоден и для случая, когда входной буфер может хранить несколько сегментов (как часто бывает на практике). Быстрый отправитель заполнит входной буфер, но приемник укажет, что не имеет свободного места для размещения информации, и не откроет этот ресурс, пока его размер не достигнет целого сегмента.

10.13.3 Алгоритм Нейгла

Отправитель должен независимо от получателя исключить пересылку очень коротких сегментов, аккумулируя данные перед отправлением. Алгоритм Нейгла (Nagle) реализует очень простую идею, позволяющую снизить количество пересылаемых по сети коротких датаграмм.

Алгоритм рекомендует задержать пересылку данных (и их выталкивание) на время ожидания ACK от ранее переданных данных. Аккумулируемые данные пересылаются после получения ACK на ранее отправленную порцию информации, либо после получения для отправки данных в размере полного сегмента или по завершении тайм-аута. Этот алгоритм не следует применять для приложений реального времени, которые должны отправлять данные как можно быстрее.

10.13.4 Задержанный ACK

Еще одним механизмом повышения производительности является способ задержки ACK. Сокращение числа ACK снижает полосу пропускания, которую можно использовать для пересылки другого трафика. Если партнер по TCP чуть задержит отправку ACK, то:

- Можно подтвердить прием нескольких сегментов одним ACK.
- Принимающее приложение способно получить некоторый объем данных в пределах интервала тайм-аута, т.е. в ACK может попасть выходной заголовок, и не потребуется формирование отдельного сообщения.

С целью исключения задержек при пересылке потока полноразмерных сегментов (например, при обмене файлами), ACK должен отсылаться, по крайней мере, для каждого второго полного сегмента.

Многие реализации используют тайм-аут в 200 мс. Но задержанный ACK не снижает скорость обмена. При поступлении короткого сегмента во входном буфере остается еще достаточно свободного места для получения новых данных, а отправитель может продолжить пересылку (кроме того, повторная пересылка обычно выполняется гораздо медленнее). Если же поступает целый сегмент, нужно в ту же секунду ответить на него сообщением ACK.

10.13.5 Тайм-аут повторной пересылки

После отправки сегмента TCP устанавливает таймер и отслеживает поступление ACK. Если ACK не получен в течение периода тайм-аута, TCP выполняет повторную пересылку сегмента (ретрансляцию). Однако каким должен быть период тайм-аута?

Если он слишком короткий, отправитель заполнит сеть пересылкой ненужных сегментов, дублирующих уже отправленную информацию. Слишком же большой тайм-аут будет препятствовать быстрому исправлению действительно разрушенных при пересылке сегментов, что снизит пропускную способность.

Как выбрать правильный промежуток для тайм-аута? Значение, пригодное для высокоскоростной локальной сети, не подойдет для удаленного соединения со множеством попаданий. Значит, принцип "одно значение для любых условий" явно непригоден. Более того, даже для существующего конкретного соединения могут измениться сетевые условия, а задержки — увеличиться или снизиться.

Алгоритмы Джекобсона, Керна и Партриджа (описанные в статьях *Congestion Avoidance and Control*, Van Jacobson, и *Improving Round-Trip Time Estimates in Reliable Transport Protocols*, Karn и Partridge) позволяют адаптировать TCP к изменению сетевых условий. Эти алгоритмы рекомендованы к использованию в новых реализациях. Мы кратко рассмотрим их ниже.

Здравый смысл подсказывает, что наилучшей основой оценки правильного времени тайм-аута для конкретного соединения может быть отслеживание *времени цикла* (round-trip time) как промежутка между отправкой данных и получением подтверждения об их приеме.



Рис. 10.19. Распределение значений времени цикла

Хорошие решения для следующих величин можно получить на основе элементарных статистических сведений (см. рис. 10.19), которые помогут вычислить время тайм-аута. Однако не нужно полагаться на усредненные величины, поскольку более половины оценок будет больше, чем среднестатистическая величина. Рассмотрев пару отклонений, можно получить более правильные оценки, учитывающие нормальное распределение и снижающие слишком долгое время ожидания повторной пересылки.

Нет необходимости в большом объеме вычислений для получения формальных математических оценок отклонений. Можно использовать достаточно грубые оценки на основе абсолютной величины разницы между последним значением и среднестатистической оценкой:

$$\text{Последнее отклонение} = |\ \text{Последний цикл} - \text{Средняя величина}\ |$$

Для вычисления правильного значения тайм-аута нужно учитывать еще один фактор — изменение времени цикла из-за текущих сетевых условий. Происходившее в сети в последнюю минуту более важно, чем то, что было час назад.

Предположим, что вычисляется среднее значение цикла для очень длинного по времени сеанса. Пусть вначале сеть была мало загружена, и мы определили 1000 небольших значений, однако далее произошло увеличение трафика с существенным увеличением времени задержки.

Например, если 1000 значений дали среднестатистическую величину в 170 единиц, но далее были замерены 50 значений со средним в 282, то текущее среднее будет:

$$170 \times 1000/1050 + 282 \times 50/1050 = 175$$

Более резонной будет величина *сглаженного времени цикла* (Smoothed Round-Trip Time — SRTT), которая учитывает приоритет более поздних значений:

Новое SRTT = (1 − α) × (старое SRTT) + α × Последнее значение цикла

Значение α находится между 0 и 1. *Увеличение* α приводит к большему влиянию текущего времени цикла на сглаженное среднее значение. Поскольку компьютеры быстро могут выполнять деление на степени числа 2, сдвигая двоичные числа направо, для α всегда выбирается значение $(1/2)^n$ (обычно 1/8), поэтому:

Новое SRTT = 7/8 × старое SRTT + 1/8 × Последнее время цикла

В таблице 10.2 показано, как формула для SRTT подстраивается под текущее значение SRTT в 230 единиц, когда изменение в сетевых условиях приводит к последовательному увеличению времени цикла (при условии, что не наступает тайм-аут). Значения в столбце 3 используются как значения столбца 1 для следующей строки таблицы (т.е. как старое SRTT).

Таблица 10.2

Вычисление сглаженного времени цикла

Старое SRTT	Самое последнее RTT	(7/8) × (старое SRTT) + (1/8) × (RTT)
230.00	294	238.00
238.00	264	241.25
241.25	340	253.59
253.59	246	252.64
252.64	201	246.19
246.19	340	257.92
257.92	272	259.68
259.68	311	266.10
266.10	282	268.09
268.09	246	265.33
265.33	304	270.16
270.16	308	274.89
274.89	230	269.28
269.28	328	276.62
276.62	266	275.29
275.29	257	273.00
273.00	305	277.00

Теперь возникает вопрос о выборе значения для тайм-аута повторной пересылки. Анализ величин времени цикла показывает существенное отклонение этих значений от текущей средней величины. Имеет смысл установить границу для величины отклонений (девиаций). Хорошие величины для тайм-аута повторной пересылки (в стандартах RFC эту величину именуют Retransmission TimeOut — RTO) дает следующая формула с ограничением сглаженного отклонения (SDEV):

T = Тайм-аут повторной пересылки = SRTT + 2 × SDEV

Именно эта формула рекомендована в RFC 1122. Однако некоторые реализации используют другую:

T = SRTT + 4 × SDEV

Для вычисления SDEV сначала определяют абсолютное значение текущего отклонения:

DEV = | Последнее время цикла − старое SRTT |

Затем используют формулу сглаживания, чтобы учесть последнее значение:

Новое SDEV = 3/4 × старое SDEV + 1/4 × DEV

Остается один вопрос — какие взять начальные значения? Рекомендуется:

Начальный тайм-аут = 3 с

Начальный SRTT = 0

Начальный SDEV = 1,5 с

Ван Джекобсон определил быстрый алгоритм, который очень эффективно вычисляет тайм-аут повторной пересылки данных.

10.13.6 Пример статистики

Насколько успешно будет работать вычисленный выше тайм-аут? При реализации полученного значения наблюдались значительные повышения производительности. Примером могут служить статистические данные команды *netstat*, полученные на системе *tigger* — сервере Интернета, к которому обращается множество хостов со всего мира.

```
tcp:
1301644 packets sent
        879137 data packets (226966295 bytes)
        21815 data packets (8100927 bytes) retransmitted

2012869 packets received
        762469 acks (for 226904227 bytes)
        35803 duplicate acks
        0 acks for unsent data
        1510769 packets (314955304 bytes) received in-sequence
        9006 completely duplicate packets (867042 bytes)
        74 packets with some dup. data (12193 bytes duped)
        13452 out-of-order packets (2515087 bytes)
```

Системой *tigger* были повторно переданы менее чем 2,5% сегментов данных TCP. Для полутора миллионов входящих сегментов данных (остальные являются чистыми сообщениями ACK) дублированы были только 0,6%. При этом следует учитывать, что уровень потерь во входных данных примерно соответствует уровню для выходных сегментов. Таким образом, бесполезный трафик повторной пересылки составляет около 0,6% от общего трафика.

10.13.7 Вычисления после повторной отправки

В представленных выше формулах используется значение времени цикла как интервала между отправкой сегмента и получением подтверждения его приема. Однако предположим, что в течение периода тайм-аута подтверждение не получено и данные должны быть оправлены повторно.

Алгоритм Керна предполагает, что в этом случае не следует изменять время цикла. Текущее сглаженное значение времени цикла и *сглаженное отклонение* сохраняют свои значения, пока не будет получено подтверждение на пересылку некоторого сегмента без его повторной отправки. С этого момента возобновляются вычисления на основе сохраненных величин и новых замеров.

10.13.8 Действия после повторной пересылки

Но что происходит до получения подтверждения? После повторной пересылки поведение TCP радикально меняется в основном из-за потери данных от перегрузки в сети. Следовательно, реакцией на повторную отправку данных будет:

- Снижение скорости повторной пересылки
- Борьба с перегрузкой сети с помощью сокращения общего трафика

10.13.9 Экспоненциальное торможение

После повторной пересылки удваивается интервал тайм-аута. Однако что произойдет при повторном переполнении таймера? Данные будут оправлены еще раз, а период повторной пересылки снова удвоится. Этот процесс называется *экспоненциальным торможением* (exponential backoff).

Если продолжает проявляться неисправность сети, период тайм-аута будет удваиваться до достижения предустановленного максимального значения (обычно — 1 мин). После тайм-аута может быть

отправлен только один сегмент. Тайм-аут наступает и при превышении заранее установленного значения для количества пересылок данных без получения ACK.

10.13.10 Снижение перегрузок за счет уменьшения пересылаемых по сети данных

Сокращение объема пересылаемых данных несколько сложнее, чем рассмотренные выше механизмы. Оно начинает работать, как и уже упомянутый медленный старт. Но, поскольку устанавливается граница для уровня трафика, который может в начальный момент привести к проблемам, будет реально замедляться скорость обмена вследствие увеличения размера нагрузочного окна по одному сегменту. Нужно установить значения границы для реального сокращения скорости отправки. Сначала вычисляется граница опасности (danger threshold):

Граница = 1/2 *minimum* (*текущее нагрузочное окно, приемное окно партнера*)

Если полученная величина будет более двух сегментов, ее используют как границу. Иначе размер границы устанавливается равным двум сегментам. Полный алгоритм восстановления требует:

- Установить размер нагрузочного окна в один сегмент.
- Для каждого полученного ACK увеличивать размер нагрузочного окна на один сегмент, пока не будет достигнута граница (что очень напоминает механизм медленного старта).
- После этого с каждым полученным ACK к нагрузочному окну добавлять меньшее значение, которое выбирается на основе скорости увеличения по одному сегменту для времени цикла (увеличение вычисляется как MSS/N, где N — размер нагрузочного окна в сегментах).

Сценарий для идеального варианта может упрощенно представить работу механизма восстановления. Предположим, что приемное окно партнера (и текущее нагрузочное окно) имело до выявления тайм-аута размер в 8 сегментов, а граница определена в 4 сегмента. Если принимающее приложение мгновенно читает данные из буфера, размер приемного окна останется равным 8 сегментам.

1. Отправляется 1 сегмент (нагрузочное окно = 1 сегмент).
2. Получен ACK — отправляется 2 сегмента.
3. Получен ACK для 2 сегментов — посылается 4 сегмента. (достигается граница).
4. Получен ACK для 4 сегментов. Посылается 5 сегментов.
5. Получен ACK для 5 сегментов. Посылается 6 сегментов.
6. Получен ACK для 6 сегментов. Посылается 7 сегментов.
7. Получен ACK для 7 сегментов. Посылается 8 сегментов (нагрузочное окно по размеру снова стало равно приемному окну).

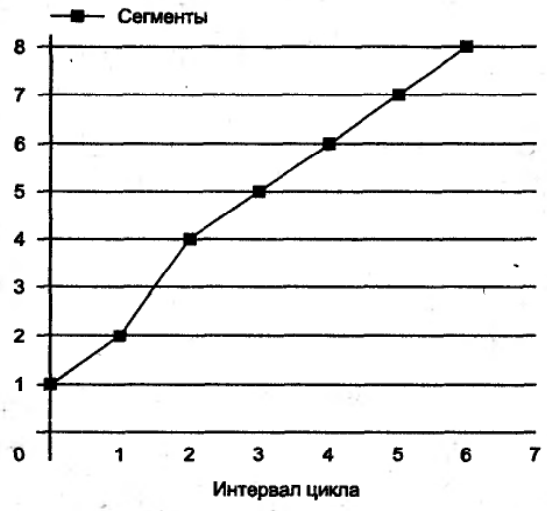

Рис. 10.20. Ограничение скорости пересылки во время перегрузки

Поскольку во время повторной пересылки по тайм-ауту требуется подтверждение приема всех отправленных данных, процесс продолжается до достижения нагрузочным окном размера приемного окна. Происходящие события показаны на рис. 10.20. Размер окна увеличивается экспоненциально, удваиваясь во время периода медленного старта, а по достижении границы увеличение происходит по линейному закону.

10.13.11 Дублированные ACK

В некоторых реализациях применяется необязательная возможность — так называемая *быстрая повторная пересылка* (fast retransmit) — с целью ускорить повторную отправку данных при определенных условиях. Ее основная идея связана с отправкой получателем дополнительных ACK, указывающих на пробел в принятых данных.

Принимая сегмент, поступивший не по порядку, получатель отсылает обратно ACK, указывающий на первый байт *потерянных* данных (см. рис. 10.21).

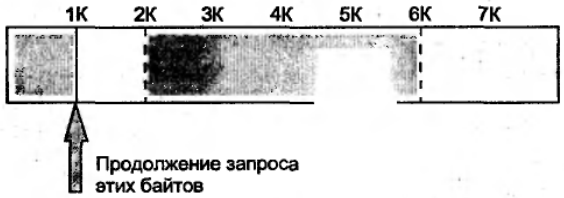

Рис. 10.21. Дублированные ACK

Отправитель не выполняет мгновенной повторной пересылки данных, поскольку IP может и в нормальном режиме доставлять данные получателю без последовательности отправки. Но когда получено несколько дополнительных ACK на дублирование данных (например, три), то отсутствующий сегмент будет отправлен, не дожидаясь завершения тайм-аута.

Отметим, что каждый дублирующий ACK указывает на получение сегмента данных. Несколько дублирующих ACK позволяют понять, что сеть способна доставлять достаточный объем данных, следовательно, не слишком сильно нагружена. Как часть общего алгоритма выполняется небольшое сокращение размера нагрузочного окна при реальном увеличении сетевого трафика. В данном случае процесс радикального изменения размера при восстановлении работы не применяется.

10.13.12 Что делается после подавления источника?

В соответствии со стандартом *Host Requirements* (требования к хостам) TCP должен выполнять тот же самый медленный старт, как это описано выше, при подавлении источника (source quench). Однако сообщение об этом не является целенаправленным или эффективным, поскольку получившее это сообщение соединение может и не создавать слишком большого трафика. Текущая спецификация *Router Requirements* (требования к маршрутизаторам) указывает, что маршрутизаторы *не должны* посылать сообщений о подавлении источника.

10.13.13 Статистика TCP

Наконец, давайте рассмотрим статистические сообщения команды *netstat*, чтобы увидеть в работе многие из описанных выше механизмов.

```
tcp:
1301644 packets sent                                    Пакетами именуются сегменты.
879137 data packets (226966295 bytes)
21815 data packets (8100927 bytes) retransmitted        Повторная пересылка.
132957 ack-only packets (104216 delayed)                Отметим большое количество
                                                        задержанных ACK.
4 URG only packets
1038 window probe packets                               Зондирование открытия окна
                                                        нулевого размера.
248582 window update packets
22413 control packets                                   Это сообщения SYN и FIN.
2012869 packets received
```

```
762469 acks (for 226904227 bytes)
35803 duplicate acks                                  Сигнал о пакетах, прибывших вне
                                                      последовательности.
0 acks for unsent data
1510769 packets (314955304 bytes)
 received in-sequence
9006 completely duplicate packets (867042 bytes)      Результат тайм-аута при реальной
                                                      доставке данных.
74 packets with some dup. data (12193 bytes duped)    С целью большей эффективности
                                                      некоторые данные при повторной
                                                      отправке были перепакетированы
                                                      для включения дополнительных
                                                      байт.
13452 out-of-order packets (2515087 bytes)
530 packets (8551 bytes) of data after window         Возможно, эти данные были
                                                      включены в сообщения
                                                      зондирования.
526 window probes
14158 window update packets
402 packets received after close                      Это последующие повторные
                                                      отправки.
108 discarded for bad checksums                       Неверная контрольная сумма TCP.
0 discarded for bad header offset fields
7 discarded because packet too short
6378 connection requests
9539 connection accepts
14677 connections established (including accepts)
18929 connections closed (including 643 drops)
4100 embryonic connections dropped
572187 segments updated rtt (of 587397 attempts)      Неудачные попытки изменения
                                                      времени цикла, поскольку ACK
                                                      не успел прийти до завершения
                                                      тайм-аута.
11014 retransmit timeouts
26 connections dropped by rexmit timeout              Последующие неудачные попытки
                                                      повторной отправки, что указывает
                                                      на потерянное соединение.
1048 persist timeouts                                 Тайм-ауты по зондированию
                                                      нулевого окна.
535 keepalive timeouts                                Тайм-ауты по проверке
                                                      неработающего соединения.
472 connections dropped by keepalive
```

10.14 Соответствие требованиям разработчика

Текущий стандарт TCP требует, чтобы реализации твердо придерживались процедуры медленного старта при инициализации соединения и использовали алгоритмы Керна и Джекобсона для оценки тайм-аута повторной отправки данных и управления нагрузкой. Тесты показали, что эти механизмы приводят к значительному повышению производительности.

Что произойдет при установке системы, которая не придерживается твердо этих стандартов? Она не сможет обеспечить должную производительность для собственных пользователей, и будет плохим соседом для других систем сети, препятствуя восстановлению нормальной работы после временной перегрузки и создавая чрезмерный трафик, приводящий к отбрасыванию датаграмм.

10.15 Барьеры для производительности

TCP доказал свою гибкость, работая в сетях со скоростью обмена в сотни или миллионы бит за секунду. Этот протокол позволил достичь хороших результатов в современных локальных сетях с топологиями Ethernet, Token-Ring и Fiber Distributed Data Interface (FDDI), а также для низкоскоростных линий связи или соединений на дальние расстояния (подобных спутниковым связям).

TCP разработан так, чтобы реагировать на экстремальные условия, например на перегрузки в сети. Однако в текущей версии протокола имеются особенности, ограничивающие производительность в перспективных технологиях, которые предлагают полосу пропускания в сотни и тысячи мегабайт. Чтобы понять возникающие проблемы, рассмотрим простой (хотя и нереалистичный) пример.

Предположим, что при перемещении файла между двумя системами необходимо выполнять обмен непрерывным потоком настолько эффективно, насколько это возможно. Допустим, что:

- Максимальный размер сегмента приемника — 1 Кбайт.
- Приемное окно — 4 Кбайт.
- *Полоса пропускания позволяет пересылать по два сегмента за 1 с.*
- Принимающее приложение поглощает данные по мере их поступления.
- Сообщения ACK прибывают через 2 с.

Отправитель способен посылать данные непрерывно. Ведь когда выделенный для окна объем будет заполнен, прибывает ACK, разрешающий отправку другого сегмента:

```
SEND SEGMENT 1.
SEND SEGMENT 2.
SEND SEGMENT 3.
SEND SEGMENT 4.
```

Через 2 с:

```
RECEIVE ACK OF SEGMENT 1, CAN SEND SEGMENT 5.
RECEIVE ACK OF SEGMENT 2, CAN SEND SEGMENT 6.
RECEIVE ACK OF SEGMENT 3, CAN SEND SEGMENT 7.
RECEIVE ACK OF SEGMENT 4, CAN SEND SEGMENT 8.
```

Еще через 2 с:

```
RECEIVE ACK OF SEGMENT 5, CAN SEND SEGMENT 9.
. . .
```

Если приемное окно было только в 2 Кбайт, отправитель будет вынужден ждать одну секунду из каждых двух перед отправкой следующих данных. Фактически, чтобы удержать непрерывный поток данных, приемное окно должно иметь размер не менее:

$$\text{Окно} = \text{Полоса пропускания} \times \text{Время цикла}$$

Хотя пример несколько преувеличен (для обеспечения более простых чисел), малое окно может привести к проблемам при соединениях через спутниковые связи с большой задержкой.

Теперь рассмотрим, что происходит с высокоскоростными соединениями. Например, если полоса пропускания и скорость пересылки измеряются 10 млн. бит в секунду, но время цикла составляет 100 мс (1/10 секунды), то для непрерывного потока приемное окно должно хранить, по крайней мере, 1 000 000 бит, т.е. 125 000 байт. Но наибольшее число, которое можно записать в поле заголовка для приемного окна TCP, равно 65 536.

Другая проблема возникает при высоких скоростях обмена, поскольку порядковые номера очень быстро закончатся. Если по соединению можно будет пересылать данные со скоростью 4 Гбайт/с, то порядковые номера должны обновляться в течение каждой секунды. Не будет возможности различить старые дублирующие датаграммы, которые были отсрочены более чем на секунду при перемещении по Интернету, от свежих новых данных.

Сейчас активно проводятся новые исследования для улучшения TCP/IP и устранения упомянутых выше препятствий.

10.16 Функции TCP

Данная глава посвящена многочисленным функциям TCP. Ниже перечислены основные из них:

- Связывание портов с соединениями
- Инициализация соединений посредством трехшагового подтверждения
- Выполнение медленного старта, исключающего перегрузку сети
- Сегментация данных при пересылке
- Нумерация данных
- Обработка поступающих дублированных сегментов
- Вычисление контрольных сумм
- Регулирование потока данных через приемное окно и окно отправки
- Завершение соединения установленным способом
- Прерывание соединения
- Пересылка срочных данных
- Положительное подтверждение повторной пересылки
- Вычисление тайм-аута повторной пересылки
- Снижение обратного трафика при перегрузке сети
- Сигнализация поступления сегментов не по порядку
- Зондирование закрытия приемного окна

10.17 Состояния TCP

Соединение TCP проходит несколько стадий: устанавливается соединение посредством обмена сообщениями, затем пересылаются данные, а далее соединение закрывается с помощью обмена специальными сообщениями. Каждый шаг в работе соединения соответствует определенному *состоянию* этого соединения. Программное обеспечение TCP на каждом конце соединения постоянно отслеживает текущее состояние другой стороны соединения.

Ниже мы кратко рассмотрим типичную смену состояний сервера и клиента, расположенных на разных концах соединения. Мы не ставим целью дать исчерпывающее описание всех возможных состояний при пересылке данных. Оно приведено в RFC 793 и документе *Host Requirements*.

Во время установки соединений сервер и клиент проходят схожие последовательности состояний. Состояния сервера показаны в таблице 10.3, а состояния клиента — в таблице 10.4.

Таблица 10.3

Последовательность состояний сервера

Состояние сервера	Событие	Описание
CLOSED (закрыто)		Фиктивное состояние перед началом установки соединения.
	Пассивное открытие серверным приложением.	
LISTEN (отслеживание)		Сервер ожидает соединения с клиентом.
	Сервер TCP получает SYN и посылает SYN/ACK.	Сервер получил SYN и послал SYN/ACK. Переходит к ожиданию ACK.
SYN-RECEIVED		
	Сервер TCP получает ACK.	
ESTABLISHED (установлено)		Получен ACK, открыто соединение.

Таблица 10.4

Последовательность состояний клиента

Состояние клиента	Событие	Описание
CLOSED		Фиктивное состояние перед началом соединения.
	Клиентское приложение запрашивает соединение. Клиент TCP посылает SYN.	
SYN-SENT		Клиент TCP послал SYN серверу.
	Клиент TCP получает SYN/ACK и посылает ACK. Клиент получил SYN/ACK от сервера и отправил обратно ACK.	
ESTABLISHED (установлено)		Можно перейти к пересылке данных.

Если бы партнеры одновременно пытались установить соединение друг с другом (что случается крайне редко), каждый прошел бы через состояния CLOSED, SYN-SENT, SYN-RECEIVED и ESTABLISHED.

Конечные стороны соединения остаются в состоянии ESTABLISHED, пока одна из сторон не приступит к *закрытию* соединения, послав сегмент FIN. В процессе обычного закрытия сторона, инициирующая это закрытие, проходит через состояния, показанные в таблице 10.5. Ее партнер проходит через состояния, представленные в таблице 10.6.

Таблица 10.5

Последовательность состояний стороны, закрывающей соединение

Состояния закрывающей стороны	Событие	Описание
ESTABLISHED	Локальное приложение запрашивает закрытие соединения.	
	TCP посылает FIN/ACK.	
FIN-WAIT-1		Закрывающая сторона ожидает ответа партнера. Напомним, что от партнера все еще могут прибывать новые данные.
	TCP получает ACK.	
FIN-WAIT-2		Закрывающая сторона получила ACK от партнера, но еще не пришел FIN. Закрывающая сторона ожидает FIN, принимая поступающие данные.
	TCP получает FIN/ACK. Посылает ACK.	
TIME-WAIT		Соединение поддерживается в неопределенном состоянии, чтобы позволить прибыть или отбросить все еще существующие в сети дублированные данные или дублированный FIN. Период ожидания вдвое больше оценки максимального времени жизни сегмента.
CLOSED		Удалена вся информация о соединении.

Таблица 10.6

Последовательность состояний партнера по закрытию соединения

Состояние партнера	Событие	Описание
ESTABLISHED	TCP получает FIN/ACK.	
CLOSE-WAIT		Прибыл FIN.
	TCP посылает ACK.	
		TCP ожидает от своего приложения закрытия соединения. В этот момент приложение может посылать достаточно большое количество данных.
	Локальное приложение инициализирует закрытие соединения.	
	TCP посылает FIN/ACK.	
LAST-ACK		TCP ожидает конечный ACK.
	TCP получает ACK.	
CLOSED		Удалена вся информация о соединении.

10.17.1 Анализ состояний соединения TCP

Команда *netstat -an* позволяет проверить текущее состояние соединения. Ниже показаны соединения в состояниях *listen*, *startup*, *established*, *closing* и *time-wait*.

Отметим, что номер порта соединения указан в конце каждого локального и внешнего адреса. Видно, что имеется трафик TCP как для входной, так и для выходной очередей.

```
> netstat -an
Active Internet connections
Pro  Recv-Q  Send-Q   Local Address          Foreign Address        (state)
t
Tcp    0        0      128.121.50.145.25      128.252.223.5.1526     SYN_RCVD
Tcp    0        0      128.121.50.145.25      148.79.160.65.3368     ESTABLISHED
Tcp    0        0      127.0.0.1.1339         127.0.0.1.111          TIME_WAIT
Tcp    0      438      128.121.50.145.23      130.132.57.246.2219    ESTABLISHED
Tcp    0        0      128.121.50.145.25      192.5.5.1.4022         TIME_WAIT
Tcp    0        0      128.121.50.145.25      141.218.1.100.3968     TIME_WAIT
Tcp    0      848      128.121.50.145.23      192.67.236.10.1050     ESTABLISHED
Tcp    0        0      128.121.50.145.1082    128.121.50.141.6000    ESTABLISHED
Tcp    0        0      128.121.50.145.1022    128.121.50.141.1017    ESTABLISHED
Tcp    0        0      128.121.50.145.514     128.121.50.141.1020    CLOSE_WAIT
Tcp    0     1152      128.121.50.145.119     192.67.239.23.3572     ESTABLISHED
Tcp    0        0      128.121.50.145.1070    192.41.171.5.119       TIME_WAIT
Tcp  579     4096      128.121.50.145.119     204.143.19.30.1884     ESTABLISHED
Tcp    0        0      128.121.50.145.119     192.67.243.13.3704     ESTABLISHED
Tcp    0       53      128.121.50.145.119     192.67.236.218.2018    FIN_WAIT_1
Tcp    0        0      128.121.50.145.119     192.67.239.14.1545     ESTABLISHED
Tcp    0        0      *.19                   *.*                    LISTEN
Tcp    0        0      *.13                   *.*                    LISTEN
Tcp    0        0      *.9                    *.*                    LISTEN
Tcp    0        0      *.7                    *.*                    LISTEN
Tcp    0        0      *.37                   *.*                    LISTEN
```

10.18 Замечания о реализациях

С самого начала протокол TCP предназначен для взаимодействия сетевого оборудования от различных производителей. Спецификация TCP не указывает точно, как должны работать внутренние структуры реализации. Эти вопросы оставлены для разработчиков, которые призваны найти наилучшие механизмы для каждой конкретной реализации.

Даже RFC 1122 (документ *Host Requirements* — требования к хостам) оставляет достаточную свободу для вариаций. Каждая из реализуемых функций маркируется определенным уровнем совместимости:

- MUST (Необходимо)
- SHOULD (Рекомендовано)
- MAY (Разрешено)
- SHOULD NOT (Не рекомендовано)
- MUST NOT (Не нужно)

К сожалению, иногда встречаются продукты, не реализующие требования MUST. В результате пользователи испытывают неудобства от снижения производительности.

Некоторые хорошие методы реализации не учитываются в стандартах. Например, улучшение безопасности возможно при ограничении использования общеизвестных портов привилегированными процессами системы, если в локальной операционной системе поддерживается этот метод. С целью увеличения производительности в реализациях должно быть как можно меньше копирования и перемещения посланных или извлеченных данных.

Стандартный прикладной интерфейс программирования *не определен* (как и политика безопасности), чтобы осталось свободное поле деятельности для экспериментирования с разными комплектами программных инструментов. Однако это может привести к использованию различных программных интерфейсов на каждой из платформ и не позволит перемещать прикладное программное обеспечение между платформами.

Фактически разработчики основывают свои комплекты инструментов на программном интерфейсе Socket, заимствованном из Berkeley. Значение программного интерфейса возросло с появлением WINSock (Windows Socket), что привело к быстрому увеличению новых приложений для настольных систем, которые могли работать поверх любого интерфейса WINSock, совместимого со стеком TCP/IP.

10.19 Дополнительная литература

Оригинал стандарта TCP определен в RFC 793. Модернизации, исправления, и требования совместимости рассмотрены в RFC 1122. Керн (Karn) и Партридж (Partridge) опубликовали статью *Improving Round-Trip Estimates in Reliable Transport Protocols* в журнале *Proceedings of the ACM SIGCOMM 1987.* Статья Джекобсона (Jacobson) *Congestion Avoidance and Control* появилась в *Proceedings of the ACM SIGCOMM 1988 Workshop.* Джекобсон издал также несколько RFC, пересматривающих алгоритмы повышения производительности.

Глава
11

Конфигурация с помощью BOOTP и DHCP

11.1 Введение

Наиболее заметным явлением в компьютерной области, произошедшим в последние несколько лет, является распространение сетей TCP/IP на настольные системы. Необходимая для этого инфраструктура — маршрутизаторы, мосты, коммутаторы и концентраторы — увеличилась до соответствующего такому расширению количества.

Обслуживающий персонал столкнулся с проблемами постоянного подключения и перемещения новых устройств, а также с необходимостью изменения сетевой конфигурации для соответствия современным требованиям к сетям. Все это привело к необходимости создания механизма для автоматизации конфигурации сетевых узлов, распределенных операционных систем и сетевого программного обеспечения. Наиболее эффективным способом реализации такого механизма может быть сохранение конфигурационных параметров и образов программного обеспечения на одном или нескольких серверах *загрузки* (boot server). Во время запуска система взаимодействует с таким сервером, получает от него начальные параметры конфигурации и при необходимости загружает с него нужное программное обеспечение.

В этой главе мы познакомимся с двумя протоколами. Первым был создан протокол Bootstrap Protocol (*BOOTP*), обеспечивающий присваивание IP-адресов по таблице соответствия между физическими и IP-адресами. Администратор должен вручную создать такую таблицу на сервере BOOTP. Усовершенствованная версия BOOTP названа протоколом динамической конфигурации хостов (*Dynamic Host Configuration Protocol* — DHCP). DHCP позволяет полностью автоматизировать присваивание IP-адресов и обладает другими полезными свойствами.

11.2 Требования протокола BOOTP

Некоторым компьютерам для запуска требуется небольшое число конфигурационных параметров, другим — длинный подробный список значений множества таких параметров. Некоторым операционным системам, например настольным сетевым станциям, хостам Unix, требуется полная загрузка всего образа программного обеспечения. Системам, подобным маршрутизаторам, мостам, коммутаторам или концентраторам, требуется как получение конфигурационных параметров, так и загрузка необходимого программного обеспечения.

Протокол конфигурирования должен быть гибким и надежным. В зависимости от размера сети, топологии и выдвигаемых требований, может быть полезна централизация загрузочной информации на одном сервере, распределение ее по нескольким серверам или выполнение реплицирования такой информации.

11.3 Возможности BOOTP

BOOTP был первым стандартом по автоматизации загрузки в окружении TCP/IP. После того как протокол прошел несколько этапов улучшения, он стал способен предоставлять системам все базовые конфигурационные параметры, а также несколько специализированных. Кроме того, BOOTP разрешает системам проводить поиск размещения необходимого программного обеспечения для загрузки.

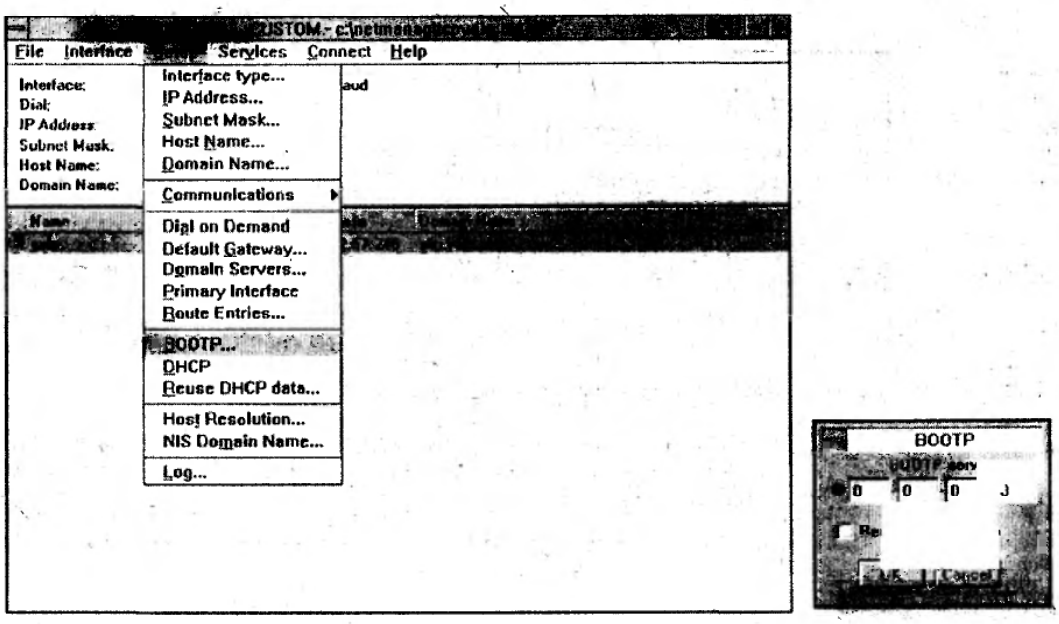

Рис. 11.1. Конфигурирование BOOTP на настольном клиентском компьютере

Конфигурировать клиента BOOTP или DHCP очень просто. На рис. 11.1 показан выбор протокола в меню установки параметров программы *Chameleon*. Раскрывающееся окно разрешает пользователю указать адрес сервера BOOTP (если он известен). Если же адрес не введен, запрос на загрузку будет отправлен в широковещательной рассылке.

11.4 Необходимость DHCP

Область использования BOOTP ограничивает действия администратора, которому необходимо - *автоматизировать* конфигурирование IP-адресов и не вводить вручную длинные списки аппаратных адресов вместе с соответствующими им IP-адресами. Администратору требуется защита от *случайного изменения* при конфигурировании IP-адресов, чтобы пользователь мог спокойно отключить систему и, перенеся ее на другое место сети, получить для системы правильные конфигурационные данные, а следовательно, быстро и без проблем запустить систему на новом месте.

DHCP расширяет возможности BOOTP и устраняет некоторые неопределенности, возникающие при использовании BOOTP и приводящие к неоптимальному взаимодействию в сети.

11.5 Первая версия BOOTP

Первоначально BOOTP разрабатывался для бездисковых рабочих станций. Современные условия привели к необходимости автоматизации загрузки систем, имеющих в ПЗУ (постоянном запоминающем устройстве, которое сохраняет информацию даже после отключения компьютера от сети.— *Прим. пер.*) только базовые средства для IP, UDP и TFTP. Исходный сценарий загрузки (см. рис. 11.2) выглядел следующим образом:

- Клиент отправляет в широковещательной рассылке сообщение UDP на загрузочную информацию.
- Сервер возвращает клиенту его IP-адрес и, при необходимости, местоположение файла загрузки.
- С помощью *простейшего протокола пересылки файлов* (Trivial File Transfer Protocol — TFTP) клиент загружает в собственную память необходимое программное обеспечение и начинает работу.

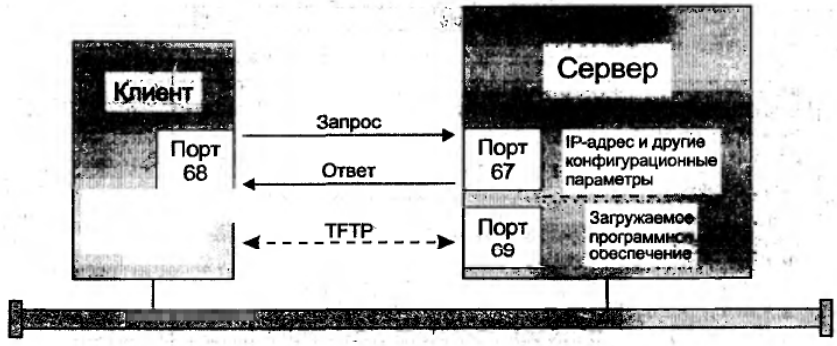

Рис. 11.2. *Локальное взаимодействие между сервером загрузки и клиентом*

Администраторы быстро поняли, что лучше использовать BOOTP для загрузки большего объема конфигурационных данных и настраивать по этому протоколу системы с собственными жесткими дисками (которым не требуется загрузка программного обеспечения).

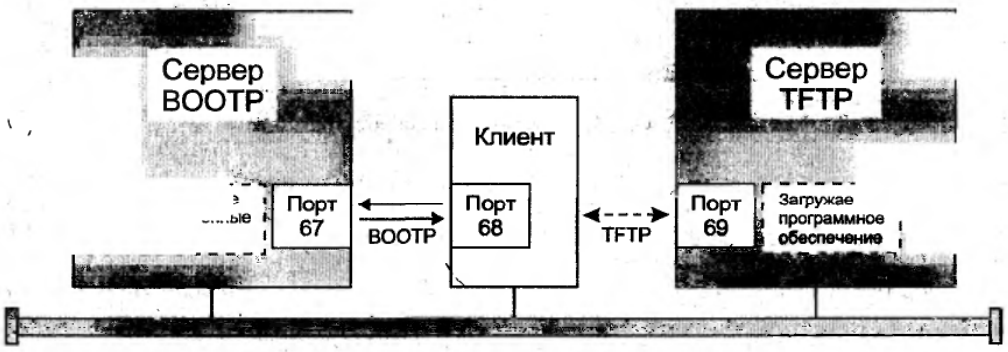

Рис. 11.3. *Использование отдельных серверов для загрузки параметров и программного обеспечения*

Системам, которым требуется TFTP для загрузки программного обеспечения, удобнее использовать один сервер для параметров BOOTP, а другой (или несколько) — для загрузки программного обеспечения (см. рис. 11.3). Например, программное обеспечение операционной системы лучше получать с сервера с тем же типом операционной системы, что и у клиента.

11.6 Эволюция BOOTP

Протокол BOOTP обеспечивает в работе достаточную гибкость:

- Перед запуском клиент может не иметь никакой информации или быть частично сконфигурированным.
- Клиент может получить информацию на сервере загрузки или выбрать для этого специально указанный сервер.
- Клиент может не загружать программное обеспечение, загрузить его по умолчанию или загрузить определенный файл.

Прошло немного времени, и в сообщения BOOTP потребовалось включить дополнительные параметры — маску подсети, адрес маршрутизатора по умолчанию, адреса DNS и другую информацию.

Список дополнительных параметров постоянно увеличивался (полный список параметров BOOTP на момент выхода книги приведен в таблице 11.1), и скоро даже их часть уже не могла разместиться в сообщении UDP реалистичного размера. Для решения этой проблемы избыточные значения поместили в конфигурационном файле, который должен загружаться клиентом через TFTP. Идентификатор этого файла находится в основном сообщении UDP.

Таблица 11.1

Параметры BOOTP и DHCP

Конфигурационные параметры IP	
Subnet mask (маска подсети)	
Time Offset Difference (различия в смещениях времени)	Разность в секундах между местным и универсальным временем (Coordinated Universal Time — UTC)
Client Host Name (имя хоста клиента)	С использованием или без применения локального имени домена
Domain Name (имя домена)	Используется для разрешения имен хостов
Enable/Disable IP Forwarding (разрешение/запрещение перенаправления в IP)	Указывает на маршрутизацию системой датаграмм
Enable/Disable Non-Local Source Routing (разрешение/запрещение нелокальной маршрутизации от источника)	Указывает на перенаправление датаграмм с маршрутизацией от источника
Policy Filter (фильтр политики безопасности)	Список IP-адресов и масок подсети для фильтрации маршрутов, поступающих от источника
Maximum Datagram Reassembly Size (максимальный размер перепостроения датаграмм)	Максимальный размер датаграммы, которую клиент должен подготовить для повторного создания
Default IP Time-to-Live (время жизни по умолчанию для IP)	Начальное значение для поля TTL
Списки IP-адресов	
Routers (маршрутизаторы)	
Time Servers (серверы времени)	
IEN 116 Name Servers (серверы имен по спецификации IEN 116)	(Устарело)
Domain Name Servers (серверы имен доменов)	
Log Servers (серверы регистрации)	
Cookie Servers (серверы цитат)	Ежедневно обновляемые цитаты от сервера Fortune Cookie.
LPR Servers (серверы построчной печати)	Серверы построчных принтеров
Imagen Impress Print Servers (серверы литерной печати)	
Resource Location Servers (серверы размещения ресурсов)	Серверы по спецификации RFC 887
Различные параметры	
Boot File Size (размер файла загрузки)	Размер файла загрузки в 512-октетных блоках
Dump File (файл дампа)	Путь к файлу дампа образа ядра операционной системы, создаваемого при крахе клиента
Swap Server (сервер свопинга)	IP-адрес сервера для свопинга диска
Root Path (корневой путь)	Путь к корневому диску клиента
Extensions Path (расширенный путь)	Путь к файлу с загружаемыми через TFTP конфигурационными параметрами

Таблица 11.1 (*продолжение*)

Параметры BOOTP и DHCP

Параметры Maximum Transmission Unit (MTU)	
Path MTU Aging Timeout (тайм-аут по возрасту пути MTU)	Тайм-аут для возобновления исследования пути MTU
Path MTU Plateau Table (стабилизационная таблица значений для пути MTU)	Серия значений для размера MTU, используемая в исследовании пути MTU, когда маршрутизатор не может предоставить в сообщении ICMP допустимое значение
Параметры IP для интерфейса	
Interface MTU (интерфейс MTU)	Наибольший размер датаграммы, способной пройти через интерфейс
All Subnets Are Local (все подсети локальные)	Указывает, поддерживается ли всеми подсетями тот же самый MTU, что и для локальной подсети
Broadcast Address for the Interface (широковещательный адрес интерфейса)	
Perform Mask Discovery (выполнение поиска маски)	Указывает на использование клиентом ICMP для получения маски подсети
Mask Supplier (система, предоставляющая маску)	Указывает, должен ли клиент отвечать на запросы ICMP при исследовании маски подсети
Perform Router Discovery (выполнение поиска маршрутизаторов)	Указывает на использование клиентом процедуры Router Discovery
Router Solicitation Address (адрес для ходатайства к маршрутизатору)	Адрес, на который клиент должен пересылать запросы ходатайства к маршрутизатору)
Static Routes (статические маршруты)	Список статических маршрутов (пары назначение/маршрутизатор) для таблицы маршрутизации клиента
Параметры уровня связи данных для интерфейса	
Trailer Encapsulation (инкапсуляция заключительной части)	Применяется при согласовании (уже устарело) заключительных частей сообщений ARP
ARP Cache Timeout (тайм-аут кеша ARP)	Тайм-аут для обновления таблицы ARP
Ethernet Encapsulation (инкапсуляция Ethernet)	Ethernet версии 2 (DIX) или IEEE 802.3
Параметры TCP	
TCP Default TTL (значение времени жизни по умолчанию для TCP)	Время жизни (Time-To-Live) для пересылки сегментов TCP
TCP Keep-Alive Interval (интервал поддержания сеанса TCP)	Тайм-аут для проверки сообщениями Keep-Alive внешне неактивных сеансов TCP. 0 означает отмену отправки таких сообщений, пока это не будет затребовано приложением.
Send TCP Keep-Alive Garbage Octet (отправка случайного октета при поддержании сеанса TCP)	Включает в сообщения Keep-Alive ненужный октет

Таблица 11.1 (*продолжение*)

Параметры BOOTP и DHCP

Параметры приложений и служб	
NIS Domain (домен сетевой информационной службы)	Имя домена Network Information Service (когда запущена служба базы данных NIS)
Network Information Server (NIS) Addresses (адреса NIS)	
Network Time Protocol Server Addresses (адреса сервера для протокола сетевого времени)	
Vendor Specific Information (область для разработчиков)	Разработчик указывается через идентификатор класса
List of NetBIOS over TCP/IP Name Servers (список имен серверов NetBIOS, работающих поверх TCP/IP)	
NetBIOS over TCP/IP Datagram Distribution Server (серверы распространения данных NetBIOS датаграммами TCP/IP)	
NetBIOS over TCP/IP Node Type (тип узла, работающего в режиме NetBIOS поверх TCP/IP)	
NetBIOS over TCP/IP Scope (уровень вложенности режима NetBIOS поверх TCP/IP)	
X Window System Font Server (сервер системных шрифтов для X Window)	Список IP-адресов
X Window System Display Managers (диспетчер монитора системы X Window)	Список IP-адресов

11.7 Протокол BOOTP

Рассмотрим более подробно протокол начальной загрузки (Bootstrap Protocol — BOOTP). Он является простейшим приложением для запроса/ответа по протоколу UDP.

- Клиент отсылает сообщение *запроса на загрузку* (bootrequest) из порта 68 на порт 67 сервера.
- Сервер реагирует на это сообщением *ответа на загрузку* (bootreply), отсылая его на порт 68 клиента.

Поскольку UDP не обеспечивает надежного обмена, клиенту может потребоваться отправить запрос повторно, если ответ не будет получен в течение тайм-аута.

11.7.1 Формат сообщения BOOTP

Для *запроса* и *ответа загрузки* используется одинаковый формат сообщения. В запросе некоторые поля имеют нулевые значения. Формат сообщения показан на рис. 11.4.

Поля, которые должны быть заполнены в *запросе загрузки*, выделены полужирным шрифтом. Поля, которые могут быть *указаны* клиентом, набраны курсивом (подробные сведения о полях сообщения и соответствующих параметрах описаны в разделе 11.13).

Один октет	Один октет	Один октет	Один октет
1 = запрос 2 = ответ	Тип оборудования	Длина аппаратного адреса	**Счетчик попаданий** (начальное значение 0)
Идентификатор транзакции (одинаков для запроса и ответа)			
Счетчик секунд от момента отправки клиентом первого запроса (начальное значение 0)		Поле флагов (*флаг широковещательной рассылки*)	
IP-адрес клиента (если он известен клиенту, иначе — 0)			
В ответе: IP-адрес, предоставленный клиенту сервером			
В ответе: IP-адрес сервера (обеспечивающий доступ к загружаемому конфигурационному файлу)			
IP-адрес промежуточного маршрутизатора			
Аппаратный адрес клиента			
Имя хоста сервера (клиент может при необходимости указать нужный ему сервер)			
Имя конфигурационного файла (клиент может указать базовое имя — Windows или SunOS, а сервер обеспечивает реальный путь к файлу)			
Область для разработчиков Дополнительные параметры			

Рис. 11.4. Формат *запроса* и *ответа* сообщения *загрузки*

11.7.2 Доставка запроса от клиента на сервер

Клиент не имеет сведений об адресе для направления запроса и отправляет его с IP-адресом источника 0.0.0.0 и IP-адресом приемника 255.255.255.255.

Сервер (или серверы) в одной с клиентом локальной сети услышит посланный запрос. Если клиент заполнил в сообщении "поле" *имя хоста сервера*, ответит только указанный в этом поле сервер (см. рис. 11.5). Если же имя не указано, ответить могут несколько серверов.

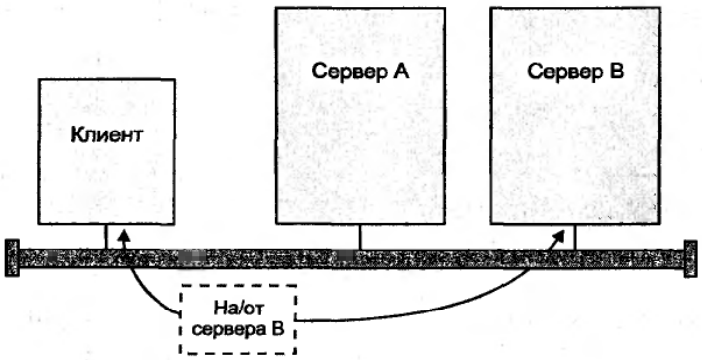

Рис. 11.5. Выбор заданного сервера

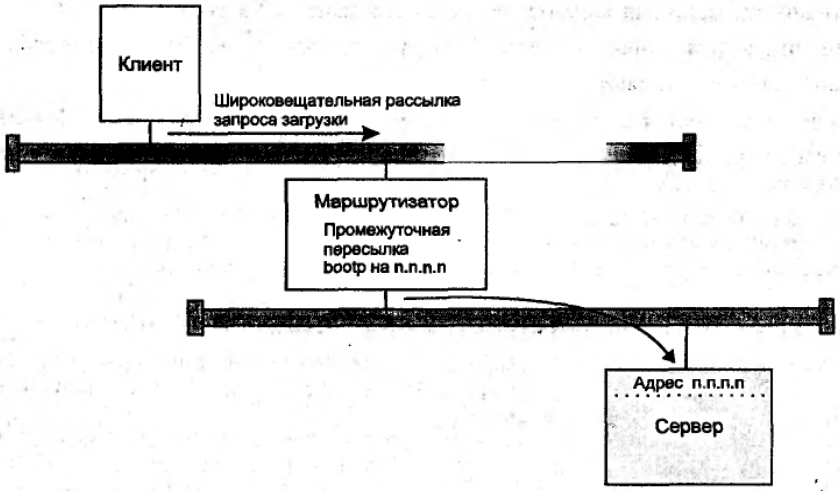

Рис. 11.6. Промежуточная пересылка запроса загрузки на удаленный сервер

11.7.3 Использование промежуточного агента

Гораздо удобнее использовать один или несколько централизованных серверов загрузки, чем размещать такие серверы в каждой из локальных сетей. Однако как широковещательный запрос от клиента может достигнуть удаленного сервера по локальной сети? Для этого используется специальная система, помогающая переслать такой запрос (см. рис. 11.6).

Промежуточный агент (relay agent) помогает системе отправить локальный запрос BOOTP на удаленный сервер. В качестве промежуточных агентов используются маршрутизаторы (хотя стандарты позволяют работать в этом режиме и хостам).

Обычно маршрутизатор конфигурируется с IP-адресом (адресами) для пересылки запросов загрузки на один или несколько серверов (так делается в удачных реализациях, хотя стандарты допускают широковещательные рассылки запросов загрузки по указанным соединениям для поиска сервера загрузки, когда его IP-адрес неизвестен, и именно для этого в сообщении загрузки имеется поле *счетчика попаданий*, препятствующее зацикливанию). Когда промежуточный агент получает от клиента запрос на загрузку, то:

- *Промежуточный маршрутизатор* запроса BOOTP проверяет поле. Если оно равно нулю, промежуточный агент вставляет в него IP-адрес интерфейса, по которому и был получен данный запрос. Сервер BOOTP использует этот адрес для возврата ответа загрузки клиенту через промежуточный агент.
- Затем агент пересылает запрос клиента на один или несколько предварительно указанных адресов серверов.

11.7.4 Присваивание IP-адресов

Администратор конфигурирует сервер BOOTP для присваивания системам IP-адресов посредством ручного создания таблицы отображения на IP-адрес комбинации типа оборудования и аппаратного адреса клиента. Кодирование типов оборудования определяется документом *Assigned Numbers*. Например, для Ethernet код типа оборудования = 1. Таблица должна выглядеть как:

Тип оборудования	Аппаратный адрес	IP-адрес
1	02 60 8C 12 14 AA	128.121.2.5
1	08 00 20 D3 20 14	128.121.2.19

Многие реализации включают в таблицу дополнительные столбцы, идентифицирующие имена хостов для удобочитаемости таблицы.

Простейший сценарий для клиента, не знающего своего IP-адреса:

- Клиент отправляет в широковещательной рассылке запрос (на порт 67 сервера).
- Сервер получает этот запрос.
- По типу оборудования и аппаратному адресу клиента сервер выбирает в таблице IP-адрес.
- Если клиент расположен локально, сервер отправляет ответ в широковещательной рассылке (на порт 68 клиента).
- Если клиент удален от сервера, ответ посылается на порт 67 по адресу, указанному в поле *IP-адреса промежуточного агента*. Затем промежуточный агент пересылает его в локальной широковещательной рассылке на клиентскую систему.

11.7.5 Загрузка клиента, знающего собственный IP-адрес

Предположим, что клиент уже предварительно сконфигурирован на определенный IP-адрес или сохранил IP-адрес, присвоенный ему во время прошлой загрузки. В этом случае в поле *IP-адреса клиента* указывается уже присвоенный ему адрес.

В соответствии с первоначальным стандартом BOOTP сервер может позволить клиенту сохранить старый адрес и будет использовать его в заголовке IP для доставки клиенту ответа. Отметим, что, если после предыдущей загрузки компьютер был физически перемещен в иную подсеть или иную локальную сеть, клиент не сможет получить ответ, поскольку в нем будет использован адрес старой подсети.

Решить эту проблему поможет использование протокола DHCP.

11.7.6 Конфигурирование загрузки программного обеспечения

Первоначально предполагалось размещение на одном сервере как конфигурационных данных, так и загружаемого через TFTP программного обеспечения. Однако разделить эти службы очень просто. Сервер конфигурации BOOTP может просто указать IP-адрес хоста для сервера TFTP, а также путь к загружаемому файлу.

Как же сервером BOOTP выбираются сервер TFTP и загружаемый файл, если загружаемое программное обеспечение может быть распределено по нескольким физическим серверам?

Сервер BOOTP можно сконфигурировать с таблицей отображения коротких имен систем в IP-адреса серверов TFTP, содержащих загружаемые файлы, с указанием пути для каждого файла. Например:

Короткое имя	IP-адрес сервера TFTP	Путь к файлу
SunOS	128.121.50.2	/bin/vmunix

Клиент BOOTP посылает соответствующее короткое имя на сервер в поле *имени загружаемого файла* (вместо этого в DHCP используется отдельное поле *идентификатора класса*). По короткому имени сервер выбирает из таблицы нужные сведения и помещает полный путь к файлу в поле *имени загружаемого файла*, а IP-адрес сервера TFTP записывает в поле *"IP-адрес сервера"* сообщения.

Клиент отсылает сообщение с нулевым значением в поле *имени загружаемого файла*, когда нет необходимости в файле загрузки или когда можно использовать предопределенные значения по умолчанию.

11.7.7 Область для разработчиков

Первоначально *область для разработчиков* (vendor specific area) использовалась в сообщениях для пересылки сведений, специфичных для конкретной реализации. Однако в начале применения BOOTP эта область оставалась свободной, хотя большой объем информации (например, маска подсети или адрес маршрутизатора по умолчанию) формально не включался в сообщения. *Область для разработчиков* служила для размещения дополнительных конфигурационных параметров, а также сведений, специфичных для разработчика. В этой области определено достаточно много различных полей.

11.7.8 Ответ безадресному клиенту

Если клиент не заполнил поле предварительно заданного IP-адреса, такой адрес присваивается сервером. Простейшим способом возврата ответа клиенту в этом случае будет такой:

- Применение заголовка IP с новым IP-адресом в качестве адреса назначения
- Заключение датаграммы в кадр, адресованный на физический адрес клиента

Однако некоторые старые клиенты неспособны принимать датаграммы IP с явно указанным IP-адресом, пока не будут сконфигурированы на этот адрес. Данная проблема называется "яйцо или курица" (что было раньше? — *Прим. пер.*).

Такие клиенты могут принимать датаграммы на порт назначения 68 и с широковещательным IP-адресом 255.255.255.255. Новые клиенты BOOTP предпочитают прием ответа по широковещательному IP-адресу посредством установки в 1 *флага широковещательной рассылки* (находится в поле флагов) при отправке своего запроса.

11.7.9 Счетчик секунд

Когда клиент отсылает первый запрос на загрузку данных, поле *счетчика секунд* имеет нулевое значение. Если на запрос не приходит ответа, по завершении тайм-аута клиент снова отправляет запрос, изменяя значение в поле счетчика секунд. Для тайм-аута клиент использует случайный интервал, увеличивающийся до значения 60 с.

Данное поле не имеет специального назначения. Его содержимое может проверять сервер или сетевой монитор для определения длительности ожидания клиентом загрузки по сети. Сервер может использовать значения из поля счетчика секунд для ранжирования запросов по приоритетам, однако в настоящее время в большинстве реализаций это поле игнорируется.

11.8 Возможности DHCP

DHCP существенно расширяет возможности BOOTP. К наиболее значимым изменениям относятся:

- Упрощение администрирования
- Автоматизация конфигурирования
- Поддержка перемещений или изменений сети
- Возможность запроса клиентом значений определенных параметров
- Новые типы сообщений DHCP, обеспечивающие более надежное соединение между клиентом и сервером

Еще одним примечательным свойством является возможность клиента BOOTP получить доступ к серверу DHCP, т.е. обеспечивается обратная совместимость.

11.8.1 Администрирование и автоматизация конфигурирования

DHCP позволяет существенно снизить объем администрирования для конфигурирования системы. При необходимости можно просто указать блок IP-адресов, из которого сервер DHCP будет присваивать адреса клиентам в локальной сети. Это легко может сделать администратор системы, освободив время для ввода данных о других критических параметрах (например, маски подсети, адресов DNS или адресов маршрутизаторов по умолчанию для данной локальной сети). Если необходимо, администратор может указать дополнительные конфигурационные параметры.

Протокол позволяет серверу автоматически присвоить клиенту IP-адрес из выделенного блока и послать ему заданный набор параметров.

В DHCP наследуется способность BOOTP предоставлять подробные или только определенные конфигурационные данные, а также идентификация загружаемого образа программного обеспечения. Недостатком BOOTP было то, что клиент не мог управлять получаемым им набором параметров. DHCP позволяет клиенту запрашивать только заданный набор таких параметров.

11.8.2 Перемещения и изменения

Что произойдет, если пользователь переместит компьютер в другое место, подключив его к иной сети или подсети? Во время загрузки компьютер, использующий DHCP, автоматически изменит свой IP-адрес и маску подсети, а также при необходимости — маршрутизатор по умолчанию и DNS.

Без DHCP все эти изменения приходилось выполнять вручную.

11.9 Механизмы DHCP

11.9.1 Присваивание IP-адресов

В DHCP поддерживаются три типа *присвоения* адресов:

- *Ручное*, когда IP-адрес вводится на сервере и назначается клиенту постоянно
- *Автоматическое*, когда IP-адрес выбирается сервером из пула доступных адресов и назначается клиенту постоянно
- *Динамическое*, когда IP-адрес присваивается клиенту на ограниченный срок или до завершения его использования данным клиентом

Например, пользователь мобильного компьютера может подключаться к сети в случайные моменты времени, и ему нет надобности присваивать постоянный IP-адрес или адрес на длительный срок.

11.9.2 Аренда адресов

Процесс выделения адресов предполагает запрос клиентом IP-адреса на определенный период времени (возможно, что и навсегда). Сервер предоставляет клиенту адрес в *аренду*, указывая период использования данного адреса. Клиент должен периодически обновлять свои права на аренду, иначе через заданный интервал времени он потеряет это право. Потерянный адрес можно использовать повторно (например, для другого клиента.— *Прим. пер.*).

Для продления аренды адреса клиент идентифицирует свои права. При первичном выделении адреса клиенту через поле *идентификатора клиента DHCP* присваивается определенное значение, которое и будет применяться во всех последующих взаимодействиях с сервером. Иначе аренда идентифицируется типом оборудования клиента, аппаратным адресом и присвоенным IP-адресом.

11.9.3 Связывание

Сервер DHCP хранит таблицу соответствия между клиентами и их конфигурационными параметрами. *Связывание* заключается в назначении каждому клиенту IP-адреса и набора конфигурационных параметров.

11.10 Совместимость и различия

Для обеспечения совместимости с BOOTP формат сообщений DHCP идентичен сообщениям BOOTP. В результате:

- Клиент BOOTP может обращаться к серверу DHCP
- Клиент DHCP может использовать службу промежуточных агентов BOOTP

Самым заметным изменением стало переименование *области для разработчика* в поле *Options* (варианты). Добавлены и несколько дополнительных вторичных полей, включая следующие:

- Поле для параметра *DHCP Class Identifier* (идентификатор класса DHCP). Этот параметр клиент отправляет серверу с целью использования в качестве ключа при поиске специфичных для клиента конфигурационных сведений.
- *Клиент идентифицируется* арендой и связыванием (с набором конфигурационных параметров), а не типом оборудования и аппаратным адресом.
- Введен параметр для указания максимального размера сообщения, которое может получить клиент от сервера.

Существенным изменением в протоколе стала способность согласовывать набор конфигурационных параметров. Для этого определено несколько новых типов сообщений. Если в ответе клиент не получил всех нужных ему параметров, DHCP позволит продолжить их запрос от сервера.

11.10.1 Типы сообщений

Тип сообщения DHCP определяется полем *DHCP Message Type* (тип сообщения DHCP). Пользоваться этим может только клиент DHCP, но не клиент BOOTP. Имеются следующие типы сообщений:

DHCPDISCOVER	Клиент посылает сообщение для поиска серверов.
DHCPOFFER	Сервер отвечает клиенту и предоставляет ему IP-адрес и конфигурационные параметры.
DHCPREQUEST	Клиент выбирает один из серверов и посылает запрос. При необходимости можно запросить у сервера дополнительные параметры.
DHCPACK	Сервер отвечает и предоставляет дополнительные параметры, если они были запрошены.
DHCPNAK	Сервер отменяет запрос (например, клиент мог запросить уже используемый IP-адрес). Клиент должен возобновить процесс загрузки с самого начала.
DHCPDECLINE	Клиент отменяет принятые конфигурационные параметры, поскольку один из них был некорректным.
DHCPRELEASE	Клиенту более не нужен IP-адрес, и поэтому он освобождает его.

11.10.2 Типичный начальный обмен сообщениями между клиентом и сервером

Пример успешного начального взаимодействия между клиентом и сервером:

1. Клиент посылает широковещательную рассылку (*DHCPDISCOVER*) для поиска одного или нескольких серверов.
2. Клиенту могут ответить несколько серверов. Он ждет получения одного или нескольких ответов (*DHCPOFFER*). Каждый ответ содержит IP-адрес, маску подсети, дату окончания аренды адреса, признак идентичности клиента серверу (в DHCP — поле *идентификатора сервера DHCP*) и некоторые конфигурационные параметры.
3. На основе полученных ответов клиент выбирает сервер и отправляет запрос по широковещательной рассылке (*DHCPREQUEST*) с указанием в поле *идентификатора сервера* нужной системы. Сообщение клиента может содержать *список запрашиваемых параметров*, который идентифицирует нужные клиенту дополнительные данные конфигурации.
4. Выбранный клиентом сервер сохраняет характеристики связывания для клиента на диске, индексируя их соответствующим ключом. Сервер посылает клиенту параметры в сообщении *DHCPACK*. Клиент должен использовать запрос ARP, чтобы удостовериться в том, что никакое другое устройство не использует назначенный ему IP-адрес.

11.10.3 Возобновление

Клиенты могут продлить аренду адресов посредством быстрого обмена с сервером:

- Предварительно сконфигурированный клиент может посылать *DHCPREQUEST* с указанием в нем своего IP-адреса.
- Сервер (или серверы), хранящий конфигурацию клиента, отвечает сообщением *DHCPACK* (если все будет успешно).
- Если информация от клиента больше не имеет силы (например, рабочая станция пользователя была подключена к другой локальной сети), серверы отвечают сообщением *DHCPNAK*, а клиент должен повторно начать процедуру полной конфигурации.
- Клиент должен снова начать конфигурирование, если в сообщении *DHCPNAK* информация от сервера некорректна.

11.11 Параметры загрузки

Параметры таблицы 11.1 могут содержаться в ответах протоколов BOOTP или DHCP, а параметры таблицы 11.2 могут использоваться только в DHCP.

Таблица 11.2

Параметры DHCP

Дополнительные параметры только для DHCP	
Requested IP Address (запрошенный IP-адрес)	Клиент запросил выделение определенного IP-адреса.
IP Address Lease Time (время аренды IP-адреса)	В запросе клиент указывает необходимое время аренды адреса, реальное значение которого содержится в ответе сервера.
Option Overload (дополнительная нагрузка)	Указывает, что в сообщении, кроме стандартных значений, находятся поля *сервера имен хостов DHCP* или *имя файла загрузки*.
DHCP Message Type (тип сообщения DHCP)	Например, *DISCOVER*, *OFFER* или *REQUEST*.
Server Identifier (идентификатор сервера)	Используется для разделения серверов, чтобы клиент мог выяснить, от какого из них используется аренда.
Parameter Request List (список запрашиваемых параметров)	Список необязательных кодов, разрешающих клиенту запрос значений определенных параметров.
Message (сообщение)	Используется как сообщение об ошибке в ответе сервера (например, о недоступности IP-адресов). Клиент может применить его для указания причины отклонения предложенного ему набора параметров.
Maximum DHCP Message Size (максимальный размер сообщения DHCP)	Максимальный размер сообщения DHCP, которое желает получать клиент.
Renewal (T1) Time Value (значение времени восстановления)	Интервал времени от назначения адреса клиенту до попытки взаимодействия клиента с сервером для продления аренды IP-адреса.
Rebinding (T2 > T1) Time Value (значение времени повторного связывания)	Интервал времени от назначения адреса клиенту до попытки продления аренды IP-адреса у *любого* из серверов, если клиент не может получить ответ от исходного сервера.
Class Identifier (идентификатор класса)	Локально назначенный идентификатор, используемый клиентом для определения своего типа и конфигурации. Некоторые параметры могут быть возвращены на основе указанного класса.
Client Identifier (идентификатор клиента)	Уникальный идентификатор для клиента, включенный в сообщение *DHCPDISCOVER*. Идентификатором может быть имя DNS или другой присвоенный клиенту идентификатор. Используется для связи клиента с данными о его аренде IP-адреса.

Допустимо включать в списки большее число параметров. Текущие требования рассмотрены в последней версии RFC *Assigned Numbers*.

Многие значения представляют собой списки IP-адресов, где адреса должны появляться в порядке предпочтения.

11.12 Другие методы автоматизации конфигурирования

Было предпринято несколько других попыток автоматизировать отдельные части процесса конфигурирования. Подключенные к локальной сети системы могут использовать *обратные ARP* (RARP), чтобы обнаружить свой IP-адрес. Запрос ICMP *Address Mask* (маска адреса ICMP) и ответ на него обеспечивают получение маски подсети. Но нет никакого смысла применять несколько отдельных протоколов и сообщений для получения сведений, предоставляемых в одном ответе BOOTP или DHCP.

Механизм исследования ICMP-маршрутизаторов имеет некоторое преимущество, поскольку предоставляет непрерывно обновляемую информацию о доступных в сети маршрутизаторах.

11.13 Дополнительная литература

Приведенный список литературы действителен на момент написания книги:

- BOOTP определен в RFC 951.
- RFC 1533 рассматривает варианты DHCP и расширения BOOTP для разработчиков.
- RFC 1534 описывает взаимодействие между DHCP и BOOTP.
- RFC 1542 предоставляет разъяснения и описание изменений в BOOTP.
- Протокол DHCP специфицирован в RFC 1541.

Глава
12

DNS

12.1 Введение

Часто конечный пользователь знает имя хоста, но не имеет понятия о его адресе. Но адрес нужно знать для взаимодействия с хостом, поэтому конечному пользователю или запущенному им приложению необходим способ получения адреса по имени хоста.

В небольшой изолированной сети такое преобразование выполняется через единую таблицу, а отдельные системы могут получать самые свежие сведения, копируя содержимое таблицы на свои жесткие диски.

В первые дни существования Интернета использовалась также централизованная таблица. Ее главную копию обслуживал сетевой информационный центр Министерства обороны США (DOD NIC), что позволяло проводить преобразование имен в адреса другим системам, периодически копировавшим содержимое главной таблицы. Однако со временем этот метод стал неэффективным.

12.2 Назначение DNS

Система имен доменов (Domain Name System — DNS) обеспечивает более эффективный способ согласования имен и адресов Интернета. База данных DNS обеспечивает автоматизированную службу преобразования имен в адреса. Эта система успешно работает, и многие организации, даже не подключенные к Интернету, применяют DNS для обслуживания собственных внутренних имен компьютеров.

DNS является *распределенной* базой данных. Имена и адреса Интернета хранятся на множестве серверов по всему миру (ниже мы узнаем о хранении на этих же серверах важных сведений о маршрутизации электронной почты). Организация, владеющая именем домена (например, *yale.edu*), несет ответственность за обслуживание и работу с сервером имен, который выполняет трансляцию имен этого домена в адреса. Локальный обслуживающий персонал быстро отслеживает изменения, добавление или удаление сетевых узлов и согласует эти операции с *первичным сервером* (primary server) домена. Поскольку данные для трансляции имен в адреса очень важны, данная информация реплицируется на один или несколько *вторичных серверов* (secondary server).

12.3 Программное обеспечение BIND

Многие разработчики компьютеров предоставляют бесплатное программное обеспечение для сервера имен. Обычно оно является адаптацией пакета *Berkeley Internet Domain* (BIND) для конкретных условий. Периодически в Интернете появляются новые бесплатные версии пакета BIND.

Организации могут пользоваться бесплатным программным обеспечением для собственных служб трансляции имен в адреса. Но, если предполагается соединение с Интернетом, необходимо обеспечить не менее двух общедоступных серверов имен, которые станут частью единой системы имен доменов Интернета.

12.4 Определители

Клиентская программа для просмотра информации DNS является стандартной частью любого продукта TCP/IP и называется *определителем* (resolver). Обычно определитель работает в фоновом режиме, и пользователь даже не замечает его присутствия. В приведенном ниже примере пользователь

запрашивает соединение по *telnet* с системой *minnie.jvnc.net*. Пользовательское приложение *telnet* запрашивает локальную *программу-определитель* об IP-адресе для указанного сайта:

```
> telnet minnie.jvnc.net
Trying 128.121.50.141 ...
Connected to minnie.jvnc.net.
```

При установке TCP/IP на хосте, используемом для просмотра базы данных имен доменов, в конфигурационную информацию этого хоста нужно включить сведения об IP-адресах одной или нескольких систем DNS. Программа-определитель должна знать адреса DNS, к которым она будет обращаться.

Примером может служить система *tigger*, используемая провайдером *Global Enterprise System* в качестве сервера электронной почты, сетевых новостей и сервера *telnet*. Как и большинство систем Unix, *tigger* имеет конфигурационный файл */etc/resolv.conf*, в котором указаны локальные имена доменов и IP-адреса двух серверов имен доменов для данного домена.

```
> more /etc/resolv.conf
domain jvnc.net
128.121.50.2
128.121.50.7
```

Настольным системам TCP/IP также требуется информация DNS. Как показано на рис. 12.1, программный пакет *Chameleon* для Microsoft Windows предоставляет раскрывающееся меню, в которое вводят сведения об IP-адресах серверов имен доменов.

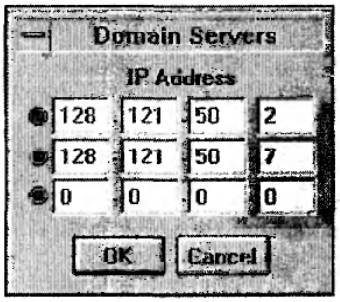

Рис. 12.1. Конфигурирование DNS

12.5 Просмотр адресов хостов

Как мы уже знаем, многие системы предоставляют интерактивные программы-определители, дающие возможность пользователям напрямую обращаться к серверам DNS, посылая к ним запросы и получая ответы. Приведем пример работы с программой-определителем *nslookup* для Unix:

1. Сразу после ввода пользователем имени программы локальный сервер по умолчанию идентифицирует себя, выводя собственное имя и адрес. В нашем примере именем будет *r2d2.jvnc.net*, а адресом — 128.121.50.2.
2. Пользователь вводит имя хоста, адрес которого нужно узнать.
3. Запрос отправляется на сервер.
4. После каждого запроса сервер (*r2d2*) идентифицирует себя и выводит ответ.
5. Если пользователь запрашивает локальную информацию, то сервер извлекает ответ из собственной базы данных.
6. Если пользователю требуются сведения о внешнем хосте, сервер сначала проверяет их наличие в собственном *кеше* (хранящем данные о последних запросах пользователей) и извлекает их (если они есть) либо (если их нет) взаимодействует с удаленным авторитетным сервером для получения ответа из его базы данных.

7. Ответ от удаленного авторитетного сервера сохраняется в дисковом кеше локального сервера для будущего использования и пересылается пользователю, запросившему этот ответ.

Каждый этап диалога с программой разъясняется комментариями в правой части страницы. Отметим, что ответ, извлеченный из кеша сервера, маркируется как *неавторитетный*.

```
>nslookup

Default Server:
R2d2.jvnc.net                          Выводится имя и адрес локального сервера.
Address: 128.121.50.2

>Mickey.jvnc.net.                      Пользователь вводит запрос, ответ на который
                                       находится в локальной базе данных.
Server: r2d2.jvnc.net                  Снова вывод идентификатора и адреса сервера.
Address: 128.121.50.2
Name: mickey.jvnc.net                  Указанное в запросе имя.
Address: 128.121.50.143                Ответ.

>Www.novell.com.                       Пользователь вводит запрос об удаленном хосте.

Server: r2d2.jvnc.net                  Снова вывод идентификатора и адреса сервера.
Address: 128.121.50.2

Name: www.novell.com                   Запрашиваемое имя.
Address: 137.65.2.5                    Ответ сохранялся на диске r2d2 и был выведен
                                       пользователю.

>Www.novell.com.                       Пользователь повторяет запрос об удаленном
                                       хосте.
Server: r2d2.jvnc.net                  Снова вывод идентификатора и адреса сервера.
Address: 128.121.50.2

Non-authoritative answer:              Ответ получен из локального кеша.
Name: www.novell.com                   Запрашиваемое имя.
Address: 137.65.2.5                    Ответ.
```

Для чего сервер постоянно идентифицирует себя? Вспомним, что организацию могут обслуживать два или более серверов, один из которых может оказаться слишком загруженным или выключенным на профилактику. В этом случае определитель не сможет получить ответ от первой в своем списке системы и пошлет запрос к следующей системе из списка. По выводимым в *nslookup* сведениям администратор сможет быстро определить, какой из серверов отвечает на запросы.

Отметим, что в конце каждого запрос стоит символ точки. Ниже в мы рассмотрим причину этого.

12.6 Авторитетные ответы и ответы из кеша

Все данные вводятся и изменяются на *первичном* сервере имен. Они хранятся на собственном жестком диске этого сервера. *Вторичный* сервер имен загружает информацию с первичного сервера.

Когда система посылает запрос к DNS, она не заботится о том, куда попадет запрос — на первичный или на вторичный сервер имен. Все серверы имен (первичные и вторичные) *авторитетны* (authoritative) для своего домена.

Для снижения трафика локальный сервер *кеширует* уже полученные ответы на своем жестком диске. При повторном запросе данные (если они еще находятся в кеше) извлекаются из кеша, формируя локальный ответ.

Как долго информация находится в кеше? Максимальное время хранения конфигурируется авторитетным сервером и сообщается запрашивающей системе вместе с ответом.

12.7 Трансляция адресов в имена

Система DNS обратима, т.е. может выполнять обратную трансляцию адресов в имена. Однако способ, используемый для этого в *nslookup*, несколько необычен:

- Установить тип запроса в *ptr*.
- Записать адрес *наоборот*, дописав в конце его *.in-addr.arpa*.

Например:

```
> set type = ptr
> 143.50.121.128.in-addr.arpa.
Server: r2d2.jvnc.net
Address: 128.121.50.2

143.50.121.128.in-addr.arpa        host name = mickey.jvnc.net
>
```

Эта странность становится осмысленной, если рассмотреть архитектуру глобального обратного просмотра. Организация, владеющая сетевым адресом, несет ответственность за запись в базе данных DNS всех своих трансляций адресов в имена. Это делается в таблице, *иной* чем таблица отображения имен в адреса.

Поддерево специального домена *in-addr.arpa* (см. рис. 12.2) создается для указания на все сетевые таблицы. Когда в это дерево помещается адрес, имеет смысл разместить первое число вверху, а оставшиеся числа сверху вниз. В этом случае все адреса 128.*x.x.x* окажутся ниже узла 128.

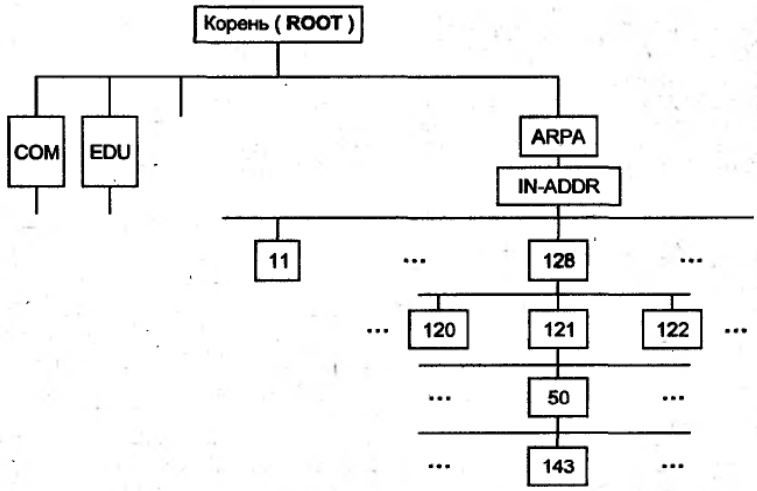

Рис. 12.2. Поддерево домена in-addr.arpa

Если читать метки на дереве с помощью тех же правил, что и для имен (сверху вниз), адреса получатся записанными в обратном порядке — в частности *143.50.121.128.in-addr.arpa*.

Разумеется, пользовательский интерфейс программы *nslookup* мог бы скрыть эту технологию. Но это все же Unix, и на рис. 12.3 показана более дружественная для пользователя программа *NSLookup*, разработанная в Ashmount Research Ltd. Запросы вводятся в небольшом вторичном окне в нижней части общего окна программы, а ответы выводятся в верхнюю область окна. Отметим, что в обоих ответах присутствуют имена и адреса сервера имен, содержащего авторитетные сведения для данного запроса.

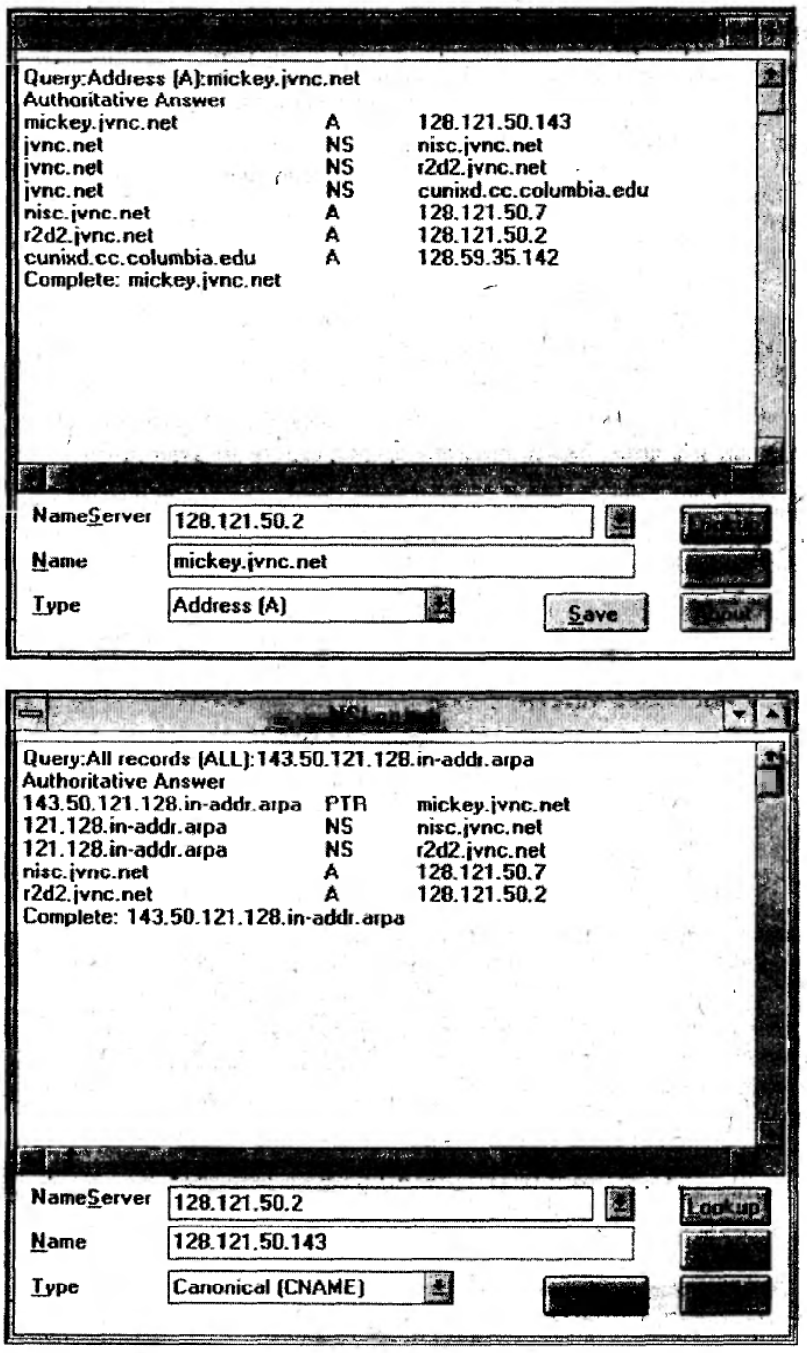

Рис. 12.3. Запрос к DNS

12.8 Локальные и глобальные серверы имен доменов

В изолированной сети TCP/IP можно применять любое бесплатное программное обеспечение DNS для создания первичной базы данных трансляции имен и репликации этой базы данных в определенные точки сети. Все пользовательские запросы будут обрабатываться локальным сервером имен.

Но при соединении сети с Интернетом сервер имен должен быть способен извлекать глобальную информацию. Ключом к пониманию данной операции является то, что, когда организация (например, *microsoft.com*) желает подключиться к Интернету, она обязана оформить сведения о себе в соответствующем комитете *регистрации авторизации* (registration authority), в нашем случае это InterNIC, и указать имена и адреса не менее чем двух серверов DNS. InterNIC добавит эти сведения в свой *корневой* список серверов имен доменов.

Корневой список реплицируется на несколько *корневых серверов*, играющих основную роль в обработке удаленных запросов DNS. Предположим, что запрос на трансляцию имени в адрес для *www.microsoft.com* поступил на локальный сервер имен *trigger*. Тогда:

- Сервер проверяет принадлежность *www.microsoft.com* локальному домену.
- Если имя не принадлежит локальному домену, проверяется его наличие в кеше.
- Если имени нет и в кеше, сервер посылает запрос корневому серверу.
- Корневой сервер возвращает имя и адрес сервера DNS, который содержит сведения о *www.microsoft.com*.

Для просмотра текущего списка корневых серверов следует запустить *nslookup* и указать тип запроса *ns*. Если ввести точку (что означает корень дерева), будут возвращены имена и адреса нескольких корневых серверов.

```
> nslookup
> set type = ns
> .
Server: r2d2.jvnc.net
Address: 128.121.50.2

Non-authoritative answer:
(Root)   nameserver = C.ROOT-SERVERS.NET
(Root)   nameserver = D.ROOT-SERVERS.NET
(Root)   nameserver = E.ROOT-SERVERS.NET
(Root)   nameserver = I.ROOT-SERVERS.NET
(Root)   nameserver = F.ROOT-SERVERS.NET
(Root)   nameserver = G.ROOT-SERVERS.NET
(Root)   nameserver = A.ROOT-SERVERS.NET
(Root)   nameserver = H.ROOT-SERVERS.NET
(Root)   nameserver = B.ROOT-SERVERS.NET

Authoritative answers can be found from:
C.ROOT-SERVERS.NET      internet address = 192.33.4.12
D.ROOT-SERVERS.NET      internet address = 128.8.10.90
E.ROOT-SERVERS.NET      internet address = 192.203.230.10
I.ROOT-SERVERS.NET      internet address = 192.36.148.17
F.ROOT-SERVERS.NET      internet address = 192.5.5.241
F.ROOT-SERVERS.NET      internet address = 39.13.229.241
G.ROOT-SERVERS.NET      internet address = 192.112.36.4
A.ROOT-SERVERS.NET      internet address = 198.41.0.4
H.ROOT-SERVERS.NET      internet address = 128.63.2.53
B.ROOT-SERVERS.NET      internet address = 128.9.0.107
>
```

Корневой сервер обеспечивает прямые ссылки на серверы доменов второго уровня (как *microsoft.com* или *yale.edu*), расположенные ниже доменов *COM, EDU, GOV* и других доменов первого уровня. Примером может служить часть информации о *3com.com.* полученная непосредственно из файла корневого списка (во втором столбце приведено значение тайм-аута в секундах, используемое в данном элементе):

```
3COM.COM.           172800      NS      NS.3COM.COM.
NS.3COM.COM.        172800      A       129.213.128.2
3COM.COM.           172800      NS      TMC.EDU.
TMC.EDU.            172800      A       128.249.1.1
```

Отметим, что второй сервер имен *3com* не принадлежит сети *3com.* Организации часто имеют серверы имен в сетях своих провайдеров или в общеуниверситетских сетях.

Можно получить полную информацию о серверах имен организации, указав в команде *nslookup* *тип* запроса *ns*:

```
> set type = ns
> 3com.com.

3com.com        nameserver = NS.3COM.COM
3com.com        nameserver = TMC.EDU
3com.com        nameserver = XANTH.CS.ODU.EDU
3com.com        nameserver = AEROSPACE.AERO.ORG
3com.com        nameserver = ANTARES.AERO.ORG
Authoritative answers can be found from:
NS.3COM.COM             inet address = 129.213.128.2
TMC.EDU                 inet address = 128.249.1.1
XANTH.CS.ODU.EDU        inet address = 128.82.4.1
AEROSPACE.AERO.ORG      inet address = 130.221.192.10
>
```

12.9 Делегирование

Комитет InterNIC не обслуживает централизованно все обновляющиеся списки серверов для Австралии, Канады или Швейцарии. Каждая страна несет ответственность за собственную службу регистрации и публикует списки своих серверов доменов на собственном корневом сервере.

Производя просмотр имен серверов по коду страны, база данных InterNIC возвращает список имен и адресов корневых серверов этой страны. Диалог с программой *nslookup* показывает получение списка корневых серверов Канады:

```
> ca.

ca                          nameserver = RELAY.CDNNET.CA
ca                          nameserver = RSO.INTERNIC.NET
ca                          nameserver = CLOUSO.CRIM.CA
ca                          nameserver = SNORT.UTCC.UTORONTO.CA
ca                          nameserver = NS2.UUNET.CA
RELAY.CDNNET.CA             inet address = 192.73.5.1
RSO.INTERNIC.NET            inet address = 198.41.0.5
CLOUSO.CRIM.CA              inet address = 192.26.210.1
SNORT.UTCC.UTORONTO.CA      inet address = 128.100.102.201
NS2.UUNET.CA                inet address = 142.77.1.5
>
```

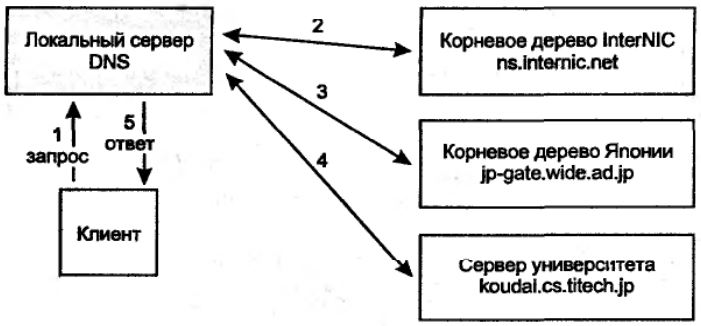

Рис. 12.4. Разрешение имен для системы из Японии

Практически DNS обеспечивает большую гибкость и позволяет формировать длинные цепочки взаимных ссылок. Страна может быть разделена на именованные регионы, и национальный корневой список будет ссылаться на корневые серверы каждого региона.

Аналогично организация может сформировать корневое дерево для собственных узлов DNS и авторизировать их как *части* именования доменов.

На практике используется относительно небольшое вторичное деление, и имена можно найти за несколько шагов. На рис. 12.4 показаны этапы разрешения (определения адреса по имени) для *viper.cs.titech.ac.jp*:

1. Производится обращение к корневому дереву InterNIC. При этом идентифицируется сервер в Японии.
2. Запрашивается один из корневых серверов Японии, который идентифицирует домен университета Titech.
3. Сервер университета Titech предоставляет адрес для хоста.

Отметим, что локальный сервер отвечает за предоставление ответа клиенту. Это правило связано с *рекурсивным* разрешением имен, что означает "искать ответ до тех пор, пока не будет получен результат".

Локальный сервер работает *нерекурсивно* (т.е. *итеративно*). Каждый из запрашиваемых серверов возвращает указатель на сервер следующего этапа поиска, и только локальный сервер посылает запрос непосредственно в базу данных.

12.10 Соединение серверов имен с Интернетом

Подключение собственного сервера DNS к общемировому Интернету предполагает несколько этапов:

1. Регистрация одного или нескольких блоков IP-адресов (возможно, и номера автономной системы)
2. Присвоение имен и адресов собственным хостам
3. Получение списка корневых серверов, объединяющих всемирную службу
4. Установка одного первичного сервера имен DNS и не менее одного вторичного
5. Тестирование серверов
6. Перевод серверов в рабочий режим
7. Регистрация имени домена организации и ее серверов в региональной регистрационной службе

12.11 Разработка базы данных сервера имен

В небольшой организации можно иметь единую базу данных. Однако это не подойдет для больших фирм, охватывающих целый географический район. Например, если компания с именем домена *fishfood.com* имеет центральный офис в штате Мэн, а региональные представительства — в Мериленде и Джорджии, то лучше делегировать управление деревом имен организации администраторам подразделений компании и создать независимые серверы имен в каждом подразделении.

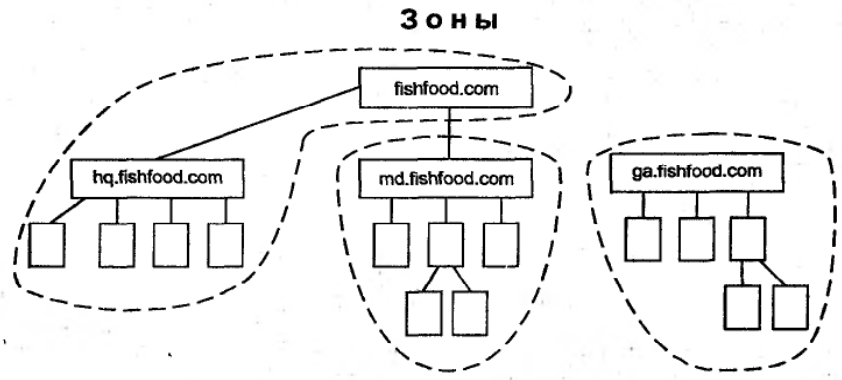

Рис. 12.5. Определение зон

12.11.1 Зоны

Дерево имен организации может состоять из одной или нескольких *зон* (zone). *Зоной называется непрерывная часть дерева имен, управляемая как единое целое.* На рис. 12.5 показана структура зон для домена *fishfood.com*.

Корневая база данных Интернета должна ссылаться на сервер имен центрального офиса компании (*fishfood.com*). Этот сервер будет формировать ответы на запросы адресов для своей зоны. Если же запрашивается имя системы из подразделений компании в Мериленде или Джорджии, то сервер центрального офиса возвратит имя и адрес сайта соответствующего подразделения компании. DNS будет пересылать запрос на сервер требуемой зоны.

12.11.2 Размещение серверов DNS

Многие организации предпочитают иметь в своей внутренней сети один комплект из первичного и вторичного серверов, даже если сеть разделена на отдельные зоны. Вполне допустимо использовать один сервер для множества зон (или для нескольких доменов). Данные для каждой зоны будут записаны в отдельном файле. Каждый такой файл при необходимости может обновляться своим собственным администратором.

12.11.3 Перенос зон

Вторичный сервер настроен на обеспечение доступа к копиям информации об одной или нескольких зонах. Он получает информацию для зоны от первичного сервера посредством *переноса зоны* (zone transfer).

Вторичный сервер может быть сконфигурирован для извлечения информации об отдельных зонах из различных первичных серверов. Таким образом, один сервер может действовать как вторичный для нескольких первичных серверов. Сервер может даже работать как первичный для одних зон и как вторичный для других. Система DNS специально разрабатывалась для обеспечения такой гибкости.

12.12 Данные DNS

Для сервера DNS требуется, по крайней мере, следующая информация:

- Список *корневых* серверов всего мира, чтобы выяснить, куда посылать внешние запросы. Файл такого списка можно скопировать с сервера регистрации InterNIC.
- Список имен и соответствующих им адресов.
- Список адресов и соответствующих им имен.

12.13 Элементы описаний в DNS

Данные DNS хранятся как набор текстовых элементов. Информационная запись содержит следующие элементы:

```
[имя] [TTL] [класс] Тип_записи Данные_записи [; комментарий]
```

Время жизни (TTL) указывает, как долго может быть кеширована запись после извлечения.

Если этот параметр не указан, то используется значение по умолчанию для *имени* или *класса* последнего элемента, включенного в список. В Интернете в текущий момент используется *единственный* класс IN, поэтому данное значение появляется только один раз, в первой записи.

Порядок расположения полей *класса* и *TTL* может быть изменен. Значение *TTL* выражается числом и, следовательно, не может быть перепутано с классом (IN).

12.13.1 Записи о ресурсах

Эта часть элемента данных состоит из:

```
[TTL] [класс] Тип_записи Данные_записи
```

и называется *записью о ресурсах* (resource record — *RR*). Существует несколько типов записей о ресурсах, каждый из которых идентифицируется символом или коротким акронимом. Типы записей о ресурсах перечислены в таблице 12.1.

Таблица 12.1

Типы записей о ресурсах

Тип записи	Описание
SOA	Start Of Authority (начало авторизации) — идентифицирует домен или зону, а также набор числовых параметров.
NS	Отображает имя домена на имя компьютера, авторитетного для данного домена.
A	Отображает имя системы на ее адрес. Если система (например, маршрутизатор) имеет несколько адресов, для каждого из них должна существовать отдельная запись.
CNAME	Отображает псевдоним на действительное *каноническое имя*.
MX	Mail Exchanger (обмен почтой). Идентифицирует систему, доставляющую в организацию сообщения электронной почты.
TXT	Обеспечивает способ добавления текстовых комментариев к базе данных. Например, запись *txt* может отображать *fishfood.com* на название компании, ее адрес или на номер телефона.
WKS	Well-Known Services (общеизвестные службы). Может перечислить доступные на хосте прикладные службы. Используется не слишком часто, если вообще применяется.
HINFO	Host Information (информация о хосте) — например тип компьютера и его модель. Используется редко.
PTR	Отображает IP-адрес на имя системы. Используется в файлах трансляции адресов в имена.

12.14 Пример файла трансляции имен в адреса

Рис. 12.6 демонстрирует файл трансляции имен в адреса для нашего мифического домена *fishfood.com*. Файл содержит несколько комментариев, которые отмечены символом точки с запятой (;).

12.14.1 Записи SOA

Первой записью в файле стоит начало авторизации (Start of Authority — SOA):

```
FISHFOOD.COM.      IN      SOA     NS.FISHFOOD.COM. (
                                   postmaster.FISHFOOD.COM.
                                    94101101 ; serial number
                                       86400 ; refresh after 24 hours
                                        7200 ; retry after 2 hours
                                     2592000 ; expire after 30 days
                                      345600 ; default TTL of 4 days
                                   )
```

```
;               fishfood.com file
FISHFOOD.COM.       IN      SOA     NS.FISHFOOD.COM. (
                                    postmaster.FISHFOOD.COM.
                                     94101101 ; serial number
                                        86400 ; refresh after 24 hours
                                         7200 ; retry after 2 hours
                                      2592000 ; expire after 30 days
                                       345600 ; default TTL of 4 days
                                    )

FISHFOOD.COM.       IN      NS      NS.FISHFOOD.COM.
FISHFOOD.COM.       IN      NS      NS2.FISHFOOD.COM

LOCALHOST           IN      A       127.0.0.1
NS                  IN      A       172.66.1.1
NS2                 IN      A       172.66.1.100
;
MAIL-RELAY      IN      A       172.66.1.2
                IN      TXT     www, ftp on mail-relay
                IN      TXT     gopher on mail-relay
                IN      HINFO   SUN UNIX     ;should not include
;
WWW                 IN      CNAME   MAIL-RELAY
FTP                 IN      CNAME   MAIL-RELAY
GOPHER              IN      CNAME   MAIL-RELAY
;
FISHFOOD.COM.       IN      MX      1       MAIL-RELAY
*                   IN      MX      1       MAIL-RELAY
NS                  IN      MX      1       MAIL-RELAY
;
;end of fishfood.com file
;
```

Рис. 12.6. Пример файла трансляции имен в адреса

Круглые скобки в записи SOA позволяют расширить эту запись на следующие строки. В запись включено несколько значений тайм-аута (измеряются в секундах). Данная запись SOA указывает:

- Сервер *ns.fishfood.com* является первичным для домена *fishfood.com*.
- Сведения о всех возникающих проблемах нужно сообщать на *postmaster@fishfood.com* (следует заменить первую точку на символ @).

Вторичные серверы будут копировать файл целиком, получая при этом важную информацию из следующих четырех пунктов в записи SOA. Из приведенной выше записи каждый вторичный сервер узнает, что ему необходимо:

- Соединяться с первичным сервером каждые 24 часа.
- Проверять, не стал ли текущий порядковый номер меньше, чем порядковый номер первичного сервера. Если это произойдет, значит информация на первичном сервере была изменена, и вторичному серверу нужно выполнить *перенос зоны*, т.е. скопировать все сведения о зоне из базы данных первичного сервера в свою систему.
- Повторить попытку соединения через 2 часа, если он не сможет соединиться с первичным в намеченное время.
- Отметить все свои данные просроченными и прекратить отвечать на запросы, если он не сможет соединяться с первичным в течение 30 дней.

Показанные в примере значения рекомендованы (см. RFC 1537) для серверов верхнего уровня.

12.14.2 Время жизни (Time-To-Live)

В RFC 1035 (специфицирует протокол DNS) заявлено, что *TTL* в записи SOA — это *минимальное* значение тайм-аута, разрешенное для всех записей. Но на практике администратору хочется использовать *TTL* в записи SOA как значение по умолчанию, указывая меньшие значения только для определенных хостов, информация на которых должна быстро измениться. В реализации следуют этому более осмысленному действию, а не требованиям стандарта.

В приведенном примере значение по умолчанию (для *TTL* — 4 дня) обычно располагается в диапазоне от одного дня до недели, в зависимости от устойчивости данного элемента сайта.

12.14.3 Дополнение имени

Имя, которое не заканчивается точкой, дополняется именем домена для зоны, например *fishfood.com*. Таким образом, в этом файле *ns* будет соответствовать *ns.fishfood.com*.

12.14.4 Запись о сервере имен

Запись о сервере имен (Name Server — NS) идентифицирует сервер для домена. Если имеются подзоны, необходимы и элементы для дочерних серверов имен подзон, чтобы сервер с более высоким уровнем мог использовать указатели на серверы низшего уровня. Записи об адресе требуются для обеспечения доступа к дочерним серверам и называются *связующими* (glue records).

Отметим, что сервер с более высоким уровнем *не авторитетен* для дочерних серверов. Отвечать за дочерние серверы могут различные администраторы. Администратор родительского сервера должен проявлять осторожность при взаимодействии с администраторами дочерних серверов и отслеживать текущие изменения в списке имен и адресов дочерних серверов.

12.14.5 Записи об адресе

Запись об адресе (address records) — это просто отображение имени в адрес. Таким образом, адресом *ns.fishfood.com* будет 172.66.1.1.

12.14.6 Записи CNAME

Вспомним, что для более осмысленного именования можно присваивать серверным хостам короткий псевдоним. В показанном примере службы World Wide Web пересылки файлов и gopher выполняются на той же машине, которая поддерживает доставку электронной почты. Запись для канонического имени (canonical name — *CNAME*) определяет короткое имя для хоста и разрешает пользователям вводить *www.fishfood.com*, *ftp.fishfood.com* или *gopher.fishfood.com*.

12.14.7 Записи для почтового обмена

Серверы обмена почтой (Mail Exchanger — MX) обеспечивают доставку в/из сети (см. главу 16). В файле существуют три записи MX, которые идентифицируют сервер MX для *fishfood.com*.

```
fishfood.com.       IN MX 1 MAIL-RELAY
*                   IN MX 1 MAIL-RELAY
ns                  IN MX 1 MAIL-RELAY
```

Эти записи фактически указывают, что:

- Почта, адресованная на *некто@fishfood.com*, должна быть доставлена на *mail-relay.fishfood.com*.
- Подстановочный символ * позволяет пересылать почту, адресованную особым хостам, которые *отсутствуют* в списке DNS. Почта, адресованная на *некто@любой_хост.fishfood.com*, должна быть доставлена на *mail-relay.fishfood.com*.
- Почта, адресованная на *некто@ns.fishfood.com*, должна быть передана по адресу *mail-relay.fishfood.com*.

Все это выглядит так, будто подстановочный символ должен полностью обеспечить обращение к *ns.fishfood.com*. Тогда зачем же нужен отдельный оператор? Правила для подстановочного символа гласят, что он может быть применен только к системам, *не имеющим* никаких других записей в базе данных DNS.

Числа, стоящие после MX, называются *предпочтительными* (preference numbers). Они будут рассмотрены в главе 16 при описании работы с электронной почтой.

12.14.8 Записи TXT и HINFO

Записи TXT не имеют никакого реального назначения, но позволяют администратору включить в базу данных произвольные комментарии.

Записи HINFO можно использовать для идентификации типа оборудования и операционной системы. Поскольку пользователи способны прочитать эти данные через программы типа *nslookup*, многие администраторы считают, что записи HINFO *не должны находиться* в базе данных, так как они могут помочь хакерам найти уязвимые системы.

12.15 Трансляция адресов в имена

Почему необходим обратный поиск и трансляция адресов в имена? Некоторые системные программы вызывают обратный поиск с целью улучшения вывода информации для администрирования. Например, показанный ниже вывод из программы *netstat* представляет все или часть имени хоста вместо IP-адресов:

```
> netstat -a
Active Internet connections (including servers )
Proto   Recv-Q   Send-Q   Local Address    Foreign Address          (state)
Tcp        0      121     tigger.nntp      c3po.4809                ESTABLISHED
Tcp        0        0     tigger.smtp      news.std.com.1472        TIME_WAIT
Tcp        0      925     tigger.1176      sun3.nsfnet-rela.smtp    ESTABLISHED
Tcp        0        0     tigger.pop-3     ringotty8.16284          TIME_WAIT
```

Кроме этого, обратный поиск используется для служб пересылки файлов и WWW, которые создают регистрационные записи о системах, использующих эти службы. *Некоторые службы отклоняют запросы клиентов, чьи IP-адреса не соответствуют одной из записей в базе данных имен доменов.*

Вспомним, что адреса размещаются в специальном домене *IN-ADDR.ARPA* и дерево этого домена должно расширяться от наиболее общей части адреса к менее общей. При этом порядок номеров в каждом адресе становится обратным. Поэтому поддерево сети 172.66 называется *66.172.in-addr.arpa*. На рис. 12.7 показан обратный поиск записи.

```
66.172.in-addr.arpa.      IN      SOA     NS.FISHFOOD.COM. (
                                              postmaster.FISHFOOD.COM.
                                               94101101 ; serial
                                                  86400 ; refresh after 24 hours
                                                   7200 ; retry after 2 hours
                                                2592000 ; expire after 30 days
                                                 345600 ; default TTL of 4 days
                                              )
66.172.in-addr.arpa.      IN      NS      NS.FISHFOOD.COM.
1.1                       IN      PTR     NS.FISHFOOD.COM.
66.172.in-addr.arpa.      IN      NS      NS2.FISHFOOD.COM.
100.1                     IN      PTR     NS2.FISHFOOD.COM.
2.1                       IN      PTR     MAIL-RELAY.FISHFOOD.COM.
```

Рис. 12.7. Таблица обратного просмотра

Элементы также будут обратными. Например, элементу 100.1 соответствует адрес 172.66.1.100.

12.16 Формат сообщений DNS

Обмен сообщениями запросов и ответов между клиентом и серверами DNS имеет простой формат. Сервер добавляет информацию ответа к исходному запросу и посылает полученное сообщение обратно. На рис. 12.8 показан полный формат сообщения.

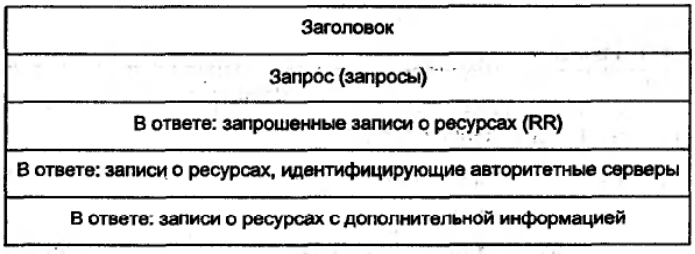

Рис. 12.8. Общий формат сообщения DNS

12.16.1 Секция заголовка

Секция заголовка содержит поля, представленные в таблице 12.2.

Таблица 12.2

Поля заголовка сообщения DNS

Поле	Описание
ID Идентификатор	Служит для согласования запроса и ответа.
Parameters Параметры:	Запрос или ответ. Обычный или обратный просмотр. Является ли ответ авторитетным. Является ли ответ усеченным. Рекурсивно или нет сообщение. Допустима ли рекурсия в ответе. Для ответа — код ошибки.
Number of queries Количество запросов	Присутствует в запросе и ответе.
Number of answers Количество ответов	Присутствует только в ответе.
Number of authority records Количество авторитетных записей	Присутствует в ответе. Информация в авторитетных записях включает имя сервера, хранящего авторитетные данные.
Number of additional records Количество дополнительных записей	Присутствует в ответе и содержит адреса авторитетных серверов.

12.16.2 Секция запроса

Запрос имеет поля, перечисленные в таблице 12.3. Обычно сообщение содержит единственный запрос. Но можно в общей секции объединить несколько различных запросов.

Таблица 12.3

Поля запросов DNS

Поле	Описание
Name (Имя)	Имя домена или IP-адрес в *поддереве IN-ADDR.ARPA*
Type (Тип)	Тип запроса, например A или NS
Class (Класс)	IN для Интернета записывается как 1

12.16.3 Секции ответа

Сам ответ, информация об авторитетности и дополнительные сведения структурированы так же, как и в запросе. Ответ состоит из последовательности записей о ресурсах, содержащих поля, показанные в таблице 12.4.

Таблица 12.4

Поля записей о ресурсах

Поле	Описание
Name (Имя)	Имя узла для данной записи
Type (Тип)	Тип записи, например SOA или A, записанный числовым кодом
Class (Класс)	IN соответствует 1
TTL	Время жизни 32-разрядное целое число со знаком, отражающее время кеширования записи
RDLENGTH Длина записи	Длина поля данных в записи о ресурсах
RDATA Данные записи	Например, для записи об адресе — значение IP-адреса. Запись SOA содержит обширные сведения.

Секция информации об авторитетности указывает авторитетные серверы имен для домена. Секция дополнительной информации предоставляет сведения, подобные IP-адресам авторитетных серверов имен.

12.17 Используемый транспорт

Запросы и ответы DNS обычно пересылаются через UDP, но разрешается применять и TCP, который используется для переносов зон.

12.18 Примеры

Некоторые реализации программы *nslookup* позволяют рассмотреть сообщения более подробно. Ниже приводится результат запуска *nslookup* на хосте Йельского университета и указывается вывод детальной отладочной информации с помощью команды *set d2*.

Запрос требовал трансляции имени *www.microsoft.com* в адрес, а в ответе было получено два адреса. Дело в том, что два различных компьютера работают как сервер WWW компании Microsoft и разделяют между собой трафик от клиентов. Если клиент не получает ответа по первому адресу (возможно, при сильной загруженности этой системы), он может обратиться ко второму компьютеру.

```
> ns lookup
Server: DEPT-GW.cs.YALE.EDU
Address: 128.36.0.36

> set d2
> www.microsoft.com.
Server: DEPT-GW.cs.YALE.EDU
Address: 128.36.0.36

res_mkquery(0, www.microsoft.com, 1, 1)
------
SendRequest(), len 35
HEADER:
      opcode = QUERY, id = 5, rcode = NOERROR
      header flags: query, want recursion
      questions = 1, answers = 0, auth. records = 0, additional = 0

QUESTIONS:
      www.microsoft.com, type = A, class = IN
------
------
```

```
Got answer (67 bytes):
HEADER:
        opcode = QUERY, id = 5, rcode = NOERROR
        header flags: response, auth. answer, want recursion,
        recursion avail.
        questions = 1, answers = 2, auth. records = 0, additional = 0

QUESTIONS:
        www.microsoft.com, type = A, class = IN
ANSWERS:
        -> www.microsoft.com
           type = A, class = IN, ttl = 86400, dlen = 4
           inet address = 198.105.232.5
        -> www.microsoft.com
           type = A, class = IN, ttl = 86400, dlen = 4
           inet address = 198.105.232.6
```

Ответ локального сервера не содержит авторитетных записей и дополнительных сведений. Однако этот сервер получал авторитетность и дополнительную информацию от запрашиваемых *им* серверов и кешировал такие сведения.

При повторном запросе ответ придет из кеша локального сервера. Так как информация не авторитетна, локальный сервер предоставляет в ответе имена и адреса авторитетных серверов для *microsoft.com*.

```
> www.microsoft.com.
Server: DEPT-GW.cs.YALE.EDU
Address: 128.36.0.36

res_mkquery(0, www.microsoft.com, 1, 1)
------
SendRequest(), len 35
HEADER:
        opcode = QUERY, id = 8, rcode = NOERROR
        header flags: query, want recursion
        questions = 1, answers = 0, auth. records = 0, additional = 0

QUESTIONS:
        www.microsoft.com, type = A, class = IN
------
------
Got answer (194 bytes):
HEADER:
        opcode = QUERY, id = 8, rcode = NOERROR
        header flags: response, want recursion, recursion avail.
        questions = 1, answers = 2, auth. records = 3, additional = 3

QUESTIONS:
        www.microsoft.com, type = A, class = IN
ANSWERS:
        -> www.microsoft.com
           type = A, class = IN, ttl = 86392, dlen = 4
           inet address = 198.105.232.5
        -> www.microsoft.com
           type = A, class = IN, ttl = 86392, dlen = 4
           inet address = 198.105.232.6
```

```
AUTHORITY RECORDS:
      -> MICROSOFT.COM
         type = NS, class = IN, ttl = 172792, dlen = 7
         nameserver = ATBD.MICROSOFT.COM
      -> MICROSOFT.COM
         type = NS, class = IN, ttl = 172792, dlen = 16
         nameserver = DNS1.NWNET.NET
      -> MICROSOFT.COM
         type = NS, class = IN, ttl = 172792, dlen = 7
         nameserver = DNS2.NWNET.NET
ADDITIONAL RECORDS:
      -> ATBD.MICROSOFT. COM
         type = A, class = IN, ttl = 187111, dlen = 4
         inet address = 131.107.1.7
      -> DNS1.NWNET.NET
         type = A, class = IN, ttl = 505653, dlen = 4
         inet address = 192.220.250.1
      -> DNS2.NWNET.NET
         type = A, class = IN, ttl = 505653, dlen = 4
         inet address = 192.220.251.1
```

Отметим, что в обоих запросах о *www.microsoft.com.* введена конечная точка. Если она опущена, запрос первоначально будет послан с добавленным в конец именем локального домена.

Это демонстрирует запуск запроса на компьютере Йельского университета, подключенном к *cs.yale.edu.* В следующем примере показаны происходящие при этом события. Запрос был отклонен, но далее автоматически переделан для исключения дополнительных обозначений в конце имени.

```
> www. microsoft.com
Server: DEPT-GW.CS.YALE.EDU
Address: 128.36.0.36

res_mkquery(0, www.microsoft.com.CS.YALE.EDU, 1, 1)
------
SendRequest(), len 47
HEADER:
      opcode = QUERY, id = 6, rcode = NOERROR
      header flags: query, want recursion
      questions = 1, answers = 0, auth. records = 0, additional = 0

QUESTIONS :
www.microsoft.com.CS.YALE.EDU, type = A, class = IN
...
```

12.19 Дополнительные типы записей

Одним из способов увеличения преимуществ Domain Name System является определение новых типов записей. За последние годы предложено множество таких типов. Полезные — добавлены в DNS, другие не вышли за рамки экспериментального применения.

Существуют определенные типы записей, используемые только в отдельных случаях, например в сетевом протоколе без установления соединений (Connectionless Network Protocol — CLNP) из третьего уровня модели OSI. Этот протокол рассматривается как часть Интернета.

В OSI используется адресация точек доступа к сетевым службам (Network Service Access Point — NSAP), обеспечивающая маршрутизацию данных на хосты. Поскольку для этого требуется отображение имен и адресов на хосты OSI, для баз данных DNS были определены записи с типом NSAP, обеспечивающие трансляцию имен в адреса. Обратное отображение обычно выполняется через записи с типом PTR.

Позже мы узнаем, что определен новый тип записи для трансляции имен в адреса IP версии 6.

12.20 Недостатки DNS

Domain Name System — очень важная система. Некорректные элементы базы данных могут сделать невозможным доступ к прикладным хостам. Поскольку многие администраторы используют распределенную базу данных с ручным вводом информации, весьма вероятно возникновение ошибок. К типичным проблемам DNS можно отнести:

- Отсутствие точки в конце полного имени.
- Отсутствие записей NS. Иногда новый сервер имен оказывается не внесенным в список везде, где на него должны присутствовать ссылки (например, в базе данных родительского домена).
- Противоположная проблема — *искаженное делегирование* (lame delegation), когда запись NS для сервера имен перестает существовать. Это может причинить *множество* неудобств.
- Неудачное изменение *связывания* записей (которые обеспечивают адреса серверов имен для дочерних зон), когда изменяются серверы имен дочерней зоны.
- Неправильная запись MX, указывающая на систему, *не являющуюся* службой почтового обмена для домена.
- Незнание правила о том, что подстановочный символ в записи MX неприменим для систем, уже имеющих запись в базе данных. Для таких систем требуется отдельная запись MX.
- Псевдоним, ссылающийся на другой псевдоним.
- Псевдонимы, указывающие на неизвестные имена хостов.
- Адресная запись без соответствующей записи PTR.
- Запись PTR без соответствующей адресной записи.

К счастью, существуют бесплатные программные инструменты для отладки баз данных DNS. Они описаны в RFC *Tools for DNS Debugging* (инструменты для отладки DNS).

12.21 Дополнительная литература

Для Domain Name System существует много документов RFC. Мы упомянем только наиболее важные из них.

RFC 1034 определяет концепции и возможности DNS. RFC 1035 описывает реализации и спецификации протокола для Domain Name System. В этих документах можно найти детальное описание форматов сообщений.

RFC 1713 специфицирует отдельные инструменты для отладки DNS. RFC 1912, 1536 и 1537 рассматривают общие ошибки конфигурирования DNS и недостатки реализаций, а также предлагает способы решения данных проблем.

Глава
13

Telnet

13.1 Введение

Кому нужна сеть с обширным набором приложений, если пользователи не могут регистрироваться на различных компьютерах и использовать эти приложения через сеть? TCP обеспечивает межкомпьютерное взаимодействие, но при этом возникают определенные препятствия. В течение длительного времени разработчики компьютеров считали рынок полностью предназначенным для лицензионных продуктов. К приложению на хосте от конкретного разработчика можно было обращаться только через специальные терминалы, изготовленные той же компанией.

Протокол сетевого взаимодействия терминалов (terminal networking — *telnet*) позволил преодолеть различия в оборудовании от разных компаний, и теперь пользователь может связаться с любым хостом сети. Эмуляция терминалов по протоколу *telnet* стала первым приложением TCP/IP. Этот протокол был разработан как основа для единообразных коммуникаций между приложениями. Поскольку организации понемногу отходили от приложений для терминалов, *telnet* все больше стал использоваться как комплект инструментов для создания приложений клиент/сервер. Фактически *telnet* лежит в основе взаимодействий между клиентом и сервером для пересылки файлов, электронной почты и работы с WWW.

В этой главе мы рассмотрим возможности *telnet*, помогающие пользователю обратиться к удаленному приложению, а также выясним, что предлагает *telnet* для создания прикладных приложений клиент/сервер.

13.2 Использование telnet для удаленной регистрации

Telnet обеспечивает эмуляцию различных типов терминалов, что позволяет осуществить доступ к компьютерам Unix, системам VAX/VMS или большим ЭВМ (майнфреймам) компании IBM. Некоторые реализации *telnet* поддерживают специальную процедуру аутентификации. Примером может служить система *Kerberos* Массачусетского технологического института (Massachusetts Institute of Technology — MIT). В Kerberos пароли никогда не передаются по сети и используется специальная процедура шифрования. Для запуска аутентификации может потребоваться ввод специальной команды, например *kinit*. Существуют и более простые процедуры, основанные на взаимной проверке.

При запуске *telnet* на многопользовательской системе, возможно, для пользователя будет применяться простой текстовый интерфейс. Использование текстового клиента *telnet* необычайно просто. Нужно набрать что-нибудь вроде:

```
> telnet имя_хоста
```

Часто эмуляция терминала IBM 3270 реализована отдельно, и для доступа к хостам IBM потребуется ввести:

```
> tn3270 имя_хоста
```

Большинство пользователей начинают успешно применять *telnet* без подробного изучения. Ниже показан пример регистрации (login) в системе *tigger* компьютера Йельского университета.

```
> telnet tigger.jvnc.net
Trying 128.121.50.145 ...
Connected to tigger.jvnc.net.
Escape character is '^]'.
```

```
SunOS UNIX (tigger.jvnc.net)

SunOS UNIX (pascal)

login: xxxxx
Password:
Last login: Wed Aug 23 19:24:02
TERM = vt100, PRINTER = 1p
```

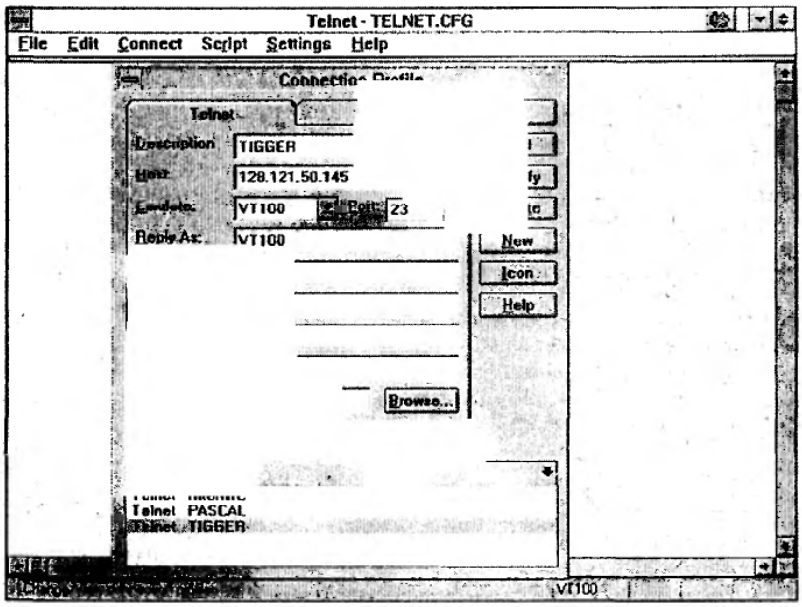

Рис. 13.1. Приложение *telnet* для настольной системы (Chameleon)

Продукты *telnet* для настольных компьютеров предлагают дополнительные функциональные возможности, например выбор из списка типа терминала, сохранение всего или части сеанса в файле журнала (log file), конфигурирование раскладки клавиатуры или запись всей информации, необходимой для доступа к часто посещаемым сайтам. Некоторые из этих возможностей показаны на рис. 13.1.

13.3 Обращение по telnet к заданному порту

Порт 23 — стандартный общеизвестный порт для терминального доступа по протоколу *telnet*. Когда клиент соединяется с портом 23, обычно в ответ следует приглашение для ввода идентификатора регистрации (login ID) и пароля.

Поскольку *telnet* был разработан как средство для коммуникаций между приложениями, он доставит клиента к любому порту. Например, в показанном ниже диалоге мы соединяемся с популярной службой прогноза погоды Мичиганского университета, которая запускается через порт 3000 и не требует ввода идентификатора регистрации или пароля:

```
> telnet madlab.sprl.umich.edu 3000
Trying 141.213.23.12 ...
Connected to madlab.engin.umich.edu.
Escape character is '^]'.
```

```
-----------------------------------------------------------------------------
*                          University of Michigan                           *
*                            WEATHER UNDERGROUND                            *
-----------------------------------------------------------------------------
*                                                                           *
*              College of Engineering, University of Michigan               *
*          Department of Atmospheric, Oceanic, and Space Sciences           *
*                      Ann Arbor, Michigan 48109-2143                       *
*                   comments: ldm@cirrus.sprl.umich.edu                     *
*                                                                           *
*        With Help from: The National Science Foundation supported          *
*                              Unidata Project                              *
*              University Corporation for Atmospheric Research              *
*                          Boulder. Colorado 80307-3000                     *
*                                                                           *
*     Commercial, for-profit users should contact our data provider,        *
*     Alden Electronics, 508-366-8851 to acquire their own data feed.       *
*                  comments: ldm@cirrus.sprl.umich.edu                      *
*                                                                           *
-----------------------------------------------------------------------------
* NOTE:--> New users, please select option "H" on the main menu:            *
* H) Help and information for new users                                     *
-----------------------------------------------------------------------------
Press Return for menu, or enter 3 letter forecast city code:
```

При всей своей полезности возможность доступа по *telnet* к любому порту в то же время является потенциальным источником проблем с безопасностью системы, поскольку взломщики могут проникнуть на сайт через плохо разработанную программу, открывающую один из портов.

13.4 Модель эмуляции терминала в Telnet

Как показано на рис. 13.2, пользователь с реального терминала взаимодействует с локальной *клиентской* программой *telnet*. Эта программа принимает введенные с клавиатуры символы, интерпретирует их и выводит результат на пользовательский экран в том виде, в каком он должен выглядеть на эмулируемом терминале.

Клиент *telnet* открывает соединение TCP с сервером *telnet* через общеизвестный порт 23. Сервер взаимодействует с приложением и помогает эмулировать исходный терминал.

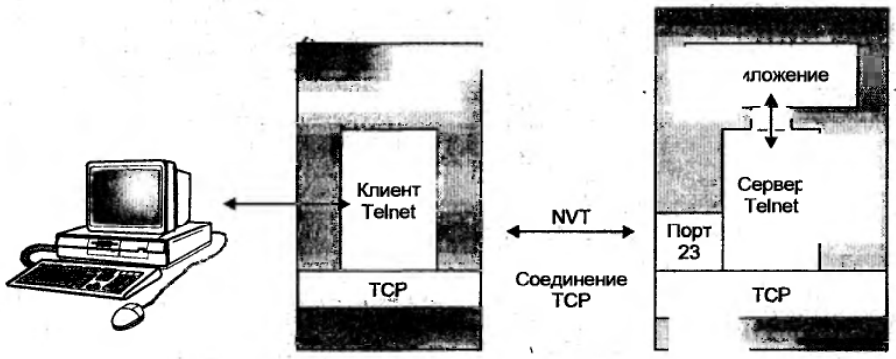

Рис. 13.2. Клиент и сервер в *Telnet*

13.4.1 Сетевой виртуальный терминал

Для работы во время сеанса обе стороны предварительно обмениваются информацией по очень простому протоколу *сетевого виртуального терминала* (Network Virtual Terminal — NVT).

Протокол NVT моделирует работу уже устаревшей полудуплексной клавиатуры и принтера, работающих в построчном режиме. Характеристики NVT общеизвестны:

- Данные состоят из 7-разрядных символов USASCII, дополненных до 8 бит начальным нулем.
- Данные пересылаются построчно.
- Каждая строка завершается комбинацией символов ASCII для возврата каретки (Carriage Return — CR) и перевода строки (Line Feed — LF).
- Байты с начальным битом, равным 1 (с наибольшим весом), используются как коды команд.
- Протокол работает в полудуплексном режиме. После отправки строки клиент переходит к ожиданию данных от сервера. Сервер посылает данные, а затем команду *Go Ahead* (продолжить), указывающую клиенту на возможность отправки следующей строки.

13.5 Наиболее распространенные типы терминалов

Обычно клиент и сервер остаются в режиме NVT очень короткое время — пока не согласуют между собой тип эмулируемого терминала (например, ASCII VT100 или IBM 3270).

За годы существования *telnet* в этот протокол были добавлены многие типы терминалов.

13.5.1 Терминалы ASCII

Терминалы ASCII используются с Unix и компьютерами VAX компании Digital Equipment Corporation. Эти терминалы обеспечивают:

- *Удаленную эхо-печать* (remote echoing) каждого символа. Т.е. каждый посланный удаленному хосту символ возвращается назад, до того как будет отображен на экране пользователя (это приводит к существенной нагрузке на сеть).
- *Полнодуплексный обмен.* Поток символов одновременно передается в обоих направлениях. Серверу не требуется посылать управляющий код *Go Ahead.*
- Поддержку *интерактивных* полноэкранных приложений (это также существенно загружает сеть).
- Набор символов ASCII больше набора символов NVT.

Основные характеристики терминалов ASCII определены в стандартах ANSI X3.64, ISO 6429 и ISO 2022. Многие модели терминалов ASCII имеют некоторые дополнительные возможности (например, ANSI, VT52, VT100, VT220, TV1950, TV1955 и WYSE50). Для регистрации с удаленных компьютеров Unix наиболее часто эмулируется терминал VT100.

13.5.2 Конфигурирование раскладки клавиатуры

Клавиатуры компьютеров PC или Macintosh не идентичны клавиатурам терминалов VT100 или 3270. Приложения *telnet* обычно обеспечивают способ конфигурирования отдельных клавиш клавиатуры или управляющих комбинаций клавиш для выполнения функций, доступных на клавиатуре эмулированных систем. Например, с терминалом VT*XXX* возникает проблема из-за того, что не стандартизирована клавиша удаления последнего введенного символа. Некоторые терминалы используют для этого клавишу *Backspace*, а другие — *Del.*

Для систем Unix можно настроить клавиатуру эмулируемого терминала через элементы конфигурационного файла */etc/termcaps.* Приложение *Chameleon* (для работы с *telnet* в среде Windows) обеспечивает более простой способ такой настройки (см. рис. 13.3). В конфигурационном экране этого приложения можно перетаскивать мышью клавиши с верхнего изображения клавиатуры на нижнее, отражающее соответствующую клавиатуру PC. Например, если нужно, чтобы клавиша *Backspace* на PC формировала в *telnet* код клавиши *Del* терминала VT100, достаточно перетащить клавишу *Del* верхнего изображения на клавишу *Backspace* нижнего изображения клавиатуры.

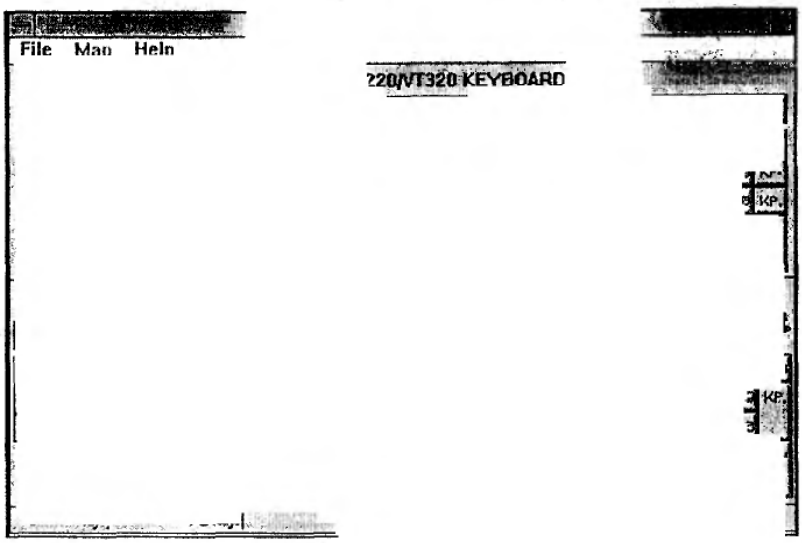

Рис. 13.3. Конфигурирование клавиатуры перетаскиванием мышью

13.5.3 Терминалы IBM 3270 и 5250

Большие ЭВМ компании IBM поддерживают работу сотен или тысяч интерактивных терминалов. Многие годы для этого использовались терминалы IBM 3270, лицензированные данной компанией. Они были специально оптимизированы для приложений обработки данных.

Терминалы IBM 3270 работают в *блочном режиме*, в котором пользователь каждый раз получает целый экран данных. Когда он нажимает клавишу ENTER или другую функциональную клавишу, на хост пересылается содержимое всего экрана. Клавиатура блокируется, а хост начинает обработку полученных данных. Затем хост отправляет обратно один или несколько экранов данных. Завершив пересылку, хост разблокирует клавиатуру терминала. Терминалы IBM 3270 имеют следующие характеристики:

- 8-разрядные коды EBCDIC
- Полудуплексный режим взаимодействия
- Блочный метод обмена

Для доступа к компьютерам AS/400 применяются терминалы IBM 5250, имеющие подобные характеристики.

13.6 Варианты

Характеристики эмуляции терминала устанавливаются с помощью обмена командами согласования *вариантов работы telnet*. Любая сторона может запросить от партнера выполнения (команда *DO*) одного из вариантов, например эхопечати каждого символа. Партнер выполняет такую команду или отклоняет ее. Любая сторона может по желанию (команда *WILL*) запросить исполнение определенного варианта, а партнер — разрешить или запретить эти действия.

Существующие четыре пары запросов/ответов используются в процессе согласования характеристик обмена:

DO (*код варианта*)	Запрос от партнера на выполнение операции.
WILL (*код варианта*)	Согласие партнера. Запрашиваемый вариант согласован.
DO (*код варианта*)	Запрос от партнера на выполнение операции.
WON'T (*код варианта*)	Отказ партнера. Состояние обмена не меняется.

WILL (код варианта)	Указывает на желание начать операцию.
DO (код варианта)	Согласие партнера. Запрашиваемый вариант согласован.
WILL (код варианта)	Указывает на желание начать операцию.
DON'T (код варианта)	Отказ партнера. Состояние обмена не меняется.
WON'T (код варианта)	Подтверждение сохранения текущего состояния обмена.

При запуске соединения между партнерами производится обмен множеством сообщений. Иногда согласование варианта работы происходит и в середине сеанса. Некоторые сигналы выбора варианта начинают *дополнительное согласование* (subnegotiations), с обменом соответствующей информацией.

Что происходит, когда обе стороны отказываются от каждого запроса выбора варианта? Ответ прост — сеанс остается в режиме NVT.

13.6.1 Типы терминалов

Очень важен выбор *типа терминала* (Terminal Type). При этом происходит дополнительное согласование. Клиент посылает *WILL TERMINAL TYPE*, сообщая серверу типы терминалов, которые он может эмулировать. При желании ознакомиться с этой информацией сервер отвечает: *DO TERMINAL TYPE.*

Далее при дополнительном согласовании сервер запросит у клиента указать один из типов терминала, которые может эмулировать клиент. Клиент ответит сообщением установленного формата. Сервер продолжит запросы, пока не найдет в ответах клиента нужного типа терминала или пока не закончится список доступных для эмуляции клиентом типов терминалов. Допустимые типы терминалов определены в RFC *Assigned Numbers*: это могут быть DEC-VT100, HP-2648 или IBM-3278-2.

13.6.2 Согласование типа терминала VT100

В приведенном ниже примере диалога мы запустили сеанс *telnet* и ввели команду *toggle options* (переключение варианта), указывающую *telnet* на отображение операций по согласованию параметров. Команда *open* используется для запуска регистрации. Партнеры согласовывают между собой эмуляцию терминала ASCII VT100, выбирая следующие характеристики:

- Сервер не будет посылать сообщений *Go Ahead*, поскольку сеанс работает в полнодуплексном режиме.
- Используется дополнительное согласование *типа терминала* для указания на эмуляцию определенной модели терминала ASCII.
- Сервер будет выполнять *эхо-печать* всех символов от клиента.

Ни одна из сторон не обязана ожидать ответа на запрос перед посылкой другого запроса. Согласующая сторона может отвечать на запросы в иной последовательности, чем они были отправлены. В результате иногда нужно распутать серию сообщений о согласовании, прежде чем станет понятна последовательность выполнения операций.

```
> telnet
telnet> toggle options
Will show option processing.

telnet> open cantor.cs.yale.edu
Trying 128.36.12.26 ...
Connected to cantor.cs.yale.edu.
Escape character is '^]'.

SENT do SUPPRESS GO AHEAD
SENT will TERMINAL TYPE (ответ)
RCVD do TERMINAL TYPE (без ответа)
RCVD will SUPPRESS GO AHEAD (без ответа)
RCVD will ECHO (ответ)
SENT do ECHO (ответ)

login:
```

13.6.3 Согласование характеристик терминала 3270

Аналогичный обмен происходит при установке эмуляции типа терминала IBM 3270. Показанный ниже диалог представляет согласование регистрации на хосте IBM VM с терминала 3270. В этом примере удаленный хост выводит на экран сведения для дополнительного согласования при установке типа терминала. Партнеры согласовывают между собой эмуляцию терминала IBM 3278 Model 2 с выбором следующих характеристик:

- *Дополнительное согласование* типа терминала специфицирует для терминала 3270 вариант "3278 модель 2".
- Клиент и сервер запрашивают вариант *END OF RECORD*, чтобы установить для терминала 3270 блочный режим.
- Обе стороны соглашаются использовать 8-разрядные двоичные данные для представления потока данных терминала 3270.

```
> tn3270

tn3270> toggle options
Will show option processing.

tn3270> open uoft.utoledo.edu
Trying...
Connected to uoft.utoledo.edu.
RCVD do TERMINAL TYPE (ответ)
SENT will TERMINAL TYPE (без ответа)
Received suboption Terminal type - request to send.
Sent suboption Terminal type is IBM-3278-2.
RCVD do END OF RECORD (ответ)
SENT will END OF RECORD (без ответа)
RCVD will END OF RECORD (ответ)
SENT do END OF RECORD (ответ)
RCVD do BINARY (ответ)
SENT will BINARY (без ответа)
RCVD will BINARY (ответ)
SENT do BINARY (ответ)

RUNNING
```

13.7 Управление текстовым клиентом telnet

Время от времени требуется осуществить взаимодействие с текстовым клиентом *telnet* и вывести или установить его параметры. Локальные команды конкретной реализации можно выяснить, если запустить *telnet* и напечатать "?" или "help".

```
> telnet

telnet> ?
Commands may be abbreviated. Commands are:

Close      close current connection
Display    display operating parameters
Mode       try to enter line-by-line or character-at-a-time mode
Open       connect to a site
Quit       exit telnet
Send       transmit special characters ('send ?' for more)
Set        set operating parameters ('set ?' for more)
Status     print status information
```

```
Toggle     toggle operating parameters ('toggle ?' for more)
Z          suspend telnet
?          print help information
```

Как только пользователь попадает в окружение *telnet*, для соединения с удаленным хостом применяется команда *open*.

```
telnet> open plum.math.yale.edu
Trying 130.132.23.16 ...
Connected to plum.math.yale.edu.
Escape character is '^]'.

login: xxxxxxxx
Password: xxxxxxxx
Last login: Sat Dec 28 06:30:44 from golem.cs.yale.ed
Sun UNIX 4.2 Release 3.4 (Plum-EGP) #3: Tue Aug 2 10:25:24 EDT 1988
**********************************************************************
*                                                                    *
*       Welcome to the Yale Mathematics Department's Fabulous        *
*                            ** Plum **                              *
**********************************************************************
You have mail.
```

13.7.1 Важные управляющие последовательности

Как пользователь может изменить характеристики активного сеанса или прервать его? Одна комбинация управляющих клавиш всегда резервируется для операции *перехода в командный режим telnet*. По умолчанию такой последовательностью обычно бывает CONTROL и] (иногда записывается как ^]). Эта esc-последовательность может быть переопределена пользователем. Вспомним, что после открытия соединения с *plum.math.yale.edu* были выведены три строки, одна из которых указывала используемый символ Esc (отмена):

```
Escape character is `^]'.
```

После вывода этой строки диалог был продолжен. Ввод esc-последовательности позволяет вывести приглашение *telnet*. Теперь можно узнать текущее состояние сеанса:

```
^]
telnet> status
Connected to plum.math.yale.edu.
Operating in character-at-a-time mode.
Escape character is `^]' .
```

Выполнив эту команду, сеанс возвращается в режим эмуляции терминала.

Для ввода следующей команды управления нужно опять воспользоваться esc-последовательностью. Запросим вывод текущих атрибутов сеанса *telnet*:

```
^]
telnet> display
will flush output when sending interrupt characters.
won't send interrupt characters in urgent mode.
won't map carriage return on output.
won't recognize certain control characters.
won't process ^S/^Q locally.
won't turn on socket level debugging.
won't print hexadecimal representation of network traffic.
won't show option processing.
```

```
[^E] echo.
[^]] escape.
[^?] erase.
[^O] flushoutput.
[^C] interrupt.
[^U] kill.
[^\] quit.
[^D] eof.
```

13.8 Возможности NVT

В следующих разделах мы подробно исследуем структуру *telnet* и изучим возможности, которые он может предоставить разработчику приложений клиент/сервер.

По окончании согласования параметров сеанса отдельные варианты эмуляции терминала могут обеспечивать большой набор символов и графических значков для взаимодействия между пользователем и приложением.

Однако, когда *telnet* используется для создания приложений клиент/сервер, все взаимодействия или большая их часть происходят в режиме NVT. Рассмотрим характеристики этого режима более подробно.

13.8.1 Набор символов NVT

Пересылаемые во время сеанса NVT октеты представляют собой символы *USASCII* и команды *telnet*. Существует 128 символов USASCII. Из них: 95 — доступные для отображения буквы, числа, символы и знаки препинания; 33 — управляющие символы ASCII (например, *горизонтальная табуляция*). Коды USASCII разработаны как 7-разрядные. Символы USASCII передаются как октеты со старшим битом, равным 0.

13.8.2 Принтер NVT

В течение основного сеанса NVT сервер *telnet* посылает алфавитно-цифровые и управляющие символы на клиентский *принтер NVT*, т.е. на экран терминала пользователя. Вывод на экран ограничен 95 символами USASCII, соответствующими кодам ASCII от 32 до 126.

Для управления экраном клиента серверу доступно небольшое подмножество управляющих символов (см. таблицу 13.1). В таблице коды ASCII представлены десятичными числами.

Таблица 13.1

Управление принтером NVT

Описание	Код ASCII
Null (Пустой, используется как заполнитель)	0
Bell (Звонок для вывода звукового сигнала)	7
Backspace (На шаг назад, перемещение на один символ влево)	8
Horizontal tab (Горизонтальная табуляция)	9
Line feed (Перевод строки)	10
Vertical tab (Вертикальная табуляция)	11
Form feed (Перевод формата, перемещение к следующей странице)	12
Carriage return (Возврат каретки)	13

13.8.3 Взаимодействие клиент/сервер telnet в режиме NVT

Вспомним, что взаимодействие NVT является полудуплексным — клиент или сервер *telnet* в каждый момент времени производит одно из следующих действий:

- После того как клиент *telnet* послал строку, завершенную CR и LF, управление передается серверу.

- Сервер посылает клиенту строки, и в конце каждой выведенной строки он использует CR и LF для перехода к позиции следующей строки на дисплее клиента.
- Клиент *telnet* принимает вывод от сервера и может начать собственный вывод данных только после получения от сервера управляющего кода *Go Ahead*.

Отметим, что пересылаемые в сеансе *telnet* строки завершаются символами CR и LF независимо от того, какие локальные символы перевода строки используют хосты клиента и сервера.

13.9 Команды telnet

До широкого распространения сетей терминалы подключались непосредственно к компьютерам. Нажатие пользователем клавиши клавиатуры немедленно интерпретировалось операционной системой локального компьютера.

Существовали специальные клавиши управления, которые активизировали операционную систему или какую-то системную команду. Например, пользователь терминала ASCII мог одновременно нажать клавиши CONTROL и C (записывается как ^C) для указания операционной системе на завершение работы текущего приложения.

Во время сеанса *telnet* управляющие коды должны быть преобразованы в команды *telnet* и переданы на *удаленный* конец сетевого соединения в соответствующую операционную систему. Для этого клиентская программа *telnet* должна обрабатывать физические действия пользователя с клавиатурой, транслировать специальные управляющие символы в команды *telnet* и пересылать их на сервер *telnet*.

Команды *telnet* отмечаются байтом "*интерпретировать как команду*" (Interpret As Command — IAC), сопровождаемым одним или несколькими байтами кодов:

Байт Interpret As Command равен X'FF (десятичное 255).

Клиент *telnet* посылает серверу последовательности команды, чтобы указать на выполнение различных функций, например:

Прерывание (Break — BRK)	Послать сигнал "прерывание" или "внимание" процессу удаленного приложения.
Прервать процесс (Interrupt Process — IP)	Сигнал для удаленной операционной системы об остановке выполнения текущей удаленной прикладной программы (например, остановить зацикленную программу).
Прервать вывод (Abort Output — AO)	Запросить у серверного приложения не посылать оставшиеся данные текущей операции.
Вы здесь? (Are You There? — AYT)	Проверить, что сервер действительно работает.
Стереть символ (Erase Character — EC)	Пользователь обычно исправляет ошибочно введенный в строке символ с помощью клавиш *Backspace* или *Del*. В режиме посимвольной пересылки ASCII ошибочный элемент данных уже был послан удаленному приложению, поэтому требуется ввод специальной команды EC.
Стереть строку (Erase Line — EL)	Запрос к удаленному приложению на уничтожение текущей строки.

Команды могут быть посланы даже после согласования параметров соединения, когда партнеры больше не находятся в режиме NVT. Но предположим, что партнеры согласовали обмен двоичными данными. Как будет тогда распознаваться последовательность символов команды? Ответ состоит в том, что всякий раз последовательность X'FF, возникая в *данных*, удваивается при отправке. Приемник устраняет дублирование. Когда он получает одиночный X'FF (или нечетное их число), становится ясно, что поступила команда.

Легко понять, как команды *telnet* должны использоваться разработчиком приложений клиент/сервер. Например, результатом щелчка мышью на кнопке STOP браузера WWW должна стать отправка команды *Abort Output*, завершающая загрузку большого по размеру изображения или документа.

Возможности *telnet* хорошо проявляются при анализе работы конечного пользователя как клиента, обращающегося к приложению на сервере. Важно отметить, что при использовании *telnet* в качестве инструментального средства разработки команды могут быть посланы в любом направлении обмена.

13.9.1 Сигнал синхронизации

Для некоторых функций (например, *Interrupt Process*) включение команды в общий поток данных не приводит к нужным результатам. Когда *реальный* терминал посылает сигнал прерывания, хост операционной системы получает этот сигнал сразу и быстро останавливает текущее приложение.

Однако, когда *telnet* работает поверх сеанса TCP, данные доставляются *по мере получения*. Обычно удаленный сервер *telnet последовательно* обрабатывает все полученные данные. Может пройти много времени, прежде чем он увидит команду прерывания в поступающем потоке данных.

Клиент хочет быстро обратить внимание сервера на эту команду и должен сообщить ему: "Отбросить все *уже буферированные* символы, за исключением команд". Для этого клиент посылает серверу специальный сегмент TCP, называемый *сигналом синхронизации* (Synch signal).

- Такой сегмент маркирован как *срочные данные* (Urgent Data).
- Сервер будет отбрасывать всю информацию от клиента, за исключением команд, пока не достигнет специального командного кода, называемого *меткой данных* (Data Mark — DM).
- DM маркирует место, где сервер должен *прекратить* отбрасывание данных.

Когда поступает сегмент сигнала синхронизации, сервер извлекает из потока данных команды NVT, отбрасывая все остальное, пока не дойдет до Data Mark. Затем он переходит к выполнению извлеченных команд, а далее возобновляется нормальная обработка данных (стоящих после Data Mark).

13.9.2 Декодирование наиболее общих команд

В таблице 13.2 приведен списков акронимов для некоторых наиболее распространенных команд (вместе с десятичными значениями их кодов). Каждой команде должен предшествовать октет 255 (X'FF), когда она пересылается по соединению *telnet*.

Таблица 13.2

Коды команд telnet

Акроним	Команда	Код
EOF	End of File (конец файла)	236
SUSP	Suspend Current Process (приостановить текущий процесс)	237
ABORT	Abort Process (аварийное завершение процесса)	238
EOR	End of Record (конец записи)	239
NOP	No Operation (нет операции)	241
DM	Data Mark (метка данных)	242
BRK	Break (прерывание)	243
IP	Interrupt Process (прерывание процесса)	244
AO	Abort Output (отменить вывод)	245
AYT	Are You There (вы здесь?)	246
EC	Erase Character (стирание символа)	247
EL	Erase Line (стирание строки)	248
GA	Go Ahead (продолжить)	249

13.9.3 Кодирование запросов выбора вариантов

Запросы выбора вариантов кодируются тремя байтами: байтом *IAC*, октетом запроса и кодом варианта. Например, десятичное представление последовательности для *WILL TERMINAL TYPE* выглядит так:

IAC	WILL	TERMINAL TYPE
255	251	24

Это один из вариантов для дополнительного согласования. Далее должны следовать:

СЕРВЕР:

IAC	SB	TERMINAL TYPE	SEND	IAC	SE
255	250	24	1	255	240

КЛИЕНТ:

IAC	SB	TERMINAL TYPE	IS	DEC-VT220	IAC	SE
255	250	24	0	DEC-VT220	255	240

В таблице 13.3 показаны десятичные значения для кодов обычных и дополнительных согласований. Приведены также коды для часто используемых вариантов. Параметры дополнительного согласования и коды добавочных вариантов определены во многих RFC, относящихся к параметрам *telnet* (эти RFC перечислены в документе *Assigned Numbers*).

Таблица 13.3

Коды согласования и выбора вариантов

Коды согласования	
Запрос	**Код**
WILL (будет)	251
WON'T (не будет)	252
DO (выполнить)	253
DON'T (не выполнять)	254
SB (Start Subnegotiation, начало дополнительного согласования)	250
SE (End Subnegotiation, конец дополнительного согласования)	240
Примеры кодов вариантов	
Command Option (вариант команды)	**Код**
Transmit Binary (пересылка двоичных данных)	0
Echo (эхо-печать)	1
Suppress Go Ahead (подавление сообщения Go Ahead)	3
Status (состояние)	5
Timing Mark (метка времени)	6
Output Line Width (длина выходной строки)	8
Output Page Size (размер выводимой страницы)	9
Extended ASCII (расширенный набор ASCII)	17
Data Entry Terminal (терминал ввода данных)	20
Terminal Type (тип терминала)	24
End of Record (конец записи)	25
Window Size (размер окна)	31
Terminal Speed (скорость терминала)	32
Remote Flow Control (удаленное управление потоком)	33
Linemode (построчный режим)	34
Authentication (аутентификация)	37
Encryption (шифрование)	38
Extended Options List (расширенный список вариантов)	255

13.9.4 Дополнительные сведения о вариантах

Более тридцати RFC детально рассматривают различные варианты, предоставляющие специальные возможности для *telnet*. Среди них можно выделить:

- Способность опрашивать партнера о текущем состоянии параметров. Запрос и ответ о состоянии партнера переносятся при дополнительном согласовании.
- Согласование размера окна. Партнеры соглашаются, что клиент может дополнительно согласовать высоту и ширину окна, которое будет использоваться в сеансе *telnet*. Эта возможность особенно полезна для запуска сеанса *telnet* в системах с многооконным интерфейсом.

Реализациям не требуется поддерживать все или многие из определенных в стандартах вариантов. Два из них, используемые при эмуляции терминала 3270, имеют специальные возможности:

- *Transmit Binary* (пересылка двоичных данных). Начало отправки 8-разрядных двоичных данных (сеансы с терминалом IBM 3270 проводятся в двоичном режиме).
- *End of Record* (конец записи). После получения DO END-OF-RECORD партнер использует стандартные управляющие коды IAC 239 для маркировки конца записи в общем потоке данных.

Вспомним, что даже после перехода в двоичный режим партнеру можно послать команды *telnet*, удваивая esc-символы IAC.

13.10 Применение telnet

С точки зрения пользователей, желающих получить доступ к приложениям через эмуляцию терминалов ASCII или IBM, наиболее важным является способность *telnet* выполнять согласование и эмуляцию. Но разработчикам прикладного программного обеспечения основанный на NVT *telnet* предлагает достаточно бедный набор средств для реализации функций клиент/сервер, которые трудно и утомительно воспроизводить в программах. Мы уже знаем, что базовыми возможностями NVT являются:

- Проверка активности равного приложения
- Сигнализация прерывания
- Запрос на прерывание удаленного текущего процесса
- Использование сигнала синхронизации для указания равному приложению на отбрасывание всех данных, кроме команд *telnet*
- Указание партнеру на отмену ожидаемой пересылки данных из буфера

13.11 Замечания о безопасности

Сегодня в локальных сетях повсеместно используются широковещательные рассылки. Многие организации используют их даже в магистральных сетях FDDI.

Пользователям PC или Macintosh очень легко найти программное обеспечение для превращения настольной системы в шпиона, который может подслушивать трафик локальной сети. Такие средства имеют многие станции Unix, владельцам которых нужно только разрешить их использование.

Традиционно пользователь доказывает свои права, посылая хосту секретный пароль. Но в локальной сети с широковещательной рассылкой передача идентификатора и пароля по сети не обеспечивает для них никакой защиты. Любой может подслушать эти сведения.

Не помогает и шифрование пароля. Взломщику даже не нужно будет расшифровывать пароль, а потребуется только переслать его в *том же* виде и таким путем получить доступ к чужим регистрационным данным. Все это свидетельствует о необходимости безопасного механизма аутентификации (установление подлинности).

13.11.1 Аутентификация в telnet

В *telnet* реализована *аутентификация*, позволяющая партнерам согласовать один из вариантов этого механизма. Последовательность действий следующая:

- Сервер посылает *DO AUTHENTICATION*
- Клиент отвечает *WILL AUTHENTICATION*

С этого момента вся информация будет пересылаться в сообщениях дополнительного согласования.

- Сервер посылает сообщение, содержащее список пар аутентификации. Каждая пара включает *тип аутентификации* (который нужно использовать) и модификатор, обеспечивающий дополнительную информацию (например, сведения об аутентификации будут посылаться только клиентом или одновременно — клиентом и сервером).
- Клиент отправляет простой идентификатор пользователя (userid) или идентификатор регистрации.
- Клиент выбирает из списка одну из пар аутентификации и посылает сообщение, идентифицирующее тип аутентификации, включая аутентификационные данные. В зависимости от протокола может потребоваться более одного сообщения.
- Сервер принимает аутентификацию.
- Если выбрана взаимная аутентификация, клиент запрашивает от сервера его аутентификационные данные.
- Сервер отвечает, сообщая свои аутентификационные данные.

Типы аутентификации зарегистрированы в IANA и имеют числовые коды. Текущее соответствие между кодами и типами таково:

Тип	KERBEROS_V4	KERBEROS_V5	SPX	RSA	LOKI	SSA
Код	1	2	3	6	10	11

В существующих реализациях все большую популярность приобретают взаимные проверки (challenge handshakes) и защитные идентификационные карты.

13.12 Замечания о производительности

Telnet не обеспечивает хорошей производительности. При эмуляции терминала ASCII (например, VT100) *telnet* очень неэффективен. Посланные клиентом сегменты часто содержат только один или несколько символов. Каждый символ нужно вернуть назад для эхо-печати. Пересылка даже небольшого количества данных приводит к серьезной загрузке сети.

Каждое интерактивное приложение имеет собственный пользовательский интерфейс с различающимися командами, управляющими кодами и правилами. Пользователям приходится обучаться работе с приложениями, и иногда требуется много времени, чтобы приобрести опыт использования программы.

Сегодня многие новые приложения построены для доступа к информации через стандартного клиента, подобного браузеру WWW. Разработчик приложения должен создать интерфейс между новым приложением и сервером WWW. Только тогда пользователи смогут работать с единообразным и знакомым интерфейсом.

13.13 X Windows

Еще недавно многие приложения разрабатывались для стандартного интерфейса X-терминала, а не для лицензированных терминалов. Система X Windows была разработана и реализована в Массачусетском технологическом институте для одновременного запуска пользователем нескольких приложений в окнах графического дисплея. Не важно, где размещаются приложения. Каждое из них фактически может выполняться на различных компьютерах сети.

Протокол X Windows обеспечивает единообразный способ управления вводом и выводом из приложения. Поэтому приложения не зависят от аппаратных средств, операционной системы и типа сети. Современные реализации этого протокола работают поверх стека TCP/IP.

Протокол может выполняться на рабочих станциях или на многопользовательском компьютере, который управляет графическими дисплеями. Существует множество специализированных программ для X Windows. Протокол этой системы очень широко распространен, и для него имеются высокофункциональные прикладные инструменты разработки. Часто эти средства встроены в продукты TCP/IP.

При использовании X Windows проявляются отдельные недостатки, связанные с необходимостью пересылки большого объема информации для вывода на экран. Это приводит к большой нагрузке на сеть.

С X Windows связаны и отдельные проблемы безопасности — очень трудно защитить систему от программ для взлома, маскирующихся под обычные приложения.

13.14 Дополнительная литература

RFC 854 определяет протокол *telnet*. Различные типы терминалов рассмотрены в: RFC 1205 для эмуляции 5250; RFC 1096 для размещения на дисплеях X-терминалов; RFC 1053 для параметров X.3 PAD; RFC 1043 для терминалов ввода данных; RFC 1041 для режимов терминала 3270. Выбор параметров терминала разъясняется в RFC 1091, а варианты размеров окна можно найти в RFC 1073. RFC 1184 описывает характеристики построчного режима *telnet*. Документы RFC с 855 по 861 рассматривают другие часто используемые параметры эмуляции.

RFC 1416 посвящен аутентификации в *telnet*. RFC 1510 представляет службу аутентификации Kerberos Network Authentication Service.

Глава
14

Протокол FTP

14.1 Введение

В сетевой среде естественным является желание копировать файлы между компьютерными системами. Почему же эту операцию не всегда легко реализовать? Разработчики компьютеров уже создали сотни различных файловых систем, значительно или не очень существенно отличающихся друг от друга. Однако проблема связана не только с продуктами различных компаний. Иногда трудно копировать файлы между различными типами компьютеров одного и того же разработчика.

Среди проблем, с которыми обычно приходится сталкиваться при работе в многосистемном сетевом окружении, можно отметить следующие:

- Различные правила именования файлов
- Различные правила перемещения по каталогам файловой системы
- Ограничения на доступ к файлам
- Различные способы представления текста и данных внутри файлов

Разработчики стека протоколов TCP/IP старались найти не слишком сложное решение этих проблем и создали достаточно общий, но очень элегантный *протокол пересылки файлов* (File Transfer Protocol — FTP), который легко обслуживается и прост в использовании.

Протокол FTP создан для взаимодействия с интерактивным конечным пользователем или прикладной программой. Мы ограничимся рассмотрением интерактивных служб этого протокола для конечного пользователя, всегда доступных во всех реализациях TCP/IP.

Пользовательский интерфейс разработан для клиента пересылки файлов операционной системы Berkeley Unix (BSD) и далее перенесен на различные типы многопользовательских компьютеров. В этой главе мы рассмотрим диалоги конечного пользователя с текстовым интерфейсом, а также несколько графических интерфейсов для настольных компьютеров.

Основные функции пересылки файлов разрешают пользователю копировать файлы между системами, просматривать списки каталогов и выполнять файловые операции, подобные переименованию или удалению. Все эти функции являются частью стандартного стека протоколов TCP/IP.

В конце главы мы проанализируем *простейший протокол пересылки файлов* (Trivial File Transfer Protocol — TFTP), использующийся в базовых операциях по переносу файлов в определенных ситуациях, например при загрузке программного обеспечения в маршрутизаторы, мосты или бездисковые рабочие станции.

14.2 Общедоступный и личный доступ FTP

Компьютерные системы обычно требуют от пользователя идентификатор регистрации и пароль до того, как разрешить пользователю просматривать или манипулировать файлами. Однако иногда полезно создать возможность работы с общедоступными файлами. FTP обеспечивает как общедоступное совместное использование информации, так и частный доступ к файлам, предлагая два вида услуг:

- Доступ к общедоступным файлам через анонимную регистрацию
- Доступ к личным файлам, разрешенный только для пользователя с системным идентификатором регистрации и паролем

14.2.1 Вводный диалог

Представленный ниже диалог демонстрирует копирование из сайта AT&T InterNIC Data Services (общедоступного репозитория документов RFC).

Сегодня многие имеют на своих настольных системах графические пользовательские интерфейсы (GUI) для пересылки файлов. С одним из таких интерфейсов мы познакомимся ниже. Однако текстовый интерфейс позволяет лучше понять происходящие в процессе пересылки файлов события, поэтому сначала мы познакомимся с подключением к InterNIC через текстового клиента.

Архив файлов InterNIC доступен для всех, так что при регистрации мы будем вводить идентификатор *ftp*. Традиционно обращение к общедоступным системам происходило через идентификатор *анонимного* (anonymous) *доступа*. В настоящее время больше применяется *ftp*, который легче напечатать. Общедоступные серверы для пересылки файла предполагают ввод пользователем адреса электронной почты в качестве пароля.

Приглашение *ftp* > выводится всякий раз, когда локальное приложение FTP ожидает ввода данных от пользователя. Строки, начинающиеся с чисел, содержат сообщения от удаленного файлового сервера.

`> ftp ftp.internic.net`	Команда ftp запускает пользовательский интерфейс программы-клиента FTP. Пользователь хочет соединиться с удаленным хостом *ftp.internic.net*.
`Connected to ftp.ds.internic .net.`	Локальный клиент FTP отчитывается об успешном соединении.
`220- InterNIC Directory and Database Services`	Это сообщение пришло от удаленной системы.
`220- . . .`	Мы опустим приветствие.
`220 ds.internic.net FTP server ready.`	
`Name (ftp.internic.net:sfeit) : ftp`	Локальная клиентская программа FTP запрашивает ввод идентификатора пользователя. Для InterNIC нужно ввести *ftp*.
`331 Guest login ok, send ident as password.`	
`Password:`	Локальная клиентская программа FTP запрашивает пароль. Вежливый ответ подразумевает ввод идентификатора электронной почты.
`230 Guest login ok, access restrictions apply.`	
`Ftp>`	Это приглашение запрашивает ввод команд.
`ftp> cd rfc`	Пользователь переходит в удаленный каталог *rfc*, в котором и хранятся документы RFC.
`250 CWD command successful.`	Команда изменения каталога (cd) пересылается на сервер как CWD (изменить рабочий каталог). Каталог сервера изменяется на rfc, и можно начинать копирование документов RFC.
`ftp> get rfc1842.txt myrfc`	Запрашивается копирование файла *rfc1842.txt*, для чего будет создано второе соединение.
`200 PORT command successful.`	Локальный клиент FTP получил второй порт и послал на сервер команду PORT, указывая серверу на соединение через этот порт.
`150 Opening ASCII mode data connection for rfc1842.txt (24143 bytes).`	Открытие соединения для пересылки файла.
`226 Transfer complete.`	Завершение пересылки файла.
`local: newfile remote: rfc1842.txt`	Создан новый локальный файл.
`24818 bytes received in 0.53 seconds (46 Kbytes/s)`	
`ftp> quit`	Завершение сеанса.
`221 Goodbye.`	

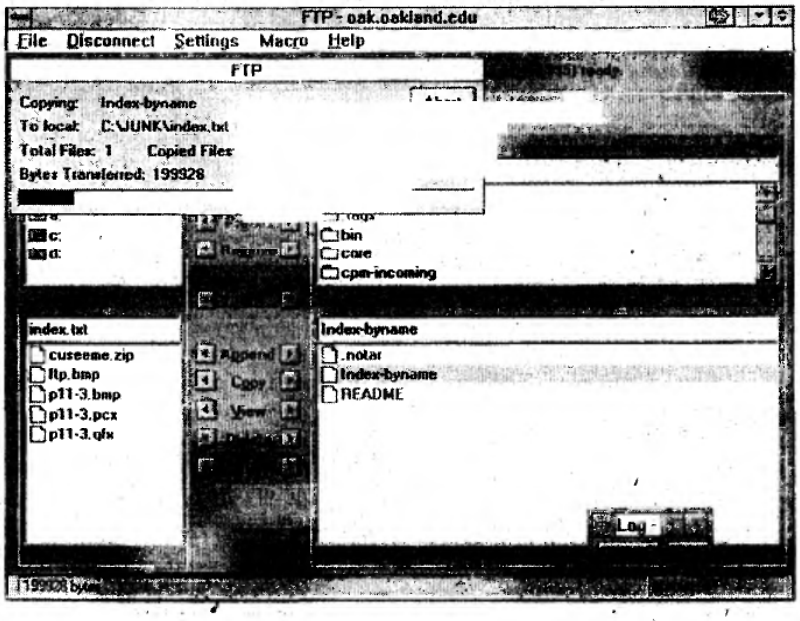

Рис. 14.1. Доступ к архиву пересылки файлов из программы *Chameleon*

Первая команда запрашивала у сервера переход в каталог *rfc*. Затем проведено копирование удаленного документа *rfc1842.txt* в локальный файл, названный *myrfc*. Если не вводить имя файла, локальный файл получит то же имя, что и удаленный файл.

FTP позволяет записывать имена удаленных файлов так же, как это делают пользователи удаленного хоста. Копируя файл на локальный компьютер, можно присвоить ему локальное имя файла. Если имя не присваивается, то при необходимости FTP преобразует имя удаленного файла в формат, допустимый для локального хоста. Иногда это приводит к преобразованию символов из нижнего регистра в верхний и к усечению имен.

Протокол FTP имеет характерный стиль операций. Всякий раз, когда должен быть скопирован файл, для пересылки данных открывается и используется второе соединение. После команды *get* (получить) в приведенном примере диалога локальный клиент FTP получает второй порт и указывает серверу на открытие соединения с этим портом. Мы не видели команду, инициирующую эту операцию, но видели ответную реакцию:

```
200 PORT command successful.
150 Opening ASCII mode data connection for rfc1842.txt (24143 bytes).
```

На рис. 14.1 показан доступ к другому общедоступному архиву, но через приложение для пересылки файлов *Chameleon* (в среде Windows), имеющее графический пользовательский интерфейс.

Файлы могут копироваться перетаскиванием их значков из одного окна в другое или щелчком мыши на кнопке со стрелкой. Имя локального файла можно ввести в окне слева, расположенном ниже метки Files.

К тому же самому сайту можно обратиться и из клиента пересылки файлов *Netscape* (см. рис. 14.2). Копирование файла выполняется щелчком мыши на его имени. Текстовые файлы выводятся на экран, и их можно сохранить на локальном компьютере через пункт *Save* меню *File*. Если запрашивается локальное сохранение двоичного файла, то выводится раскрывающееся меню с запросом о месте хранения этого файла.

14.3 Модель FTP

Как видно из приведенного выше диалога, пользователь взаимодействует с *локальным клиентом* FTP (точнее, с соответствующим процессом). Программное обеспечение локального клиента управляет преобразованием данных для *удаленного сервера* FTP через *управляющее соединение*. Когда конечный пользователь вводит команду пересылки или работы с файлом, эта команда транслируется в одно из специальных сокращений, используемых для управляющего соединения.

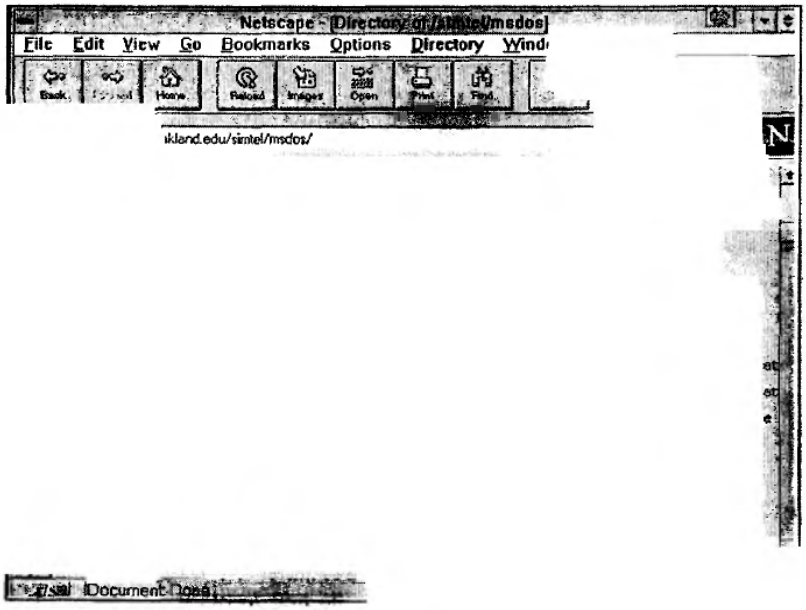

Рис. 14.2. Доступ к архиву пересылки файлов из *Netscape*

В сущности, управляющее соединение — это обычный сеанс *telnet* в режиме NVT. Клиент отправляет команду на сервер через управляющее соединение, а сервер возвращает ответ по этому же соединению.

Когда пользователь запрашивает пересылку файла, открывается отдельное соединение для передачи данных, и по нему пересылается файл. Это соединение используется и для пересылки содержимого каталогов. Модель FTP показана на рис. 14.3. Обычно сервер использует порт 20 для соединения пересылки данных.

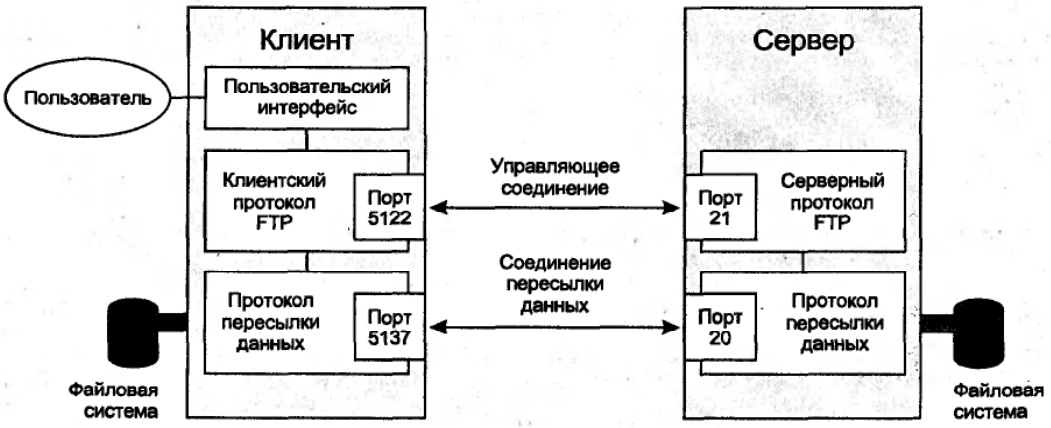

Рис. 14.3. Управляющее соединение и соединение пересылки данных в FTP

Во время вышерассмотренного диалога конечный пользователь вводил запросы на изменение удаленного каталога и пересылку файла. Эти запросы преобразовывались в формат команд FTP и пересылались по управляющему соединению на удаленный сервер FTP. Пересылка файлов производится по отдельному соединению, создаваемому для обмена данными.

14.4 Команды FTP

Какие команды можно передавать по управляющему соединению? Существуют команды аутентификации, дающие возможность пользователю указать идентификатор, пароль и регистрационную запись для работы с FTP.

Команды пересылки файлов позволяют:

- Копировать одиночный файл между хостами
- Копировать несколько файлов между хостами
- Добавлять содержимое локального файла к удаленному файлу
- Копировать файл и добавлять к его имени номер для формирования уникального имени (например, файлы ежедневной регистрации получат имена log.1, log.2 и т.д.)

Команды обслуживания файлов разрешают:

- Просмотреть список файлов каталога
- Узнать текущий каталог и изменить его на другой
- Создавать и удалять каталоги
- Переименовывать или удалять файлы

Управляющие команды служат для:

- Идентификации пересылки файлов ASCII, EBCDIC или двоичных файлов
- Проверки структурирования файла (как последовательность байт или как последовательность записей)
- Указания способа пересылки файла (например, как поток октетов)

Пересылаемые по управляющему соединению команды имеют стандартный формат. Например, команда *RETR* используется для копирования файла из сервера на сайт клиента.

FTP не накладывает ограничений на пользовательский интерфейс, поэтому разработчики могут создавать (как мы уже видели) хитроумные системы для настольных компьютеров либо простые в применении клиентские программы. Т.е. ввод с клавиатуры get, перетаскивание мышью значка или щелчок на имени файла транслируются в одну и ту же команду *RETR*.

Пользовательский интерфейс обычно имеет дополнительные команды для настройки локального окружения, например:

- Запросить FTP о выводе звукового сигнала при завершении пересылки файла
- Для текстового интерфейса запросить вывод символа диез (#) при пересылке каждого блока данных
- Установить автоматическое преобразование регистра символов в имени файла или таблицу трансляции символов

Полный набор поддерживаемых конкретным хостом функций можно узнать через справку клиента FTP или в техническом описании программы.

14.4.1 Использование команд в текстовом диалоге

Многие пользователи предпочитают графический интерфейс, доступный на настольных системах, но текстовый интерфейс позволяет лучше понять внутренние процессы протокола FTP.

Нижеприведенный текстовый диалог начинается с вывода справки. Существующие команды имеют синонимы, например *ls* и *dir* — для запроса сведений о каталоге, *put* и *send* — для копирования файла на удаленный хост, *get* и *recv* — для получения файла от удаленного хоста или *bye* и *quit* — для выхода из FTP.

Используя команды *mget* или *mput* и глобальные подстановочные символы можно одновременно копировать несколько файлов. Например, *mget* a* извлечет копии каждого файла с именем, начинающимся на букву a. Такой режим включается параметром *glob*, который разрешает или запрещает применение *глобальных* подстановочных символов.

В представленный ниже диалог включен вывод отладочной информации, чтобы дать некоторое представление о работе протокола:

- Строки, начинающиеся на -->, показывают сообщения, посланные локальным хостом по управляющему соединению.
- Строки, начинающиеся с числа, соответствуют сообщениям, посланным удаленным сервером для отчета о результате выполнения команды.

```
plum-feit > ftp
ftp> help
Commands may be abbreviated. Commands are:

!           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

ftp> debug
Debugging on (debug = 1).

ftp> open tigger.jvnc.net
Connected to tigger.jvnc.net.
220 tigger.jvnc.net FTP server (Version wu-2.4(1) Fri Apr 15 13:54:36 EDT 1994)
ready.
```

Для обращения к личным файлам введены реальные идентификатор пользователя (userid) и пароль.

```
Name (tigger.jvnc.net:sfeit): feit
--> USER feit
331 Password required for feit.
Password:
--> PASS abcd1234
230 User feit logged in.
```

Команда *status* (статус) показывает текущие параметры сеанса FTP. Многие из них будут рассмотрены ниже. Пока отметим, что *тип данных* (Type) указан как ASCII. При пересылке текстовых файлов FTP часто предполагает это значение по умолчанию.

```
ftp> status
Connected to tigger.jvnc.net.
No proxy connection.
Mode: stream; Type: ascii; Form: non-print; Structure: file
Verbose: on; Bell: off; Prompting: on; Globbing: on
Store unique: off; Receive unique: off
Case: off; CR stripping: on
Ntrans: off
Nmap: off
Hash mark printing: off; Use of PORT cmds: on
```

Затем запрашивается список файлов каталога. Такой список может быть очень большим, поэтому FTP посылает его по соединению для данных:

```
ftp> dir
```

FTP необходим порт для пересылки данных. Клиент посылает команду *PORT*, которая идентифицирует его IP-адрес (4 байта) и новый порт (2 байта), чтобы использовать эти значения при пересылке данных. Байты преобразуются в десятичный формат и разделяются запятыми. IP-адрес 128.36.4.22 будет записан как 128,36,4,22, а порт 2613 — как 10,53.

```
--> PORT 128,36,4,22,10,53
200 PORT command successful.
```

Сервер откроет соединение по указанному адресу socket. Команда *LIST* — это формальное сообщение для запроса подробного списка файлов каталога:

```
--> LIST
```

Далее сервер открывает соединение с объявленным клиентом портом:

```
150 Opening ASCII mode data connection for /bin/ls.
total 531
-rw-r--r-- 1 feit    tigers       0 Oct 24 1994   .addressbook
-rw-r--r-- 1 feit    tigers    2808 Sep 23 1994   .article
-rw-r--r-- 1 feit    tigers     397 Mar 14 1993   .cshrc
. . .
-rw-r--r-- 1 feit    tigers    3113 Jul 31 13:29 subnets
-rw-r--r-- 1 feit    tigers   59901 Jun  5 17:48 typescript
226 Transfer complete.
2239 bytes received in 0.31 seconds (7 Kbytes/s)
```

Сразу после пересылки списка файлов соединение данных будет закрыто. Затем мы можем получить файл.

```
ftp> get subnets
```

Клиент указывает новый адрес socket для переноса файла. Отметим, что на сей раз используется клиентский порт 2614 (10,54).

```
--> PORT 128,36,4,22,10,54
200 PORT command successful.
--> RETR subnets
150 Opening ASCII mode data connection for subnets (3113 bytes).
226 Transfer complete.
```

По завершении пересылки файла соединение для данных закрывается.

```
local: subnets remote: subnets
3187 bytes received in 0.27 seconds (11 Kbytes/s)
ftp> quit
--> QUIT
221 Goodbye.
plum-feit>
```

Отметим, что сценарий для соединения данных был таким:

- Локальный клиент получил новый порт и использовал управляющее соединение, чтобы сообщить серверу FTP номер своего порта.
- FTP-сервер связался с новым портом данных клиента.

- Данные были переданы.
- Соединение было закрыто.

Можно применять альтернативный сценарий. Если клиент посылает команду *PASV*, сервер возвращает номер порта и переходит к прослушиванию установки соединения данных от клиента. Ранее преобладало использование команды *PORT*. Однако теперь клиент может послать команду *PASV* для пересылки файлов через простую систему защиты (firewall), которая не разрешает установку соединений из поступающих сообщений (этот вариант будет подробно рассмотрен чуть позже).

При работе с файлами большого размера иногда обнаруживается, что пересылается не тот файл. Хорошая реализация должна позволять отменить пересылку. Для текстового интерфейса это обычно делается через комбинацию клавиш CONTROL-C, а в графическом интерфейсе — специальной кнопкой *Abort* (остановить).

14.5 Типы данных, структуры файлов и методы пересылки

На обоих концах соединения необходимо обеспечить единый формат для пересылаемых данных. Этот файл текстовый или двоичный? Он структурирован по записям или по блокам?

Для описания формата пересылки используются три атрибута: *тип данных* (data type), *структура файла* (file structure) и *режим пересылки* (transmission mode). Допустимые значения этих атрибутов рассмотрены ниже. В общем случае применяются:

- Пересылка текста ASCII или двоичных данных.
- Неструктурированный файл, который рассматривается как последовательность байт.
- Режим пересылки рассматривает файл как поток байт.

Однако есть и несколько исключений. Некоторые хосты структурируют текстовые файлы как последовательность записей. Хосты IBM используют для текстовых файлов кодирование EBCDIC и проводят обмен файлами как набором структурированных блоков, а не как потоком байт.

В следующих разделах мы рассмотрим различные варианты типов данных, структур файлов и методов их пересылки.

14.5.1 Типы данных

Файл может содержать текст ASCII, EBCDIC или двоичный образ данных (существует еще тип, называемый *локальным* или *логическим байтом* и применяемый для компьютеров с размером байта в 11 бит). Текстовый файл может содержать обычный текст или текст, форматированный для вывода на принтер. В последнем случае в нем будут находиться коды вертикального форматирования:

- Символы вертикального форматирования *Telnet* для режима NVT (т.е. <CR>, <LF>, <NL>, <VT>, <FF>)
- Символы вертикального форматирования ASA (ФОРТРАН)

Типом данных по умолчанию является нераспечатываемый текст ASCII (т.е. текст без управляющих символов форматирования.— *Прим. пер.*). Тип данных может быть изменен стандартной командой *TYPE*, пересылаемой по управляющему соединению.

14.5.2 Пересылка текста ASCII

Хотя текст ASCII является стандартным, компьютеры интерпретируют его по-разному из-за различия в кодах конца строки. Системы Unix используют для этого <LF>, компьютеры PC — <CR><LF>, а Macintosh — <CR>.

Для устранения этих различий FTP превращает локальный текстовый файл ASCII в формат NVT, а приемник преобразует NVT ASCII в собственный локальный формат. Например, если текстовый файл копируется с системы Unix на PC, все коды концов строк (в Unix — <LF>) при получении файла на PC нужно преобразовать в <CR><LF>.

14.5.3 Пересылка текста EBCDIC

Поддерживающие кодировку EBCDIC хосты обеспечивают весьма полезную команду пользовательского интерфейса, инициирующую пересылку по управляющему соединению команды *TYPE E*. Текстовые символы EBCDIC пересылаются по соединению в своем обычном 8-разрядном формате. Строки завершаются символом новой строки EBCDIC (<NL>).

14.5.4 Пересылка двоичных данных

С пересылки текстов ASCII легко переключиться на двоичный образ данных. В текстовом пользовательском интерфейсе для этого служит команда *binary*, а в графическом — командная кнопка binary (двоичные данные). Клиент меняет тип пересылаемых данных командой *TYPE I*, передаваемой по управляющему соединению.

Что произойдет, если пользователь забудет переключить тип данных с ASCII на двоичный при копировании двоичного файла? Хорошие реализации FTP предупредят, что задана ошибочная операция, и позволят до начала пересылки файла изменить тип данных. К сожалению, многие реализации идут еще дальше и "помогают" изменять все двоичные байты, которые выглядят как символы конца строк (исправляя их на специальные заполнители или полностью удаляя их из текста). Некоторые действительно плохие реализации все же начинают пересылку файла и аварийно завершаются в середине выполнения такой операции.

14.5.5 Структуры файлов

В FTP поддерживаются две структуры (ранее использовалась также *страничная структура* для файлов DEC TOPS-20, сейчас устаревшая):

- *Файловая структура*, соответствующая неструктурированному файлу, который рассматривается как последовательность байт.
- *Структура записей*, которая применяется для файлов, состоящих из последовательности записей.

Более распространена *файловая структура*, которая применяется по умолчанию. Перейти на *структуру записей* можно стандартной командой *STRU R*, пересылаемой по управляющему соединению.

14.5.6 Режимы пересылки

Режим пересылки и структура файла определяют, как будут форматированы данные для обмена по соединению. Существуют три режима пересылки: *stream* (поток), *block* (блочный режим) и *compressed* (сжатые данные).

- В режиме потока и файловой структуры файл передается как поток байт. FTP возлагает на TCP обеспечение целостности данных и не включает в данные никаких заголовков или разделителей. Единственным способом указания на конец файла будет нормальное завершение соединения для данных.
- Для режима потока и структуры записей каждая запись отделяется 2-байтовым управляющим кодом конца записи (End Of Record — EOR), а конец файла отмечается символами конца файла (End Of File — EOF). EOR кодируется как X'FF 01, а EOF — X'FF 02. Для последней записи файла EOR и EOF записываются как X'FF 03. Если файл содержит байт данных из одних единиц, то такой байт представляется при пересылке как X'FF FF.
- В блочном режиме файл пересылается как последовательность блоков данных. Каждый блок начинается 3-байтовым заголовком (см. рис. 14.4).
- Режим сжатия данных используется крайне редко, поскольку обеспечивает очень неудачный метод архивирования, разрушающий последовательность повторяющихся байт. Обычно пользователю проще применить одну из более удачных программ сжатия, широко доступных на современных компьютерах, и далее пересылать полученный архивный файл как двоичные данные.

8 бит	16 бит
Дескрипторные флаги	Счетчик байт
Конец блока — EOR Конец блока — EOF Restart Marker	Количество следующих далее байт

Рис. 14.4. Формат заголовка блочного режима пересылки FTP

Блок может содержать целую запись, или в записи объединяются несколько блоков. Дескриптор содержит:

- Флаг End Of Record для идентификации границы записи
- Флаг End Of File, который указывает, является ли блок последним при пересылке файла
- Флаг Restart Marker (маркер перезапуска), указывающий, содержит ли данный блок текстовую строку, которую можно использовать для указания точки перезапуска после неудачной пересылки файла в более поздней точке

Режим потока наиболее распространен и используется по умолчанию. Изменить его на блочный режим можно стандартной командой *MODE B*, пересылаемой по управляющему соединению.

Преимущество структуры записей или блочного режима состоит в том, что будет явно отмечен конец файла и после завершения его пересылки можно сохранить соединение для данных, а следовательно, использовать его для нескольких пересылок.

В показанном ранее диалоге ответ на команду *status* содержал:

```
Mode: stream; Type: ascii; Form: non-print; Structure: file
```

Т.е. по умолчанию был установлен поточный режим пересылки данных, тип данных ASCII без форматирования для печати и файловая структура (соответствующая неструктурированному файлу).

14.6 Протокол FTP

С протоколом FTP связаны следующие понятия:

- Команды и их параметры, пересылаемые по управляющему соединению
- Числовые коды, возвращенные в ответ на команду
- Формат пересылаемых данных

Ниже рассмотрен набор команд FTP. Они передаются по управляющему соединению. За последние годы набор команд существенно увеличился, однако хостам необязательно реализовывать все спецификированные команды.

Иногда локальный пользовательский интерфейс не поддерживает команды непосредственно, а оставляет их реализацию для удаленного хоста. Хорошая реализация FTP обеспечивает команду *quote* (цитата), которая позволяет вводить нужную команду в ее стандартном виде. Введенные пользователем символы далее пересылаются по управляющему соединению без каких-либо преобразований. Такой способ полезен, когда пользователю известны стандартные команды и их параметры.

14.6.1 Команды управления доступом

Команды и параметры, которые определяют доступ пользователя к хранилищу файлов удаленного хоста, определены в таблице 14.1.

Таблица 14.1

Команды авторизации пользователя для доступа к архиву файлов

Команда	Определение	Параметр(ы)
USER	Идентифицирует пользователя	Идентификатор пользователя
PASS	Ввод пароля	Пароль
ACCT	Указание регистрационной записи пользователя	Идентификатор регистрационной записи
REIN	Повторная инициализация для указания состояния	Нет
QUIT	Выход	Нет
ABOR	Отмена предыдущей команды и запущенной этой командой пересылки данных	Нет

14.6.2 Команды управления файлами

Команды из таблицы 14.2 дают возможность выполнять типичные операции позиционирования на каталог и управления файлами удаленного хоста. Рабочим каталогом (working directory) называется текущий каталог пользователя.

Таблица 14.2

Команды выбора каталога и управления файлами

Команда	Определение	Параметр(ы)
CWD	Перейти в другой каталог сервера	Имя каталога
CDUP	Перейти в родительский каталог	Нет
DELE	Удалить файл	Имя файла
LIST	Вывести информацию о файлах	Имя каталога, список файлов (без параметра — вывод информации о рабочем каталоге)
MKD	Создать каталог	Имя каталога
NLST	Вывести список файлов каталога	Имя каталога (для рабочего каталога может отсутствовать)
PWD	Вывести имя рабочего каталога	Нет
RMD	Удалить каталог	Имя каталога
RNFR	Указать файл, который будет переименован	Имя файла
RNTO	Переименовать файл	Имя файла
SMNT	Монтировать другую файловую систему	Идентификатор (Identifier)

14.6.3 Команды установки формата данных

Команды из таблицы 14.3 используются для указания формата данных, структуры файла и режима пересылки, которые будут применяться при копировании файлов.

Таблица 14.3

Команды описания типа, структуры и режима

Команда	Определение	Параметр(ы)
TYPE	Указание типа данных и необязательного формата вывода на принтер	A (ASCII), E (EBCDIC), I (двоичный образ), N (не распечатываемые), T (*telnet*), C (ASA)
STRU	Структура файла	F (файл) или R (записи)
MODE	Формат пересылки	S (поток), B (блок) или C (сжатие)

14.6.4 Команды пересылки файлов

Команды из таблицы 14.4 применяются с целью установки соединения для данных, копирования файлов и восстановления при перезапуске.

Таблица 14.4

Команды поддержки пересылки файлов

Команда	Определение	Параметр(ы)
ALLO	Выделяет (резервирует) достаточное пространство для поступающих данных	Целое число байт
APPE	Добавляет локальный файл в конец удаленного файла	Имя файла
PASV	Запрашивает у сервера IP-адрес и порт для инициализируемого клиентом соединения пересылки данных.	Нет. Сервер возвратит IP-адрес и номер порта

Таблица 14.4 (*продолжение*)

Команды поддержки пересылки файлов

Команда	Определение	Параметр(ы)
PORT	Идентифицирует сетевой адрес и номер порта для инициируемого сервером соединения	IP-адрес и номер порта
REST	Устанавливает маркер перезапуска (вводится сразу за перезапускаемой командой пересылки)	Значение маркера
RETR	Извлечение или получение файла	Имя файла (файлов)
STOR	Сохранение или помещение файла	Имя файла (файлов)
STOU	Сохранение файла с уникальным именем	Имя файла

14.6.5 Дополнительные команды

Последний набор команд (таблица 14.5) выводит конечному пользователю полезную информацию.

Таблица 14.5

Дополнительные информационные команды

Команда	Определение	Параметр(ы)
HELP	Вывод сведений о реализованных на сервере возможностях	Нет
NOOP	Запрос от сервера ответа OK	Нет
SITE	Используется для специфичных серверных подкоманд, которые не стандартизованы, но могут быть доступны на данном сервере	Нет
SYST	Запрос к серверу о типе его операционной системы	Нет
STAT	Запрос информации о параметрах и состоянии соединения	Нет

14.6.6 Команды сайта

Многие файловые серверы Unix используют программное обеспечение WU-FTP от Вашингтонского университета (Сент-Луис). Эта реализация имеет команду *SITE* для выполнения на файловом сервере различных специальных программ. Например, пользователь может сначала получить доступ по идентификатору *ftp*, а затем указать в команде *SITE* регистрационный идентификатор группы и пароль. В этом случае обеспечивается доступ к большему числу файлов, чем при анонимном доступе.

14.6.7 Восстановления после ошибок и перезапуск

Многим организациям необходимо пересылать очень большие файлы. Предположим, что во время пересылки такого файла произошла ошибка. Возникшие проблемы должна помочь решить служба перезапуска FTP. Она не является обязательной и, к сожалению, на момент написания книги такую службу обеспечивали только немногие продукты TCP/IP. Однако будем оптимистами и рассмотрим возможности службы перезапуска.

В блочном режиме работы FTP и при реализации службы перезапуска пересылающая информацию сторона может передавать блоки, содержащие в нужных местах общего потока данных маркеры перезапуска. Каждый маркер представляет собой распечатываемую строку текста. Например, последовательные маркеры могли бы быть: 1, 2, 3 и т.д. Всякий раз, когда приемник получает маркер, он записывает принятые данные на энергонезависимое устройство хранения и отслеживает положение маркера в общем потоке данных.

Если информацию принимает клиент, о получении каждого маркера будет информироваться конечный пользователь (как только данные были сохранены в локальной системе). Если данные получает удаленный сервер, пользователю по управляющему соединению будет возвращено сообщение, указывающее, что данные до маркера были успешно сохранены на сервере.

При отказе системы пользователь может возобновить выполнение команды, указав значение маркера как аргумент команды. Эта операция должна быть инициирована сразу после команды, во время выполнения которой произошел крах системы.

14.6.8 Коды ответов

Каждой команде в диалоге соответствует ответ, состоящий из кода ответа и сообщения. Например:

```
ftp> get subnets
--> PORT 128,36,0,22,10,54
200 PORT command successful.
--> RETR subnets
150 Opening ASCII mode data connection for subnets (3113 bytes).
226 Transfer complete.
```

Коды ответов состоят из трех цифр, каждая из которых имеет определенное назначение:

- Коды от 200 до 300 указывают на успешное выполнение команды.
- Коды от 100 до 200 указывают на начало выполнения операции.
- Коды от 300 до 400 указывают на успешное достижение промежуточной точки.
- Коды от 400 до 500 сигнализируют о временной ошибке.
- Коды от 500 свидетельствуют о постоянной ошибке (это плохие новости).

Вторая и третья цифры кодов более точно специфицируют ответ.

14.7 Безопасность

14.7.1 Проверка имен хоста клиента

Иногда пользователи сталкиваются с невозможностью анонимного доступа к файловому архиву. Если это происходит не часто, то обычно является следствием загруженности сервера. Однако если доступ невозможен постоянно, значит есть проблемы с именем домена.

Некоторые файловые серверы запрещают доступ клиентам, которые не перечислены в базе данных DNS. Сервер FTP может выполнять обратный поиск для всех входных IP-адресов. Если такого адреса нет в базе данных DNS — доступ блокируется. Единственным решением такой проблемы может быть обращение к администратору DNS для включения имени системы в базу данных. Некоторые серверы производят *двойную проверку* — транслируют адрес клиента в имя, а затем полученное имя опять в адрес для сравнения его с исходным адресом запроса. Благодаря этому исключаются обращения с подстановочными символами для элементов DNS.

14.7.2 PASV или PORT?

Организации обеспечивают безопасность своих сетей через средства защиты (firewall), применяющие к датаграммам определенный критерий фильтрации и ограничивающие входящий трафик. Часто простейшие средства защиты разрешают пользователям локальной сети инициировать соединение, но блокируют все попытки создания соединения извне.

Исходная спецификация FTP определяет команду *PORT* как средство по умолчанию для установки соединения данных. В результате многие реализации основывают установку соединения только на этой команде. Однако команда *PORT* требует открытия соединения от внешнего файлового сервера к клиенту, что обычно блокируется средством защиты локальной сети.

К счастью, новые реализации поддерживают команду *PASV*, указывающую серверу на выделение нового порта для соединения данных с пересылкой IP-адреса и номера порта сервера в ответе клиенту. Далее клиент может самостоятельно открыть соединение с сервером.

14.7.3 Промежуточные прокси

Некоторые организации создают более изощренные системы безопасности. Каждый запрос реально пересылается на промежуточный прокси, реализующий систему защиты локальной сети. Прокси становится единственной системой, которая будет видна из внешнего мира. Для работы через прокси клиент предоставляет:

- Имя или IP-адрес прокси
- Идентификатор пользователя и пароль для получения доступа к прокси

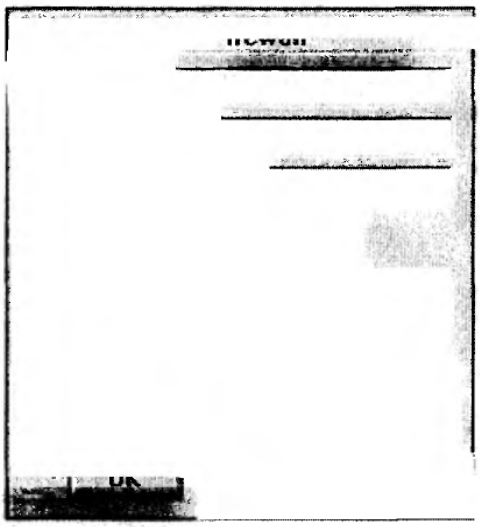

Рис. 14.5. Конфигурирование клиента для работы через средство защиты

- Номер порта для доступа к прокси пользователям пересылки файлов (необязательно порт 21)
- Дополнительную информацию, зависящую от конкретной реализации данного прокси-агента

На рис. 14.5 показан конфигурационный экран клиентского средства защиты. После ввода данных пользователь сможет работать с приложениями обычным образом. Промежуточные процессы не видны конечному пользователю (хотя это и зависит от типа прокси). Некоторые средства защиты требуют от локальных пользователей ввода идентификатора и пароля при доступе через средство защиты до того, как начнется реальная пересылка транзакций.

14.8 Замечания о производительности

На эффективность операций пересылки файлов влияют следующие факторы:

- Файловая система хоста и производительность его дисков
- Объем обработки по переформатированию данных
- Используемая служба TCP

Краткий отчет о пропускной способности приводится в конце каждой пересылки файла:

```
226 Transfer comlete
local: rfc1261 remote: rfc1261
4488 bytes sent in 0.037 seconds (1.2e + 02 Kbytes/s)
```

Средние значения производительности FTP и TCP можно получить при пересылке больших файлов.

14.9 Trivial File Transfer Protocol

Некоторым приложениям копирования файлов требуются очень простые реализации, например для начальной загрузки программного обеспечения и конфигурационных файлов в маршрутизаторы, концентраторы или бездисковые рабочие станции.

Простейший протокол пересылки файлов (Trivial File Transfer Protocol — TFTP) используется как очень полезное средство копирования файлов между компьютерами. TFTP передает данные в датаграммах UDP (при реализации в другом стеке протоколов TFTP должен запускаться поверх службы пакетной доставки данных). Для этого не потребуется слишком сложное программное обеспечение — достаточно только IP и UDP. Особенно полезен TFTP для инициализации сетевых устройств (маршрутизаторов, мостов или концентраторов).

Характеристики TFTP:

- Пересылка блоков данных размером в 512 октетов (за исключением последнего блока)
- Указание для каждого блока простого 4-октетного заголовка
- Нумерация блоков от 1
- Поддержка пересылки двоичных и ASCII октетов
- Возможность чтения и записи удаленных файлов
- Отсутствие ограничений по аутентификации пользователей

Один из партнеров по TFTP пересылает нумерованные блоки данных одинакового размера, другой партнер подтверждает их прибытие сигналом ACK. Отправитель ожидает ACK для посланного блока до того, как пошлет следующий блок. Если за время тайм-аута не поступит ACK, выполняется повторная отправка того же самого блока. Аналогично, если к получателю не поступят данные за время тайм-аута, он отправляет еще один ACK.

14.9.1 Протокол TFTP

Сеанс TFTP начинается запросами *Read Request* (запрос чтения) или *Write Request* (запрос записи). Клиент TFTP начинает работу после получения порта, посылая Read Request или Write Request на порт 69 сервера. Сервер должен идентифицировать различные номера портов клиентов и использовать их для последующей пересылки файлов. Он направляет свои сообщения на порт клиента. Пересылка данных производится как обмен блоками данных и сообщениями ACK.

Каждый блок (за исключением последнего) должен иметь размер в 512 октетов данных и завершаться EOF (конец файла). Если длина файла кратна 512, то заключительный блок содержит только заголовок и не имеет никаких данных. Блоки данных нумеруются от единицы. Каждый ACK содержит номер блока данных, получение которого он подтверждает.

14.9.2 Элементы данных протокола TFTP

В TFTP существуют пять типов элементов данных:

- Read Request (RRQ, запрос чтения)
- Write Request (WRQ, запрос записи)

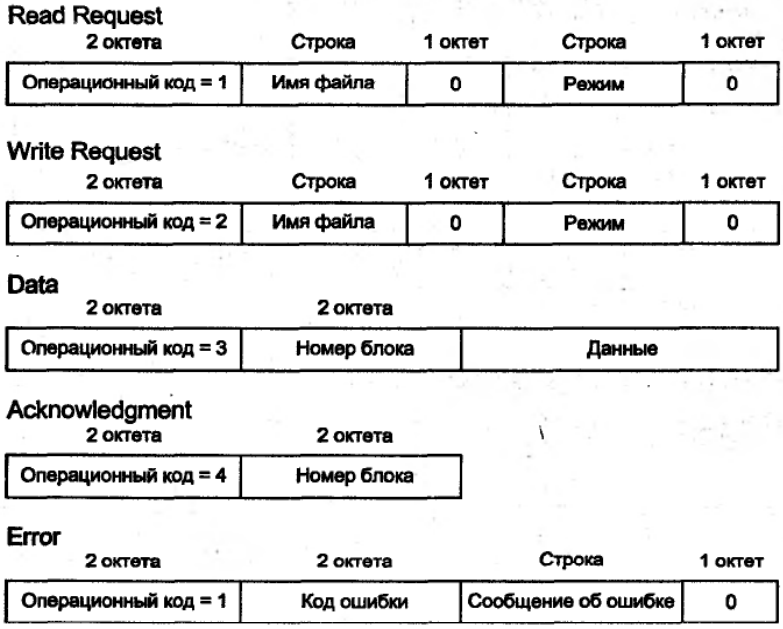

Рис. 14.6. Форматы элементов данных TFTP

- Data (DATA, данные)
- Acknowledgment (ACK, подтверждение)
- Error (ERROR, ошибка)

Сообщение об ошибке указывает на события, подобные таким: "файл не найден" или "для записи файла на диске нет места".

Каждый заголовок TFTP начинается операционным кодом, идентифицирующим тип элемента данных протокола (Protocol Data Unit — PDU). Форматы PDU показаны на рис. 14.6.

Отметим, что длина Read Request и Write Request меняется в зависимости от длины имени файла и полей режима, каждое из которых представляет собой текстовую строку ASCII, завершенную нулевым байтом. В поле режима могут присутствовать netascii (сетевой ASCII) или octet (октет).

14.9.3 Варианты TFTP

Улучшенный вариант TFTP разрешает согласование параметров через предварительные запросы чтения и записи. Его основная цель — позволить клиенту и серверу согласовывать между собой размер блока, когда он больше 512 байт (для увеличения эффективности пересылки данных).

14.9.4 Сценарий TFTP

Работу протокола TFTP можно проиллюстрировать простым сценарием. На рис. 14.7 показано, как в TFTP реализуется чтение удаленного файла. После отправки запрашиваемой стороной блока данных она переходит в режим ожидания ACK на посланный блок и, только получив этот ACK, посылает следующий блок данных.

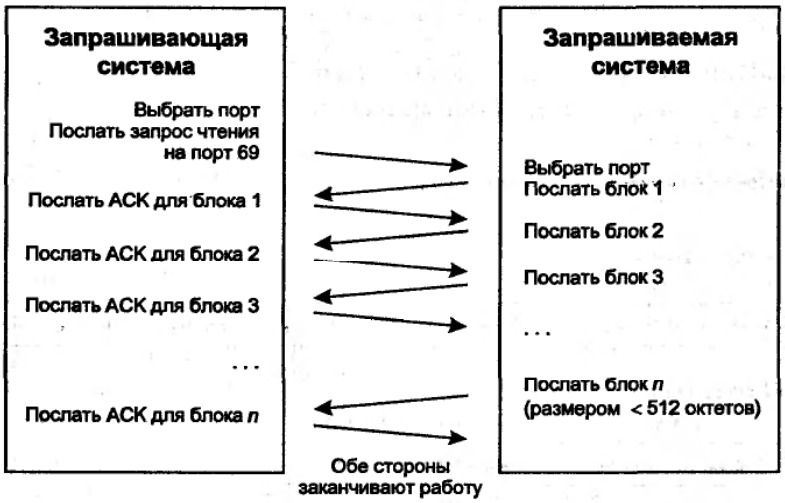

Рис. 14.7. Чтение удаленного файла в TFTP

14.10 Дополнительная литература

Протокол FTP определен в RFC 959, а TFTP — в RFC 1350.

Глава
15

RPC и NFS

15.1 Введение

За последние десять лет компьютерное оборудование существенно изменилось. Вместо подключенных к центральному компьютеру неинтеллектуальных терминалов появились сложные настольные системы, серверы и локальные сети.

Пользователи быстро поняли преимущества персональных систем, но вместе с тем возникла необходимость доступа к общесетевой информации и совместно используемым, или разделяемым, принтерам. Это привело к появлению должности сетевого администратора — лица, ответственного за конфигурирование, обслуживание и резервное копирование. Современный системный администратор должен координировать переход на новые версии программного обеспечения, отслеживать использование ресурсов, планировать резервное копирование информации и конфигурировать сетевые параметры большого числа компьютеров.

За несколько лет многие организации пришли к необходимости перевода сетевых операционных систем в режим разделения ресурсов и централизации управления. Чуть позже вычисления клиент/сервер подняли уровень сетевого взаимодействия до прикладных приложений.

15.1.1 Назначение NFS

Компания Sun разработала *сетевую файловую систему* (Network File System — NFS) для поддержки разделения ресурсов служб рабочих станций Unix в локальных сетях. NFS делает удаленный каталог с файлами частью локальной структуры каталогов — конечные пользователи и программы обращаются к удаленным файлам так же, как и к файлам из каталогов локально подключенного диска. NFS дает множество преимуществ.

Например, на сервере можно хранить единственную копию программного обеспечения или важных данных, которая доступна всем пользователям сети. Изменения будут проводиться в одном месте (на сервере), а не на каждой из рабочих станций пользователей. На рис. 15.1 показана локальная сеть с одним центральным сервером, обеспечивающим службу NFS.

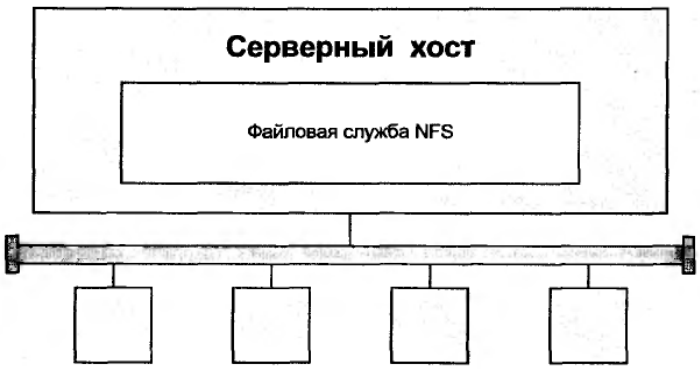

Рис. 15.1. Сервер NFS в локальной сети

15.1.2 Соотношения между NFS, RPC и XDR

NFS работает поверх *вызовов удаленных процедур* (Remote Procedure Call — RPC). RPC был разработан в начале использования приложений клиент/сервер. В этой главе мы познакомимся со службами NFS и *архитектурой открытых сетевых вычислений* (Open Network Computing — ONC), на основе которой реализуется RPC.

Стандарт *внешнего представления данных* (eXternal Data Representation — XDR) является важной частью архитектуры RPC. XDR включает язык описания типов данных и методы их кодирования в стандартный формат. Это позволяет производить обмен данными между компьютерами различных типов, например между хостами Unix, PC, Macintosh, системами VAX VMS компании Digital Equipment Corporation и большими ЭВМ компании IBM.

15.1.3 RPC как стандарт Интернета

Компания Sun Microsystems опубликовала RFC с описанием RPC в 1988 г., а NFS — в 1989 г. Однако Sun контролировала эти протоколы вплоть до 1995 г., пока не появились новые версии. С этого момента ответственность за RFC архитектуры ONC перешла к комитету IETF, а сама архитектура была принята в качестве стандарта для процессов Интернета. Sun взаимодействовала с консорциумом X/Open при разработке новой версии NFS.

15.1.4 Реализации NFS и RPC

NFS и RPC были реализованы многими разработчиками систем Unix, а также перенесены во многие лицензированные операционные системы. Например, IBM VM, IBM MVS и DEC VAX VMS могут работать как файловые серверы NFS.

Некоторые разработчики объединили программное обеспечение клиента и сервера NFS с собственными продуктами TCP/IP, в то время как другие предоставляли NFS за дополнительную плату. Во многих продуктах NFS содержится программная библиотека для RPC.

Множество продуктов TCP/IP для Windows обеспечивает работу системы Windows в качестве клиента NFS, а некоторые реализации — и как серверы NFS. Последние версии NetWare компании Novell поддерживают NFS вместе с собственными службами файлов и печати. Любой клиент может обращаться к серверу по любому из этих протоколов. В частности, поддерживаются клиенты DOS, Macintosh и Unix.

15.2 Модель RPC

Приложение клиент/сервер для архитектуры ONC функционирует поверх RPC. Работа RPC моделируется обычными вызовами подпрограмм. Например, в языке программирования C вызов обычной подпрограммы в общем случае имеет форму:

```
код_возврата = имя_процедуры (входные_параметры, выходные_параметры)
```

Перед активизацией процедуры входные данные сохраняются как входные_параметры. Если процедура завершается успешно, полученные результаты сохраняются в выходных_параметрах. По завершении код_возврата указывает на успешность работы процедуры.

RPC работает аналогичным образом. Локальная система посылает запрос вызова на удаленный сервер. Запрос идентифицирует процедуру и получает входные параметры. Удаленный сервер выполняет процедуру. По завершении работы удаленный сервер формирует ответ, указывающий на успешность процедуры и содержащий ее выходные параметры. На рис. 15.2 показан обмен запросом и ответом. Протокол RPC определяет механизм данного способа работы.

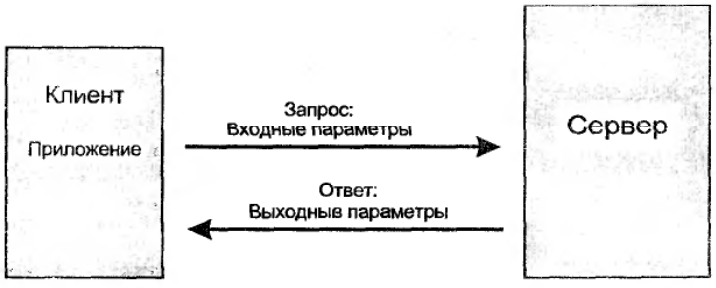

Рис. 15.2. Взаимодействие в RPC

15.3 Программы и процедуры RPC

Основные концепции RPC достаточно просты:

- Служба RPC реализуется одной или несколькими выполняющимися на сервере *программами*. Например, существуют отдельные программы управления доступом и блокировок файлов.
- Каждая программа может выполнять несколько *процедур*. Идея состоит в том, что процедура должна реализовывать одну простую, четко ограниченную функцию. Например, существуют отдельные процедуры файлового доступа NFS для операций чтения, записи, переименования и удаления файлов.
- Каждой программе присвоен числовой идентификатор.
- Каждая процедура программы также имеет числовой идентификатор.

На момент написания книги выделением уникальных номеров для программ занималась компания Sun Microsystems (в будущем это должно перейти под юрисдикцию IANA). Диапазоны идентификаторов программ показаны в таблице 15.1. Числовой идентификатор присваивается процедурам программы разработчиком этой программы. Например, процедура *чтения* NFS — 6, а *переименования* NFS — 11.

Таблица 15.1

Присваивание номеров в RPC

0-1fffffff	Определяются компанией Sun (rpc@sun.com)
20000000-3fffffff	Номера только для использования внутри сайта
40000000-5fffffff	Для приложений, динамически генерирующих номера программ
60000000-7fffffff	Зарезервировано
80000000-9fffffff	Зарезервировано
a0000000-bfffffff	Зарезервировано
c0000000-dfffffff	Зарезервировано
e0000000-ffffffff	Зарезервировано

Запрос клиента RPC идентифицирует запускаемую программу и процедуру по ее номеру. Например, чтобы прочитать файл, запрос RPC обратится к программе 100003 (*NFS*) и процедуре 6 (*чтение*). На рис. 15.3 показано клиентское приложение, обращающееся к удаленной процедуре программы 100003.

Опыт показывает, что через какое-то время программы меняются. Процедуры дорабатываются, и их становится все больше. По этой причине запрос RPC должен указывать версию программы. Очень часто на хосте сервера одновременно работает несколько версий одной программы RPC.

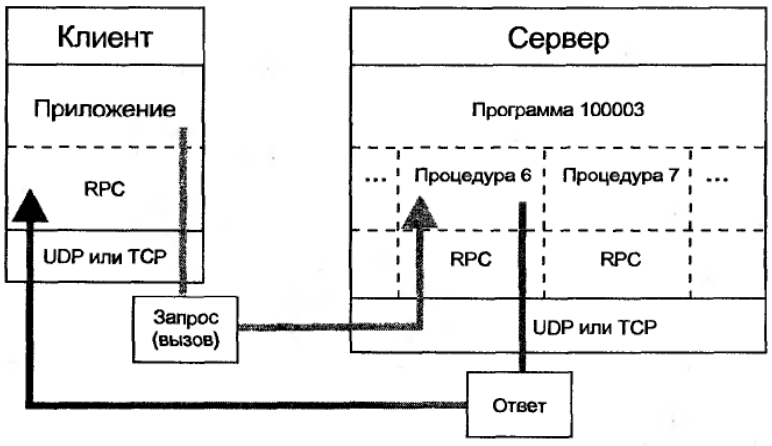

Рис. 15.3. Доступ к удаленной процедуре из клиентского приложения

Удаленный запрос к процедуре (RPC) послан от клиента серверу в форматированном сообщении. RPC не заботится о том, какой транспортный протокол используется для пересылки сообщения. В мире TCP/IP RPC может работать поверх UDP или TCP, но можно использовать и другой транспорт.

Хотя обычно предполагается взаимодействие клиента с уникальным сервером, запросы RPC могут передаваться в многоадресных или широковещательных рассылках.

15.4 Типичная программа RPC

Наиболее известной программой RPC является NFS. Соответствующая команда *mount* (монтировать) позволяет клиенту подключить к своей локальной файловой системе удаленный каталог. Эта команда также является программой RPC. Существуют *lock manager* (диспетчер блокировки) и программа *status*, которые обеспечивают основу для изменения пользователем разделяемых файлов на сервере NFS.

Spray (распыление) — пример очень простой программы RPC. Клиент *spray* посылает серию сообщений к удаленной системе и получает ответ. Представленная ниже команда посылает 100 датаграмм хосту *plum* (эта программа позволяет получить статистику пересылки группы сообщений.— *Прим. пер.*):

```
> spray -c 100 plum
sending 100 packets of lnth 86 to plum ...
in 10.1 seconds elapsed time,
29 packets (29.00%) dropped by plum
Sent: 9 packets/sec, 851 bytes/sec
Rcvd: 7 packets/sec, 604 bytes/sec
```

Программа *rusers* выясняет, кто зарегистрирован на хостах из указанного списка или на всех хостах локальной сети. Клиент *rusers* отправляет запрос RPC через широковещательные рассылки локальной сети. Ответы содержат имена хостов и список пользователей, зарегистрированных на каждом из них.

```
> rusers
Zonker.num.cs.yale.edu     leonard     jones harris
Mark.num.cs.yale.edu       davis       sherman
Duke.num.cs.yale.edu       burry       victor
. . .
```

15.5 Работа с дубликатами запросов RPC

Если служба основана на протоколе TCP, запросы и ответы будут доставляться надежно. TCP берет на себя обеспечение целостности доставляемых данных.

Если RPC базируется на UDP, то, в зависимости от требований конкретного приложения, клиент и сервер должны обеспечить собственный тайм-аут, повторную пересылку и стратегию выделения дублированных сообщений. Разработчик приложения может выбрать для клиента любую из следующих стратегий:

- Если в пределах тайм-аута не будет получен ответ, послать сообщение об ошибке конечному пользователю, который и должен снова инициировать запрос к службе.
- Если в пределах тайм-аута не будет получен ответ, отправить запрос еще раз. Повторять эту операцию до тех пор, пока не будет получен ответ или не будет достигнут максимальный предел повторной пересылки.

Если клиент повторно посылает запрос, разработчик должен реализовать на сервере стратегию обработки дубликатов сообщений. Сервер может:

- Не фиксировать ранее выполненные операции. При поступлении запроса выполнить процедуру, даже если это был дубликат запроса. Отметим, что для некоторых процедур (например, чтение набора байт из файла) в этом нет ничего страшного. Конечно, клиент может и дальше получать двойные ответы, но может и блокировать их, отслеживая ранее выполненные транзакции.
- Хранить копии ответов, которые были отправлены в течение нескольких последних минут. При поступлении запроса с тем же операционным идентификатором сервер уже знает, что процедура выполнена и на нее уже был послан ответ, следовательно, он мог бы отослать назад копию исходного ответа. Если сервер выполняет затребованную процедуру в момент поступления дубликата запроса — он должен отбросить повторный запрос.

Каждое приложение клиент/сервер может выбрать стратегию соединения, наиболее подходящую своим конкретным требованиям.

15.6 Portmapper в RPC

Уже разработано много программ клиент/сервер. А будет написано их еще больше. Предоставление каждому приложению общеизвестных портов ограничено — как же клиенты смогут распознавать все большее количество служб?

15.6.1 Назначение Portmapper

Архитектура RPC предоставляет метод для динамического обнаружения присвоенного службе порта. На каждом серверном хосте специальная программа RPC работает как хранилище данных о *других* программах RPC этого сервера. Такая программа называется *portmapper* (отображение портов) либо в более новых версиях операционных систем — *rpcbind* (связывание в RPC). В этой главе мы будем именовать такую программу *portmapper*, подразумевая, что *rpcbind* обеспечивает аналогичные функции.

Portmapper поддерживает следующие элементы:

- Локальные активные программы RPC
- Номера версий этих программ
- Транспортный протокол или протокол обмена
- Порты, через которые работают программы

Программа *portmapper* запускается после инициализации сервера RPC на компьютере. Как показано на рис. 15.4, после запуска программы RPC операционная система предоставляет этой программе один из неиспользованных портов и сообщает *portmapper*, что данная программа готова к работе, т.е. в *portmapper* происходит регистрация порта, номера программы и ее версии.

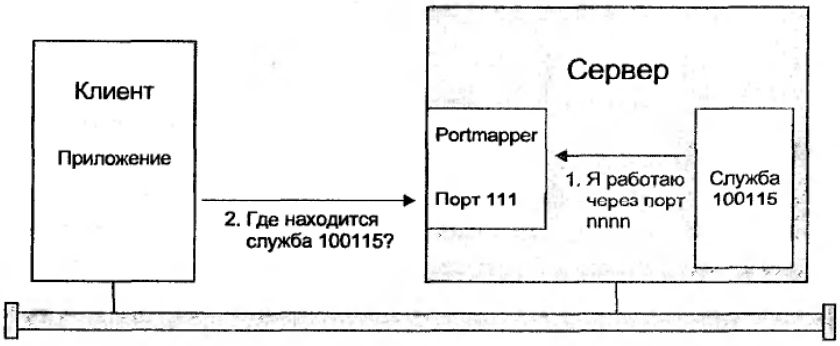

Рис. 15.4. Поиск порта службы через portmapper

Portmapper (или *rpcbind*) отслеживает запросы к общеизвестному порту 111. Когда клиенту требуется доступ к службе, он посылает запрос в сообщении RPC на порт 111 (т.е. к *portmapper*). В запросе указывается номер программы требуемой службы, ее версия и протокол пересылки (UDP или TCP). В ответе от *portmapper* клиенту возвращается текущий номер порта требуемой службы.

Кроме того, *portmapper* обеспечивает отдельные функции RPC через широковещательные рассылки. В этом случае клиент отправляет запрос RPC по одной из своих связей. Например, команда *rusers* из RPC через широковещательную рассылку запрашивает каждую из машин локальной сети о зарегистрированных на ней пользователях.

Отметим, что программа *rusers* на каждом из хостов может работать через различные порты. Какой номер порта должен поместить клиент в сообщение запроса для оправки в широковещательную рассылку?

Дело в том, что клиент вставляет свой запрос в специальный вызов косвенного запроса (*indirect request*) к *portmapper* и посылает такой запрос на порт 111. *Portmapper* пересылает полученный запрос к службе и затем возвращает ответ службы клиенту. Номер порта службы включается в ответ, чтобы последующие запросы клиента могли быть посланы непосредственно к службе, а не к portmapper.

15.6.2 Процедуры portmapper

Выполняемые программой *portmapper* процедуры перечислены в таблице 15.2.

Таблица 15.2

Процедуры portmapper

Процедура	Описание
PMAPPROC_NULL	Возвращает ответ, указывающий на активное состояние portmapper.
PMAPPROC_SET	Используется при регистрации службы (т.е. при включении в список активных служб сервера локальной программы, ее версии, протокола и номера порта).
PMAPPROC_UNSET	Применяется для отмены регистрации службы (например, при удалении локальной программы из списка активных служб сервера).
PMAPPROC_GETPORT	Используется клиентом для поиска номера порта сервера. Входными параметрами являются специальный номер программы, версия программы и транспортный протокол (UDP или TCP).
PMAPPROC_DUMP	Возвращает список всех локальных программ RPC, их версий, коммуникационных протоколов и портов (используется в *rpcinfo -p*).
PMAPPROC_CALLIT	Пересылка поступающего от клиента косвенного запроса к локальной программе RPC. При успешном завершении процедуры возвращает ответ, включая номер порта программы. Предназначен для использования при широковещательных запросах.

15.6.3 Просмотр служб RPC через portmapper

Команда *rpcinfo* из Unix выводит полезную информацию о программах RPC, посылая запрос RPC к *portmapper*. Аналогичную программу обеспечивают и другие операционные системы с поддержкой клиентов RPC.

Приведенный ниже результат работы *rpcinfo -p* содержит сведения о программах RPC, работающих на хосте *bulldog.cs.yale.edu* (т.е. был послан запрос к *процедуре PMAPPROC_DUMP* программы *portmapper*).

Результат работы команды показывает номера программ, их версии, транспортный протокол, порт и идентификатор для каждой программы сервера. Видно, что в списке находится и сама программа *portmapper* (в самом верху списка):

```
> rpcinfo -p bulldog.cs.yale.edu
Program     vers      proto       port
100000        2        tcp        111       portmapper
100000        2        udp        111       portmapper
100029        1        udp        657       keyserv
100005        1        udp        746       mountd
100005        2        udp        746       mountd
100005        1        tcp        749       mountd
100003        2        udp       2049       nfs
100005        2        tcp        749       mountd
100026        1        udp        761       bootparam
100024        1        udp        764       status
100024        1        tcp        766       status
100021        1        tcp        767       nlockmgr
100021        1        udp       1033       nlockmgr
100021        3        tcp        771       nlockmgr
100021        3        udp       1034       nlockmgr
100020        1        udp       1035       llockmgr
```

```
100020      1       tcp        776      llockmgr
100021      2       tcp        779      nlockmgr
100021      2       udp       1036      nlockmgr
100011      1       udp       1070      rquotad
100001      2       udp       1111      rstatd
100001      3       udp       1111      rstatd
100001      4       udp       1111      rstatd
100002      1       udp       1124      rusersd
100002      2       udp       1124      rusersd
100012      1       udp       1127      sprayd
100008      1       udp       1132      walld
```

Отметим интересный момент: для определения состояния приложения RPC использовалось другое приложение Remote Procedure Call.

Команда *rpcinfo -b* выполняет широковещательную рассылку в сети, запрашивая все работающие серверы о выполняемых ими программах и версиях этих программ. В приведенном ниже примере запрашиваются сведения о версии 1 программы *spray* под номером 100012.

```
> rpcinfo -b 100012 1
128.36.12.1 casper.na.cs.yale.edu
128.36.12.28 tesla.math.yale.edu
128.36.12.6 bink.na.cs.yale.edu
```

Каждая программа RPC имеет пустую процедуру с номером 0, возвращающую только ответ "Я активна". Нижеприведенная команда *rpcinfo -u* посылает сообщение пустой процедуре программы *spray* хоста *bulldog.cs.yale.edu*:

```
> rpcinfo -u bulldog.cs.yale.edu 100012
program 100012 version 1 ready and waiting
```

15.7 Программа rpcbind

В последних версиях RPC программа *portmapper* заменена на *rpcbind*. Исходная программа *portmapper* связывалась с UDP или TCP. *Rpcbind* независима от используемого транспортного протокола. Эта программа возвращает строку ASCII, содержащую адресную информацию, которая не зависит от используемого транспорта и называется *форматом универсального адреса* (universal address format).

15.7.1 Назначение rpcbind

Программа *rpcbind* основана на тех же принципах, что и *portmapper*. При инициализации программы RPC ей выделяется один или несколько динамически назначенных адресов для транспорта. Программа регистрирует полученные адреса в rpcbind, через которую они становятся известными клиентам.

Запрос клиента содержит номер программы и номер версии. Но в ответе *rpcbind* указывается универсальный адрес, который может предоставлять специальные сведения для NetWare SPX/IPX, SNA, DECnet или AppleTalk, а не для TCP или UDP. Тип предоставленного в ответе транспортного адреса зависит от используемого для запроса транспортного протокола.

Как и к *portmapper*, к *rpcbind* обращаются по общеизвестному порту 111 через UDP или TCP. Для других коммуникационных протоколов должен использоваться другой, заранее определенный локальный доступ.

Подобно *portmapper*, *rpcbind* поддерживает службу широковещательных рассылок RPC. Рассылки направляются на общеизвестную точку доступа к транспорту, определенную для службы *rpcbind*, например порт 111 для UDP или TCP. Каждая программа *rpcbind*, которая отслеживает широковещательные рассылки, может от имени клиента вызвать нужную ей локальную сервисную программу, получить ответ и переслать его клиенту. Версия 4 протокола RPC позволяет клиентам получать через *rpcbind* такой же вид косвенного обслуживания при многоадресной рассылке, как и при широковещательной.

15.7.2 Процедуры rpcbind

Процедуры программы *rpcbind* версии 4 представлены в таблице 15.3.

Таблица 15.3

Процедуры rpcbind

Процедура	Описание
RPCBPROC_SET	Используется службой регистрации программ через локальную RPCBIND.
RPCBPROC_UNSET	Используется для отмены регистрации локальной программы.
RPCBPROC_GETADDR	Возвращает клиенту универсальный адрес программы.
RPCBPROC_GETVERSADDR	В запрос включается нужный номер версии программы.
RPCBPROC_GETADDRLIST	Выводит список адресов программы. Клиент может выбрать из нескольких доступных транспортных протоколов.
RPCBPROC_DUMP	Список всех элементов базы данных RPCBIND (например, предоставление сведений для вывода командой *rpcinfo*).
RPCBPROC_BCAST	Поддержка широковещательного запроса — RPCBIND пересылает запрос локальной программе.
RPCBPROC_INDIRECT	Поддержка косвенных запросов, которые являются многоадресными — RPCBIND пересылает их локальной программе и возвращает назад результат или сообщение об ошибке.
RPCBPROC_GETTIME	Возвращает местное время сервера, отсчитанное в секундах от полночи первого дня января 1970 г.
RPCBPROC_*UADDR2TADDR*	Преобразование универсальных адресов в адреса, специфичные для данного транспорта.
RPCBPROC_TADDR2UADDR	Преобразование специфичных для транспорта адресов в универсальные адреса.
RPCBPROC_GETSTAT	Предоставление статистики о количестве и типах полученных запросов.

15.8 Сообщения RPC

Клиент RPC посылает запросы серверу и получает ответы на них в специальных сообщениях. Что должны содержать эти сообщения, чтобы клиент и сервер поняли друг друга?

Необходим идентификатор транзакции, определяющий соответствие между запросом и ответом. Запрос клиента должен указывать программу и процедуру, которую он хочет запустить. Клиенту необходим некоторый способ идентифицировать себя через мандат (credentials), доказывающий право использования службы. Наконец, запрос клиента должен содержать входные параметры. Например, запрос *чтения* NFS должен идентифицировать файл и количество читаемых байтов.

В дополнение к сообщению о результатах успешных запросов серверу необходим способ сообщения клиенту об отмене запроса и причинах такой отмены. Запрос может быть отклонен при несоответствии версий программы или ошибке при аутентификации клиента. Сервер должен сообщить об ошибках в параметрах или событиях, например: "Не могу найти файл".

На рис. 15.5 показано взаимодействие клиента с программой сервера. Клиент посылает запрос. Когда работа затребованной процедуры завершается, серверная программа возвращает ответ. Как видно из рис. 15.5, запрос включает:

- Идентификатор транзакции
- Текущий номер версии RPC
- Номер программы
- Версию программы
- Номер процедуры

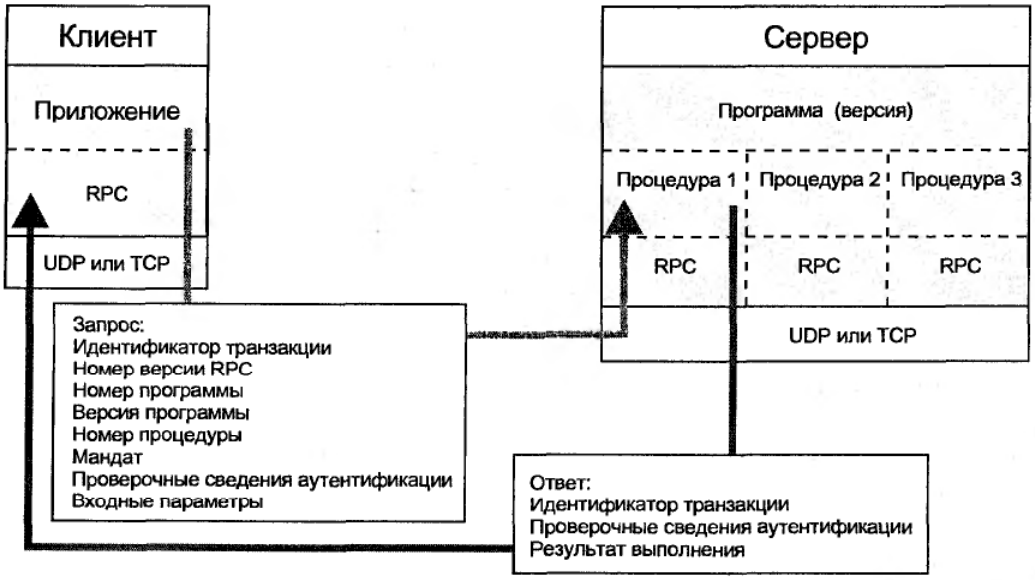

Рис. 15.5. Сообщения RPC

- Мандат аутентификации
- Проверочные сведения (verifier) аутентификации
- Входные параметры

Если процедура выполнена успешно, ответ содержит результаты. Если при выполнении выявлены проблемы, ответ будет содержать информацию об ошибках.

15.9 Аутентификация в RPC

Некоторые службы не нуждаются в защите. Для вывода времени дня на сервере служба RPC может быть оставлена открытой для общего доступа. Однако клиент, обращающийся к личным данным, должен обеспечить некоторую опознавательную информацию (проходить аутентификацию). В некоторых случаях важно, чтобы сервер также подтверждал свою подлинность. Не следует посылать номер своей кредитной карточки через систему интерактивных заказов, если не будет гарантии в подлинности сервера. Таким образом, в некоторых случаях аутентификационные данные должен предоставлять как клиент, так и сервер (они будут как в запросах, так и в ответах).

В сообщении запроса аутентификационные сведения RPC пересылаются в двух полях:

- Поле *мандата* (credentials) — содержит идентификационную информацию.
- Поле *проверочных сведений аутентификации* (verifier) — содержит дополнительную информацию и позволяет проверить идентификатор. Например, verifier мог бы содержать зашифрованный пароль и штамп времени.

Для аутентификации не существует единого стандарта. Реализовать проверку аутентификации должен разработчик каждого приложения. Именно он должен решить, что необходимо в каждом конкретном случае. Однако продолжаются работы по созданию единых стандартов для этой процедуры.

В настоящее время каждый метод аутентификации называется *flavor* (оттенок). Тип такого оттенка, используемый в мандатах или полях verifier, идентифицирован целым числом в начале поля. Новые типы аутентификации могут быть зарегистрированы таким же образом, что и новые программы. Поля мандата и verifier начинаются с целого числа.

15.9.1 Нулевая аутентификация

Нулевая аутентификация полностью соответствует своему названию. Не используется никакой аутентификационной информации — в полях мандата и verifier сообщений запросов и ответов содержатся одни нули.

15.9.2 Аутентификация систем

Аутентификация систем моделирует аналогичную информацию операционной системы Unix. Мандат системы содержит:

stamp (штамп)	Случайный идентификатор, сгенерированный вызывающим компьютером
machinename (имя машины)	Имя запрашивающей машины
uid (идентификатор пользователя)	Реальный номер идентификации пользователя, инициировавшего запрос
gid (идентификатор группы)	Реальный номер идентификации группы пользователя, инициировавшего запрос
gids (идентификаторы групп)	Список групп, к которым принадлежит пользователь

Поле проверочных сведений аутентификации — нулевое.

Проверочные сведения аутентификации (verifier), возвращаемые сервером, могут быть пустыми или иметь оттенок short (краткие), означающий возвращение октета, определяющего систему. В некоторых реализациях этот октет используется как мандат в последующем сообщении от вызывающей стороны (мандат будет заменять информацию о пользователе и его группе).

Отметим, что этот способ не обеспечивает защиты. Следующие два метода применяют шифрование для защиты аутентификационной информации. Однако приходится выбирать между обеспечением безопасных служб RPC и достижением удовлетворительной производительности. Шифрование даже одного поля приводит к существенному снижению быстродействия службы, например NFS.

15.9.3 Аутентификация DES

Стандарт шифрования данных (Data Encryption Standard — DES) использует симметричный алгоритм шифрования. DES — это федеральный стандарт обработки информации (Federal Information Processing Standard — FIPS), который был определен Национальным бюро стандартов США, в настоящее время называемым Национальным институтом стандартов и технологий (National Institute of Standards and Technology — NIST).

Аутентификация DES для RPC основана на сочетании асимметричных общедоступных и личных ключей с симметричным шифрованием DES:

- Имя пользователя связано с общедоступным ключом.
- Сервер шифрует ключ сеанса DES с помощью общедоступного ключа и посылает его клиентскому процессу пользователя.
- Ключ сеанса DES используется для шифрования аутентификационной информации клиента и сервера.

15.9.4 Аутентификация в Kerberos

При аутентификации в системе Kerberos (по имени трехглавого сторожевого пса Цербера из древнегреческой мифологии.— *Прим. пер.*) используется сервер безопасности Kerberos, хранящий ключи пользователей и серверов (основанные на паролях). Kerberos аутентифицирует службу RPC с помощью:

- использования секретных ключей (клиента и сервера), зарегистрированных на сервере безопасности Kerberos и распространяемых как ключи сеансов DES для клиентов и серверов
- применения ключа сеанса DES для шифрования аутентификационной информации клиента и сервера

15.10 Пример сообщений RPC версии 2

На рис. 15.6 показан результат обработки монитором *Sniffer* компании Network General заголовка UDP и полей RPC из сообщения запроса к NFS о выводе атрибутов файла. Заголовки уровня связи данных и IP опущены, чтобы не загромождать рисунок.

```
UDP:  -- UDP Header --
UDP:
UDP:  Source port = 1023 (Sun RPC)
UDP:  Destination port = 2049
UDP:  Length = 124
UDP:  No checksum
UDP:
RPC:  -- SUN RPC header --
RPC:
RPC:  Transaction id = 641815012
RPC:  Type = 0 (Call)
RPC:  RPC version = 2
RPC:  Program = 100003 (NFS), version = 2
RPC:  Procedure = 4 (Look up file name)
RPC:  Credentials: authorization flavor = 1 (Unix)
RPC:  len = 32, stamp = 642455371
RPD:  machine = atlantis
RPC:  uid = 0, gid = 1
RPC:  1 other group id(s) :
RPC:  gid 1
RPC:  Verifier: authorization flavor = 0 (Null)
RPC:  [Verifier: 0 byte(s) of authorization data]
RPC:
RPC:  [Обычное завершение заголовка "SUN RPC".]
RPC:
NFS:  -- SUM NFS --
NFS:
NFS:  [Параметры для процедуры 4 (Look up file name) follow]
NFS:  File handle = 0000070A00000001000A0000000091E3
NFS:  5E707D6A000A0000000044C018F294BE
NFS:  File name = README
NFS:
NFS:  [Обычное завершение "SUN NFS".]
NFS:
```

Рис. 15.6. Формат сообщения RPC с запросом к NFS

Заметим, что запрос RPC имеет *тип сообщения 0*. Ответ будет иметь *тип 1*. Протокол RPC периодически обновляется, поэтому в сообщении указывается версия RPC (в нашем случае это версия 2).

Вызывающая сторона использует *мандат Unix*, определяющий реальные идентификаторы пользователя и группы (userid и groupid). Имеется дополнительный идентификатор группы. *Штампом* служит произвольный идентификатор, созданный вызывающей стороной. Поле *проверочных сведений аутентификации* имеет оттенок 0 (не обеспечивает никакой дополнительной информации). NFS часто реализуется с частичной аутентификацией, поскольку более полная защита снижает производительность.

За идентификатором программы 100003 (NFS) и процедуры 4 (просмотр имен файлов) следуют параметры: *описатель файла* (file handle) и имя файла.

Описатель файла — это специальный идентификатор, связанный с каталогом или файлом сервера. В версии 2 протокола RPC описатель файла представлен строкой фиксированной длины в 32 бита, в версии 3 он задается строкой переменной длины с максимальной длиной в 64 бита. В запросе указан файл *README*, расположенный в каталоге, идентифицированном описателем файла.

Поля в сообщении запроса кодируются по правилам форматирования XDR (см. следующий раздел).

Мы можем получить представление о работе XDR, рассмотрев некоторые шестнадцатеричные коды в сообщении запроса:

Тип сообщения = 0, кодируется (в шестнадцатеричных значениях) как:

```
00 00 00 00
```

Версия RPC = 2, кодируется как:

```
00 00 00 02
```

Машина = atlantis, кодируется как:

```
(длина строки =8)  a  t  l  a  n  t  i  s
  00 00 00 08      61 74 6C 61 6E 74 69 73
```

```
RPC:  -- SUN RPC header --
RPC:
RPC:  Transaction id = 641815012
RPC:  Type = 1 (Reply)
RPC:  Status = 0 (Accepted)
RPC:  Verifier: authorization flavor = 0 (Null)
RPC:  [Verifier: 0 byte(s) of authorization data]
RPC:  Accept status = 0 (Success)
RPC:
RPC:  [Обычное завершение заголовка "SUN RPC" .]
RPC:
NFS:  -- SUN NFS --
NFS:
NFS:  Proc = 4 (Look up file name)
NFS:  Status = 0 (OK)
NFS:  File handle = 0000070A00000001000A000000005AC9
NFS:                3298621C000A0000000044C018F294BE
NFS:  File type = 1 (Regular file)
NFS:  Mode = 0100644
NFS:       Type = Regular file
NFS:       Owner's permissions = rw-
NFS:       Group's permissions = r-
NFS:       Others; permissions = r-
NFS:  Link count = 1, UID = 303, GID = 1
NFS:  File size = 130, Block size = 8192, No. of blocks = 2
NFS:  File system id = 1802, File id = 23241
NFS:  Access time = 23-Oct-95 16:35:01 GMT
NFS:  Modification time = 20-Oct-95 12:10:43 GMT
NFS:  Inode change time = 20-Oct-95 12:10:43 GMT
NFS:
NFS:  [Обычное завершение "SUN NFS".]
NFS:
```

Рис. 15.7. Формат сообщения RPC с ответом от NFS

В показанном на рис. 15.7 ответе присутствует тот же идентификатор транзакции. Указана нулевая аутентификационная информация. Запрос был принят, и его обработка завершилась успешно. Ответ содержит много полезной информации о файле *README*:

- *Идентификатор описателя файла.* Любые дальнейшие операции с этим файлом будут использовать указанный в ответе описатель.

- *Режим* (mode) описывает тип файла и указывает, кто может получить доступ к этому файлу (владелец, группа или любой пользователь). Режим объявляет, может ли пользователь читать или записывать файл. Если файл представляет собой прикладное программное обеспечение, режим показывает, могут ли пользователи запускать такое приложение.
- Имеются дополнительные атрибуты файла, например его размер, время последнего обращения и обновления. Можно ожидать, что эти атрибуты поддерживаются в любой файловой системе.

15.11 XDR

Как будут взаимодействовать разнородные (гетерогенные) машины в окружении клиент/сервер, чтобы понимать посылаемые друг другу данные? Например, клиент NFS может захотеть, чтобы сервер прочитал в файле 1000 байтов данных от некоторой позиции. Как должны кодироваться параметры такого запроса? Типичными параметрами являются имя файла или имя каталога, равно как и атрибуты файла (размер файла и целые значения, точно определяющие количество байт или текущее смещение в файле).

Все параметры в сообщениях Sun RPC *определены* и *кодируются* по протоколу *представления внешних данных* (eXternal Data Representation — XDR). Этот протокол специфицирует:

- Язык описания данных XDR, определяющий тип данных в запросах и ответах
- Правила кодирования XDR по форматированию данных при пересылке

Большая часть библиотеки программирования RPC состоит из запросов, которые преобразуют типы данных в/из сетевого формата XDR.

15.11.1 Язык описания данных XDR

Описания данных XDR похожи на описания данных в языках программирования и не являются слишком сложными. Существует несколько основных типов данных XDR: целые числа со знаком и без знака, последовательные (или порядковые) целые числа, строки ASCII, логические значения и числа с плавающей точкой. Для пересылки строк октетов общего вида применяется тип данных *opaque* (непрозрачный, без преобразования). В полях с этим типом данных может пересылаться зашифрованная информация. К более сложным типам данных относятся массивы, структуры и объединенные типы данных (union datatypes), построенные на основе базовых типов.

Порядковые целые числа позволяют присвоить значения каждому элементу короткого списка целых чисел. Простым примером порядкового целого числа является *тип сообщения* (msg_type), определяющий, будет ли сообщение запросом или ответом:

```
enum msg_type {
  CALL = 0,
  REPLY = 1
  };
```

В данном поле может появляться только одно из целых чисел: 0 или 1. Ввод любого другого целого числа приведет к ошибке.

Структура определения тела сообщения запроса RPC:

```
struct call_body {
  unsigned int rpcvers;     /* Версия должна быть равна 2   */
  unsigned int prog;        /* Это номер программы          */
  unsigned int vers;        /* Это версия программы         */
  unsigned int proc;        /* Определение процедуры        */
  opaque_auth cred;         /* Мандат, например userid      */
  opaque_auth verf;         /* Verifier для мандата         */
                  /* Здесь может быть зашифрованное поле   */
  /* начало описания специфичных для процедуры параметров  */
```

15.11.2 Кодирование в XDR

Сообщения запросов и ответов для данной версии программы или процедуры имеют фиксированный формат. Тип данных поля определяется положением этого поля в сообщении. Длина каждого поля должна быть кратна 4 байт. Многие параметры представляются целыми числами без знака длиной в 4 байта. Например, *процедура с номером 5* будет представлена как:

```
00 00 00 05
```

Строки ASCII кодируются как 4-октетное целое число, содержащее длину строки со следующими далее символами ASCII, дополненными до полей, кратных 4 байт. Например, строка *README* будет выглядеть как:

```
(длина строки = 6) R  E  A  D  M  E (заполнитель)
  00 00 00 06      52 45 41 44 4D 45 00 00
```

Альтернативный метод определения и кодирования специфицирует стандарт описания данных в *первой абстрактной синтаксической нотации* OSI (OSI Abstract Syntax Notation 1 — ASN.1) и стандарт *базовых правил кодирования* (Basic Encoding Rules — BER,). ASN.1 и BER используются некоторыми приложениями TCP/IP. Наиболее значимым из них является Simple Network Management Protocol (SNMP).

Стандарт кодирования BER предполагает размещение перед каждой порцией данных специального поля, идентифицирующего эти данные и определяющего их длину (ASN.1 и BER обсуждаются в главе 20). Преимущество XDR состоит в том, что данные кодируются существенно меньшим количеством байт, а недостаток — в том, что каждое поле должно быть в предопределенном месте сообщения.

15.12 Программные интерфейсы RPC и XDR

Приложения клиент/сервер для RPC строятся на основе библиотеки подпрограмм для создания, отправки и получения сообщений RPC. Другие программы библиотеки служат для преобразования между локальным представлением данных для параметров сообщения и форматом XDR. Типичная подпрограмма RPC:

```
int callrpc (хост, номер_программы, номер_версии, номер_процедуры,
входная_программа, входные_параметры, выходная_программа, выходные_параметры)
```

Параметр "*хост*" идентифицирует компьютер сервера, *номер_программы* определяет программу, а *номер_процедуры* — выполняемую процедуру. Передаваемые в сообщении запроса входные параметры описываются структурой *входные_параметры*, а *входная_программа* преобразует эти параметры в формат XDR. Когда прибывает ответ, программа *выходная_программа* преобразует параметры ответа XDR в локальный формат и сохраняет их в структуре *выходные_параметры*.

Компании NetWise и Sun разработали комплект программных инструментов, который упрощает создание приложений клиент/сервер для RPC и скрывает от разработчика запросы RPC нижнего уровня.

15.13 Введение в NFS

Сетевая файловая система (Network File System — NFS) — это архитектура файлового сервера для различного оборудования, операционных систем, транспортных протоколов или сетевых топологий. Однако первоначально она была разработана для Unix.

Перед использованием NFS хост клиента проводит *монтирование* (mounting) удаленного поддерева каталогов в свою собственную файловую систему, посылая запрос RPC к программе *mount* сервера.

Конечный пользователь или приложение могут даже не догадываться о существовании NFS. Когда формируется запрос на выполнение операции с файлами (например, открытие, чтение, запись, копирование, переименование, удаление и т.д.) и нужный файл находится на удаленном сервере, то операционная система переадресует запрос в NFS. Запрос пересылается в сообщении RPC. Входные и выходные параметры кодируются по стандарту XDR.

На рис. 15.8 показаны компоненты для поддержки запроса NFS. Обычно NFS реализуется поверх транспортного протокола UDP, однако современные продукты работают через соединения TCP. UDP прекрасно подходит в том случае, когда клиент и сервер находятся в одной локальной сети. TCP более применим для коммуникаций через региональные сети, в которых требуется вычисление тайм-аута повторной пересылки и согласование нагрузки.

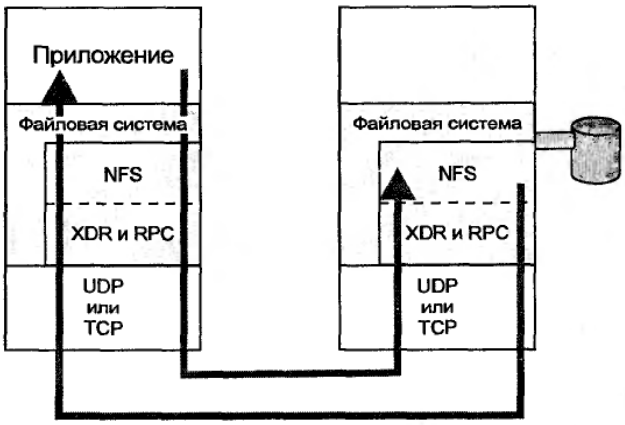

Рис. 15.8. Компоненты поддержки NFS

Обычно NFS реализуется через несколько одновременных процессов на сервере, значит многие клиенты могут работать параллельно.

15.14 Модель файлов NFS

NFS прекрасно согласуется с клиентами и серверами, имеющими файловую структуру, подобную Unix. Операционная система Unix хранит файлы в иерархическом дереве каталогов (хотя существуют успешные реализации NFS с плоской структурой каталогов, например на серверах IBM VM).

Файлы и каталоги Unix идентифицируются путем, состоящим из имен, проходимых при перемещении по дереву каталогов от корня к данному файлу или каталогу. Каждое имя отделяется символом косой черты, например */etc/hosts* или */usr/john/abc*.

Синтаксис записи путей в других операционных системах может быть не таким, как в Unix. Например, в DOS имя файла записывается как *E:\WP\LETTER.DOC*. В NFS предполагается, что каждый файл полностью определяется своим путем.

15.14.1 Источник формирования модели NFS

Отдельные части системы каталогов Unix могут размещаться на различных жестких дисках. Например, файлы и каталоги */etc* могут находиться на одном физическом диске, а каталог */var* и его подкаталоги — на другом. Команда *mount* операционной системы Unix служит для соединения отдельных частей каталогов в единое дерево. Типичная команда mount выглядит так:

```
mount /dev/xy0b /var
```

В данном случае файлы физического устройства *xy0b* будут идентифицироваться как файлы каталога */var*.

При разработке NFS возможности команды *mount* были расширены на удаленные поддеревья, которые стали также подключаться к дереву каталогов компьютера. Например, если сетевой администратор хочет использовать место на компьютере *bighost* для резервного копирования файлов хоста *tiger*, то он создает на компьютере *bighost* каталог */users*. С системы *tiger* администратор вводит команду:

```
mount -t nfs bighost:/users /usr
```

Каталог сервера */users* и все его подкаталоги логически подключаются к дереву каталогов системы *tiger* в точке */usr*. Для конечных пользователей *tiger* дерево каталогов расширяется (см. рис. 15.9). Однако файлы в */usr/john/abc* реально будут находиться в каталоге */users/john/abc* сервера *bighost*.

После монтирования, когда локальный пользователь будет запрашивать файлы из каталога */usr*, операционная система будет знать, что эти файлы реально размещаются в каталоге */users* компьютера *bighost*.

Дерево каталогов Unix имеет одну корневую точку. DOS может иметь множество деревьев (не назвать ли их лесом?), начинающихся от устройств A:, B:, C: и т.д. При монтировании удаленного каталога в DOS он становиться новым устройством локальной системы (например, E:).

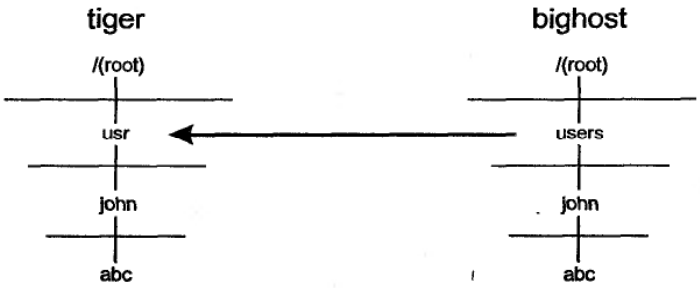

Рис. 15.9. Монтирование удаленного каталога

Иерархическую структуру каталогов имеют и другие операционные системы. Иногда приходится учитывать ограничения на глубину вложенности каталогов и длину имен файлов и каталогов.

15.15 Протокол монтирования

Команда *mount* служит для подключения (монтирования) удаленного каталога к локальной файловой системе. Эта команда реализуется программой RPC с номером 100005, а используемый порт предоставляется программой *portmapper*. *Монтирование* работает поверх UDP и TCP.

Перед тем как монтировать каталоги сервера, его нужно сконфигурировать на экспорт этих каталогов (командой *export*). Обычно это делает администратор, изменяя список экспортируемых файлов системы, включающий имена экспортируемых каталогов, имена систем (для которых разрешен доступ) и права доступа. Например, в Unix конфигурационный файл */etc/exports* может содержать:

```
/Man      -ro
/Bin      -ro,access = tiger:lion
/Users    -rw,access = tiger
```

В данном случае доступ к первому каталогу разрешен любым системам, но только для чтения (-ro). Ко второму каталогу могут обращаться для чтения хосты *tiger* и *lion*, а к каталогу */users* доступ для чтения и записи (-rw) разрешен системе *tiger*.

Сервер может экспортировать только собственные каталоги и не может разрешить экспорт каталогов, смонтированных на другом сервере NFS. Клиент может монтировать каталоги любого количества серверов. Разумеется, монтировать можно только каталоги, которые сервер разрешил использовать данному клиенту.

Клиент должен явным образом указать все монтируемые им каталоги удаленных серверов. Обычно это делается при выполнении последовательности команд *монтирования* во время запуска операционной системы клиента. Иногда команда *mount* читает специальный конфигурационный файл.

Команда *монтирования* может иметь необязательные параметры, среди которых наиболее важны те, что определяют:

- Должен ли каталог иметь полномочия для чтения и записи или только для чтения.
- Нужно ли периодически возобновлять попытки монтирования в фоновом режиме при неудаче монтирования во время запуска системы.
- Требуется ли прерывать выполнение запроса RPC к NFS при длительном ожидании ответа.
- Реализована ли в NFS одна из версий системы безопасности RPC.

Команда *mount* приводит к отправке на сервер запроса RPC с именем *Add Mount Entry* (добавить точку монтирования). *В ответе на это сообщение протокол монтирования возвращает описатель файла, который клиент будет использовать для идентификации каталога в последующих запросах.* Вспомним, что описателем является строка, содержащая идентификатор сервера и соответствующего каталога или файла. Например, когда происходит монтирование */users* как локального каталога */usr*, в ответе на запрос монтирования будет возвращен описатель файла для каталога */users*.

15.15.1 Процедуры монтирования

Процедуры, поддерживающие программу *mount* на сервере, показаны в таблице 15.4.

Таблица 15.4

Процедуры монтирования

Процедура	Описание
0	*Null* (пустая): Ответ указывает на активность программы.
1	*Add Mount Entry* (добавить точку монтирования): В список команды mount добавляется элемент для монтируемого удаленного каталога.
2	*Return Mount Entries* (возвратить точку монтирования): Возвращение клиенту текущего пути монтируемого каталога.
3	*Remove Mount Entry* (удалить точку монтирования): Удалить сведения о заданном каталоге.
4	*Remove All Mount Entries* (удалить все точки монтирования): Удалить всю информацию о монтировании удаленных каталогов.
5	*Return Export List* (возвратить список экспортируемых каталогов): Получить список каталогов и хостов, к которым разрешен доступ.

15.16 Особенности NFS

В NFS требуется как можно большая *независимость* сервера. Сервер NFS должен хранить как можно меньше сведений о клиенте, чтобы при крахе клиента или сервера восстановление было простым и безболезненным.

Клиент знает, что сервер NFS берет на себя всю работу по обслуживанию запроса и выводу ответа. Однако часто NFS работает поверх протокола UDP, не обеспечивающего целостности информации. Что делать, если на запрос не приходит ответ? NFS просто повторяет запрос после интервала тайм-аута.

Иногда случается так, что исходный запрос попадает на сервер, но ответ на этот запрос теряется. Для таких случаев сервер NFS не придерживается полной независимости от клиента и кеширует самые последние ответы для вывода их на дублированные запросы.

На какие из запросов следует кешировать ответы? Некоторые операции (например, *чтение* или *просмотр*) являются *равномощными* (idempotent), т.е. при многократном выполнении они формируют тот же самый результат. Другие операции (например, *удаление файла* или *создание каталога*) — не являются равномощными. При потере исходного результата выполнение повторного запроса на операцию приведет только к бессмысленному сообщению об ошибке. Понятно, что кешировать следует операции, не являющиеся равномощными: это позволит NFS формировать корректный ответ на повторные запросы.

15.17 Протокол NFS

Последней реализацией NFS является версия 3, хотя продолжают успешно применяться реализации версии 2. Программа NFS сервера имеет номер 100003 и, по соглашению, NFS захватывает при инициализации порт 2049.

15.17.1 Описатели файлов

Когда клиент монтирует каталог, протокол возвращает ему описатель файла (file handle), который должен идентифицировать данный каталог в последующих запросах клиента. Монтируемый каталог может содержать подкаталоги, имеющие, в свою очередь, собственные подкаталоги, и т.д. Возможно, путь к файлу будет содержать несколько уровней вложенности. Например, перед тем как клиент сможет изменить файл:

```
/usr/john/book/chapter3
```

необходимо получить описатель данного файла с сервера. Для этого NFS выполняет последовательный поиск (одно перемещение по дереву за каждый запрос). Для нашего файла клиент должен:

- Послать на сервер запрос на просмотр описателей файлов каталога */users* и указать имя *john*. В ответе будет возвращен описатель каталога */users/john*.
- Послать на сервер запрос на просмотр описателей файлов каталога */users/john* и указать имя *book*. Сервер возвратит описатель для */users/john/book*.
- Послать на сервер запрос на просмотр описателей файлов каталога */users/john/book* и указать имя *chapter3*. В ответе будет содержаться описатель нужного файла.

Таким образом, для получения описателя файла клиент NFS должен отправить несколько запросов.

15.17.2 Процедуры NFS

Существуют процедуры NFS, обеспечивающие клиенту доступ, чтение или запись удаленного файла. Клиент может узнать структуру и реальную емкость удаленной файловой системы либо запросить атрибуты удаленного файла. Допустимо удалять и переименовывать файлы. Некоторые процедуры специфичны для файловой системы Unix (например, связывание с именем псевдонима файла). Процедуры NFS версий 2 и 3 кратко представлены в таблице 15.5.

Таблица 15.5

Процедуры NFS версий 2 и 3

Процедура	Версия 2	Версия 3
0	Пустая процедура для тестирования	Пустая процедура для тестирования.
1	Получить атрибуты файла	Получить атрибуты файла.
2	Установить атрибуты файла	Установить атрибуты файла.
3	Устаревшая процедура	Просмотр имени файла. По описателю файла для каталога и имени подкаталога или файла возвратить описатель файла для подкаталога или файла.
4	Просмотр имени файла	Проверка полномочий доступа.
5	Чтение информации о связанной с файлом символьной ссылке	Чтение информации о связанной с файлом символьной ссылке.
6	Чтение данных из файла	Чтение данных из файла
7	Не используется	Записать данные в файл. Запрос может указывать, будет ли кешироваться операция записи и будет ли результат операции фиксироваться в устойчивом состоянии до отправки ответа.
8	Записать данные в файл	Создать файл.
9	Удалить файл	Создать каталог.
10	Создать файл.	Создать символьную ссылку (symbolic link).
11	Переименовать файл	Создать узел (например, специальное устройство).
12	Создать ссылку на файл	Удалить (стереть) файл.
13	Создать символьную ссылку	Удалить каталог.
14	Создать каталог	Переименовать файл или каталог.
15	Удалить каталог	Создать ссылку на объект.
16	Прочитать имя файла или файлов из каталога	Прочитать имя файла или файлов из каталога.
17	Получают информацию о файловой системе (например, о размере блока и количестве свободных блоков)	Прочитать имена файлов, поля, атрибуты и описатели из каталога.

Таблица 15.5 (*продолжение*)

Процедуры NFS версий 2 и 3

Процедура	Версия 2	Версия 3
18		Получить динамическую информацию от файловой системы (например, об общем размере и объеме свободного пространства).
19		Получить статическую информацию от файловой системы (например, о максимальном размере для запросов чтения и записи).
20		Извлечение информации POSIX (например, об атрибутах и максимальной длине имени файла).
21		Фиксация (commit): перенос предварительно размещенных в кеше данных на устройство постоянного хранения.

15.17.3 Специальные утилиты

В идеале NFS должна быть прозрачна для пользователей. Файлы сервера должны открываться, читаться, записываться и закрываться так же, как локальные файлы, а применяться для этого должны обычные локальные команды.

Когда клиент и сервер имеют одинаковые операционные системы, проблем не возникает. Иногда для NFS требуется только несколько дополнительных команд для согласования различных типов операционных систем клиента и сервера. Рассмотрим конкретный пример.

Когда клиент DOS обращается к файловому серверу Unix, создаваемые и именуемые клиентом файлы должны соответствовать требованиям DOS и являться реальной частью клиентской файловой системы.

Когда клиенту DOS нужно прочитать текстовый файл, созданный в Unix, возникает несколько проблем. Прежде всего, имена файлов в DOS ограничены 8-ю символами, а далее следуют необязательные точка и еще 3 или меньше символов (расширение имени файла). В DOS все имена файлов принято записывать символами верхнего регистра. Например: COMMAND.COM. Имена файлов в Unix могут быть гораздо длиннее и состоять из символов верхнего и нижнего регистров. Например, в Unix вполне допустимо имя *aLongerName.More*.

Как же пользователь DOS получит доступ к такому файлу? Обычно разработчики реализуют автоматическую трансляцию имен или включают специальные утилиты, разрешающие пользователям указывать исходные имена файлов на сервере. (Более распространена эмуляция на клиентском компьютере операционной системы сервера — тогда при доступе к файлам можно не только использовать родные соглашения об именовании файлов, но и применять родные команды операционной системы для обработки этих файлов; когда же возникает необходимость в переносе файла из одной операционной системы в другую, применяются специальные программы-конвертеры.— *Прим. пер.*)

Однако существуют и другие проблемы. Строки текстовых файлов DOS завершаются символами возврата каретки (CR) и перевода строки (LF), в то время как в Unix применяется только LF. Некоторые разработчики реализуют автоматическую трансляцию на основе специальных утилит преобразования к локальному формату.

15.17.4 Блокировка файлов

К некоторым файлам могут одновременно обратиться несколько пользователей. Например, конфигурационные файлы могут читаться несколькими процессами. Для изменения совместно используемого файла пользователь должен получить специальные полномочия — эксклюзивный доступ к этому файлу с помощью блокировки доступа для других пользователей на время внесения изменений.

Блокировка файлов в NFS реализуется двумя службами: *диспетчером блокировки* (lock manager) и программой *статуса* (status). *Диспетчер блокировки* управляет клиентскими запросами на блокировку файлов. Программа *status* на сервере отслеживает текущие блокировки, выполняемые клиентскими хостами. При крахе сервера программа статуса отсылает уведомление зарегистрированным клиентским хостам, запрашивая от них снятие блокировок файлов.

15.17.5 Заметки о реализациях NFS

Программа может постоянно запрашивать от своей операционной системы чтение или запись небольшого числа байт. Постоянный доступ к жесткому диску за небольшим количеством данных крайне неэффективен. Обычно операционная система проводит упреждающее чтение целого блока данных и отвечает на запрос, используя данные из собственной памяти. Аналогичным образом производится кеширование данных при записи на жесткий диск.

Частый доступ к удаленному серверу NFS за небольшим количеством данных еще более неэффективен, чем доступ к локальному жесткому диску. Клиентские реализации NFS тоже выполняют упреждающее чтение блоков данных.

Сервер NFS может существенно повысить производительность, сохраняя в памяти информацию о каталогах и атрибутах файлов, равно как и выполняя упреждающее чтение запрашиваемой клиентом информации. В NFS версии 3 поддерживается запись в кеш с *фиксацией* (commit) содержимого кеша на устройстве постоянного хранения.

15.17.6 Мониторинг NFS

Команда *nfsstat* из Unix выводит сведения о действиях NFS. Подобные команды доступны и в других операционных системах. В представленном ниже примере локальная система работает и как сервер, и как клиент. Ее деятельность в качестве сервера почти незаметна. Однако пользователи системы формируют большое число клиентских запросов.

В отчете команды показано количество использований запросов каждого типа за период мониторинга. Видно множество операций *просмотра* (lookups), что связано с последовательным, пошаговым получением описателей файлов при движении вниз по дереву каталогов.

```
> ntsstat

Server rpc:
Calls      badcalls     nullrecv      badlen     xdrcall
25162314       0            0            0          0

Server nfs:
Calls        badcalls
25162314       491
Null       getattr      setattr      root         lookup        readlink
478 0%   9689121 38%    380591 1%    0 0%       5596396 22%   5992775 23%
read
1009813 4%

Wrcache    write      create     remove       rename     link      symlink
0 0%     1146142 4% 627381 2%  66180 0%     13089 0%  6042 0%     265 0%
Mkdir      rmdir      readdir      fsstat
1718 0%     66 0%     626437 2%     5820 0%

Client rpc:
Calls     badcalls    retrans    badxid    timeout    wait    newcred    timers
3931394    2069          0         42        2037       0         0        1697

Client nfs:
Calls       badcalls     nclget      nclsleep
3929178         32        3929357         0
Null     getattr     setattr     root       lookup        readlink
0 0%   2221718 56%    6689 0%     0 0%     1423702 36%     93498 2%
Read       wrcache     write       create     remove    rename
54110 1%     0 0%     19501 0%     7362 0%    6493 0%   158 0%
Link    symlink     mkdir     rmdir     readdir      fsstat
5 0%     0 0%        28 0%     12 0%     95804 2%      98 0%
```

15.18 Дополнительная литература

На момент выхода книги *portmapper* и *RPCBIND* были определены в RFC 1833, протокол Remote Procedure Call версии 2 в RFC 1831, а XDR в RFC 1832.

Версия 2 системы NFS рассматривается в RFC 1094, а версия 3 описана в RFC 1813. Очень полная спецификация версии 2 системы NFS приведена в *X/Open CAE Specification: Protocols for X/Open Internetworking: XNFS*, опубликованной X/Open Company, Ltd.

Глава

16

Электронная почта

16.1 Введение

Среди всех приложений TCP/IP наибольшей популярностью пользуется электронная почта (далее мы будем называть ее для краткости просто почтой.— *Прим. пер.*). Когда в организации появляется хороший доступ к почтовой службе, всегда резко увеличивается число пользователей сети — ведь почтовые программы могут успешно применять даже люди, ранее и не мечтавшие использовать компьютер в своей работе.

Электронная почта обеспечивает простой и удобный способ обмена сообщениями между людьми. Показанный ниже диалог представляет взаимодействие с одной из базовых почтовых программ Unix. Программа выводит подсказку Subject (тема сообщения), и далее можно вводить сам текст сообщения. Ввод будет завершен после набора символа точки как единственного символа строки.

```
> mail fred
Subject: New Materials
The manuals have arrived.
Let's discuss them next week.
.
```

Существуют более элегантные почтовые программы, имеющие полноэкранный пользовательский интерфейс и операции с мышью. Например, на рис. 16.1 показан интерфейс почтовой программы *Chameleon* для Windows, а на рис. 16.2 — *Eudora* для Macintosh.

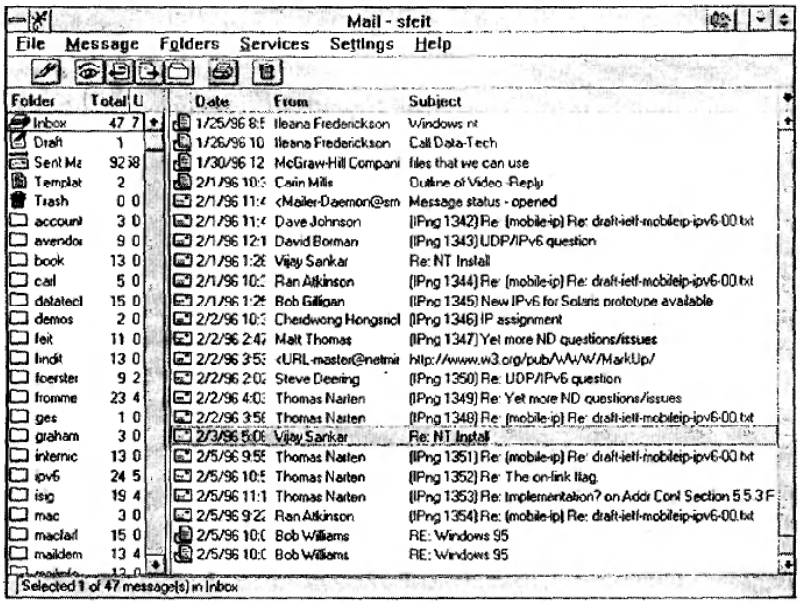

Рис. 16.1. Пользовательский интерфейс для Windows

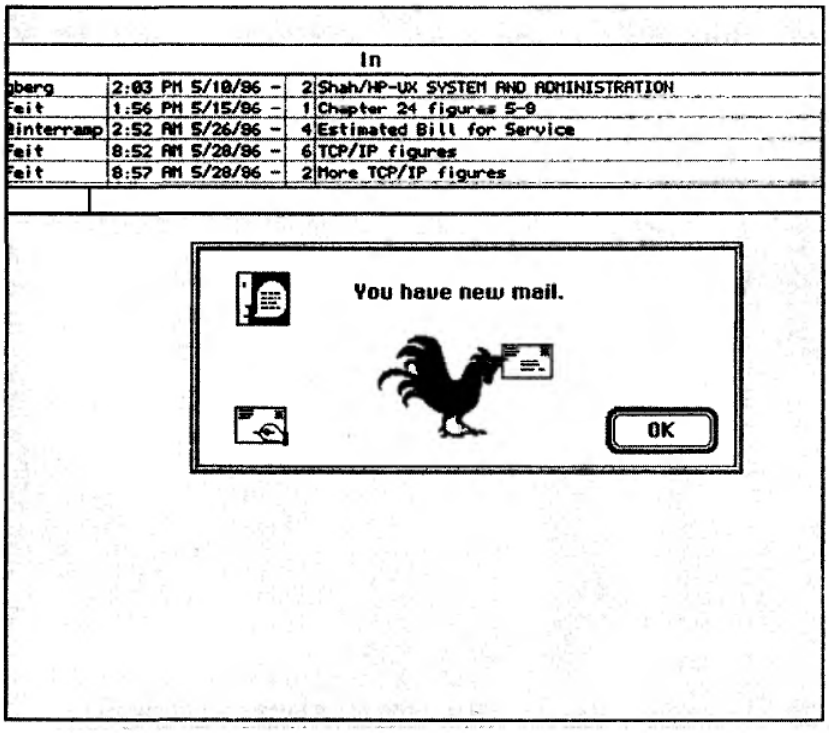

Рис. 16.2. Пользовательский интерфейс программы Eudora для Macintosh

Формальным названием почтовой программы для конечного пользователя является "*пользовательский агент*" (User Agent — UA). Этот агент должен обеспечивать несколько возможностей:

- Вывод информации о поступивших почтовых сообщениях, сохраненных в почтовом ящике (mailbox)
- Сохранение входящих и исходящих сообщений в папке или локальном файле
- Обеспечение хороших средств редактирования для ввода текста сообщений

Стиль пользовательского агента не стандартизован и полностью определяется вкусом конкретного человека. Для конечного пользователя любой агент обеспечивает одинаковые результаты — пересылку и прием почтовых сообщений.

Вернемся к рассмотренному выше примеру. Диалог выглядит очень простым, однако остается большой объем работы. Введенное имя "fred" является *кратким именем* (nickname) или *псевдонимом* (alias), который определен в адресной книге пользователя. Когда пользовательский агент просматривает адресную книгу, он по указанному псевдониму извлекает из нее реальный идентификатор получателя сообщения (например, *fred@microsoft.com*).

Такой идентификатор имеет общий для почты Интернета формат. Однако существует лицензионное программное обеспечение для почты и ее служб, предоставляющее различные варианты для формата адреса получателя. Согласование форматов происходит на *почтовых шлюзах* (mail gateways).

Как же производится доставка почты? Ранее пересылка почтовых сообщений производилась по прямым соединениям TCP между отправителем и получателем почты. Сегодня почта, как правило, пересылается через промежуточные хосты (ниже мы рассмотрим, как это происходит).

16.2 Почтовые протоколы Интернета

Почта широко распространена, поэтому было разработано несколько протоколов Интернета, обеспечивающих различные требования пользователей. На рис. 16.3 показаны почтовые протоколы Интернета.

Простой протокол почтового обмена (Simple Mail Transfer Protocol — SMTP) является классическим стандартом Интернета для пересылки почты между компьютерами. SMTP был разработан для пересылки простейших текстовых сообщений и реализован поверх сеанса *telnet* в режиме NVT.

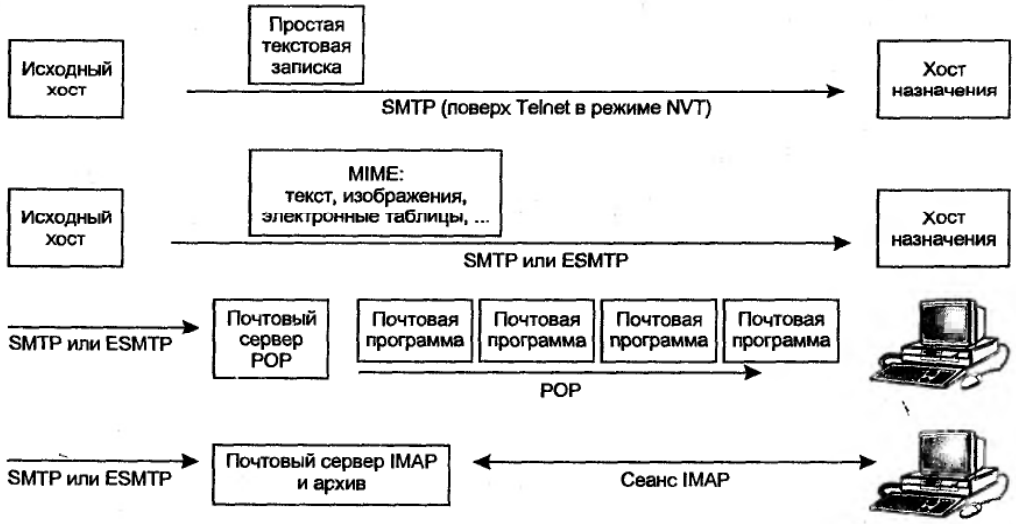

Рис. 16.3. Почтовые протоколы Интернета

По прибытии почтового сообщения пользовательский агент должен распознать отдельные части сообщения: идентификатор отправителя, размер, тему и информационную часть. Для простых текстовых сообщений в Интернете уже давно существует *стандарт формата текстовых сообщений Интернета от ARPA* (Standard for the Format of ARPA Internet Text Messages).

Серия более новых стандартов определяет *расширения для SMTP* (Extension to SMTP — ESMTP), обеспечивающих пересылку информации других типов. Сегодня сообщения из различных элементов описываются стандартом *многоцелевых почтовых расширений Интернета* (Multipurpose Internet Mail Extension — MIME). Допустимо пересылать самые различные виды данных: документы текстовых процессоров, файлы Binhex из Macintosh, графические изображения, видеоклипы, кодированные звуковые файлы, электронные таблицы, исполняемые коды программ и т.д. При необходимости вводятся новые типы MIME, регистрируемые комитетом IANA.

Еще один набор стандартов определяет способы работы с почтой. *Протокол почтового офиса* (Post Office Protocol — POP) обеспечивает загрузку сообщений с почтового сервера на настольную клиентскую систему. Альтернативный вариант, *протокол доступа к сообщениям Интернета* (Internet Message Access Protocol — IMAP) разрешает пользователям копировать, читать или удалять хранимые на сервере сообщения, однако сервер в этом случае является единственным репозиторием для сообщений. Этот вариант хорош для тех, кто желает воспользоваться преимуществами служб администрирования (например, ежедневным автоматическим резервным копированием), сохранить пространство на локальном диске или получить доступ к своим сообщениям при перемещении по сети. Почта доставляется на сервер через SMTP или ESMTP.

В некоторых организациях для доставки почты используется протокол OSI X.400, который мы рассмотрим ниже.

16.3 Модель пересылки почтового сообщения

На рис. 16.4 показаны элементы почтовой системы. Сообщение подготавливается с помощью приложения *пользовательского агента* (User Agent — UA). Обычно UA передаст сообщение другому приложению, *агенту пересылки сообщений* (Message Transfer Agent — MTA), которое устанавливает соединение с удаленным хостом и пересылает почтовое сообщение. Термины *UA* и *MTA* заимствованы из стандарта X.400, однако применимы и для стандарта SMTP.

Почту можно отправить непосредственно от источника к пункту назначения или через промежуточные MTA. В последнем случае все сообщение пересылается на промежуточный хост, где оно сохраняется до последующей отправки далее в удобное для этого время. Системы такого рода называются "*сохранить-и-переслать*" (store-and-forward).

На приемном хосте сообщение помещается в очередь входящих сообщений, а затем перемещается в область *почтового ящика* пользователя. Когда получатель запустит программу UA, обычно выводится итоговый список всех поступивших сообщений, ожидающих обработки пользователем в его почтовом ящике.

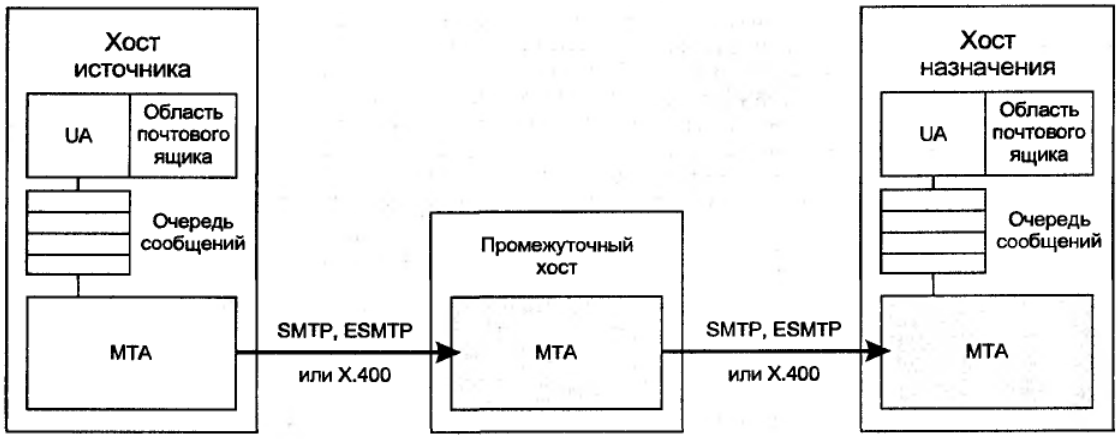

Рис. 16.4. Компоненты системы электронной почты

16.4 Пересылка почтового сообщения

Почему MTA пересылает сообщение через промежуточные хосты, а не соединяется непосредственно с системой назначения? Когда хост использует прямое соединение, должна быть уверенность в том, что сообщение достигнет конечной точки. Пересылка через промежуточные MTA требует дополнительных ресурсов и создания нескольких отдельных соединений. При пересылке нужно точно указать путь следования сообщения, иначе оно может отправиться по неэффективному маршруту.

16.4.1 Сценарий доставки почтового сообщения

Для понимания работы механизма "сохранить-и-переслать" проанализируем сценарий, приведенный на рис. 16.5. Фред работает в компании ABC Industries. Он оправляет почтовое сообщение Мери из компании JCN Computer. Компьютер Фреда является рабочей станцией локальной сети, включенной большую часть суток. Рабочая станция посылает и получает почтовые сообщения через сервер доставки своей локальной сети.

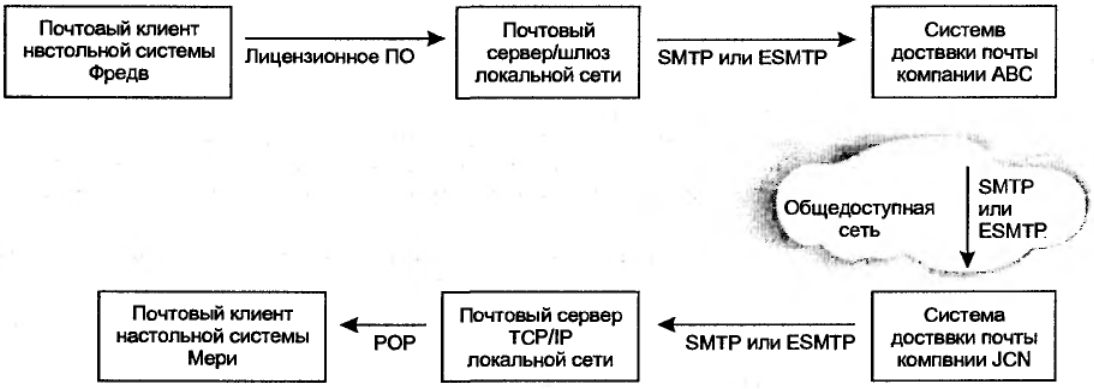

Рис. 16.5. Пересылка сообщения электронной почты

Обе компании реализуют средства защиты от внешнего доступа. Обмен почтой с внешним миром может проводиться только через специально выделенный сетевой *хост для доставки почты* (Mail Exchanger). Каждая компания соединена с внешним миром через маршрутизатор, блокирующий весь входной трафик, за исключением соединений с портом почты (25) на хосте почтовой доставки.

В локальной сети Фреда для работы с почтой используется лицензированный программный продукт, а в сети Мери — протокол TCP/IP.

Как видно на рис. 16.5, почтовое сообщение с настольного компьютера Фреда на сервер локальной сети пересылается по лицензированному почтовому протоколу. Сервер сети является программным шлюзом для трансляции между форматом лицензированного почтового программного обеспечения и форматом сообщений Интернета. После шлюза сообщение поступает на хост доставки почты компании ABC. Далее оно пересылается по внешней сети (в нашем случае — Интернет) на хост доставки компании JCN. Затем почтовое сообщение попадает на почтовый сервер локальной сети Мери, где и хранится, пока Мери не соединится с сервером и не извлечет сообщение через протокол POP.

Предложенный сценарий выявляет несколько преимуществ данного способа:

- Персональные компьютеры и рабочие станции передают серверу локальной сети права на отправку исходящих сообщений и хранение поступивших сообщений электронной почты.
- Сотрудники компании могут использовать электронную почту, но при этом сохраняются все требования к защите сети, поскольку почта пересылается по туннелю через систему почтового обмена.
- Снижаются затраты на пересылку, поскольку сообщения могут пакетироваться для одновременной доставки в любое время суток.
- При доставке может происходить преобразование формата.

В следующем разделе мы подробнее рассмотрим механизмы семейства протоколов TCP/IP, поддерживающие расширенные возможности доставки электронной почты.

16.5 Идентификация получателя и обмен сообщениями

В Интернете получатель почтового сообщения идентифицируется по имени с использованием следующего общего формата:

локальная_часть@имя_домена

Этот формат очень гибок. Многие годы в Интернете предпочитали формат с использованием имен:

идентификатор_пользователя@имя_хоста

Например:

smithm@sales.chicago.jcn.com.

В настоящее время применяется более удобный формат:

имя-фамилия@имя_почтового_домена

Например:

Mary-Smith@jcn.com

В этом случае *Mary-Smith* — это не идентификатор пользователя, а *jcn.com* — не имя хоста. Применяются локальные имена, назначенные в системе почтового обмена. Как же тогда доставлять почту адресату? Механизм почтовой доставки зависит от системы имен доменов. Его работа будет сводиться к следующему:

- Один или несколько компьютеров назначаются в качестве систем почтовой доставки для данной организации.
- Для системы почтовой доставки выбирается логическое имя, обычно имя домена организации, которое и включается в базу данных DNS.
- Программа агента пересылки сообщения (MTA) будет просматривать в адресе сообщения часть, связанную с именем почтового домена, и на ее основе извлекать из DNS реальные имена и адреса системы почтового обмена получателя. Затем туда будет направлено сообщение.

Ниже представлен пример выполняемых действий. Запускается программа *nslookup* и запрашивается идентификатор системы почтовой доставки (Mail Exchanger) компании Cisco. Как можно заметить, реально выведены сведения о двух системах — *hubbub.cisco.com* и *beasley.cicso.com*. Для большей доступности своих почтовых служб многие компании запускают несколько систем почтового обмена.

Отметим выведенные значения *предпочтения* (preference), равные 5 и 10. Более предпочтительным для отправки почты будет сервер с *меньшим номером* (hubbub). Именно с ним нужно попытаться соединиться. Если же это будет невозможно, попробуем соединиться с *beasley*. Реально сами значения предпочтения не имеют никакого смысла, важно только соотношение между ними.

```
> nslookup

Default Server: DEPT-GW.cs.YALE.EDU
Address: 128.36.0.36

> set type = mx

> cisco.com

Cisco.com      preference = 5,   mail  exchanger = hubbub.cisco.com
Cisco.com      preference = 10,  mail  exchanger = beasley.cisco.com
Hubbub.cisco.com       inet address = 198.92.30.32
Beasley.cisco.com      inet address = 171.69.2.135
Dennis.cisco.com       inet address = 171.69.2.132
Ns1.barrnet.net        inet address = 131.119.245.5
Noc.near.net           inet address = 198.112.8.2
Noc.near.net           inet address = 192.52.71.21
```

Следующий запрос показывает, как некоторые организации реализуют дополнительный слой безопасности. Отметим наличие сразу трех систем почтового обмена, две из которых фактически принадлежат провайдеру UUNET:

```
> clarinet.com
Server: DEPT-GW.cs.YALE.EDU
Address: 128.36.0.36

Clarinet.com       preference = 10,   mail  exchanger = looking.clarinet.com
Clarinet.com       preference = 100,  mail  exchanger = relay1.uu.net
Clarinet.com       preference = 100,  mail  exchanger = relay2.uu.net
Looking.clarinet.com   inet address = 192.54.253.1
Relay1.uu.net          inet address = 192.48.96.5
Relay2.uu.net          inet address = 192.48.96.7
>
```

Компания Clarinet могла бы организовать свою сеть так, чтобы поступающая почта сначала направлялась на одну из систем UUNET, а затем передавалась на *looking.clarinet.com*.

На рис. 16.6 показано, как это можно сделать. Фильтрующий маршрутизатор нужно установить на блокирование всех соединений, за исключением связи с системой почтовой доставки провайдера UUNET. Внешняя система попробует соединиться с наиболее предпочтительным сайтом *looking.clarinet.com*. Однако маршрутизатор предотвратит такое соединение. Поэтому в следующей попытке почта будет направлена на один из менее предпочтительных сайтов *relay1* или *relay2*. Теперь система UUNET сможет переслать почту системе почтового обмена компании Clarinet.

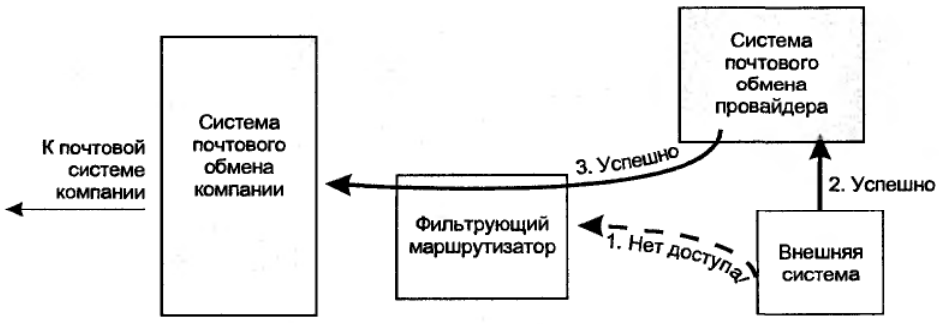

Рис. 16.6. Пересылка почтового сообщения по пути следования

Когда почта попадет на систему почтового обмена компании, будет проанализирована локальная часть логического адреса. По указанному псевдониму из специального файла будут извлечены реальные идентификатор пользователя и имя хоста, либо иной идентификатор для доставки почты в сети назначения. Таким образом, система почтовой доставки может действовать как шлюз для несовместимых с Интернетом почтовых служб.

Возникает еще одна проблема — маршрутизация почты через систему почтового обмена. Предположим, что пользователи хоста *sales.clarinet.com* имеют почтовые идентификаторы в виде *имя_пользователя@sales.clannet.com*. Что нужно сделать с адресами, подобными *jonesj@sales.clarinet.com*? Проблему решит несколько дополнительных элементов в базе данных DNS компании Clarinet:

```
*.clarinet.com. IN MX  10 looking.clarinet.com.
*.clarinet.com. IN MX 100 relay1.uu.net
*.clarinet.com. IN MX 100 relay2.uu.net
```

Подстановочный символ звездочки (*) в этих элементах позволит направить на систему почтового обмена компании сообщения, адресованные в старом стиле (*идентификатор_пользователя@имя_хоста*).

Организации заменяют свои старые идентификаторы *имя_пользователя@имя_хоста* логическими именами, которые не показывают идентификаторы пользователей посторонним. В дополнение к улучшению безопасности сети логические имена разрешают пользователям получать новый идентификатор или перемещаться между различными компьютерами без изменения почтовых идентификаторов.

16.6 Протокол SMTP

Простой протокол пересылки почты (Simple Mail Transfer Protocol — SMTP) определяет способ непосредственного перемещения почтового сообщения между хостами. В протоколе SMTP для системы описываются две роли: отправителя и получателя. Отправитель действует как клиент и устанавливает соединение TCP с получателем, который работает как сервер. Для получателя используется общеизвестный порт 25. Даже если отправителем является программа почтовой службы (Message Transfer Agent — MTA), она функционирует как клиент и использует временный порт из пула доступных портов.

В течение сеанса SMTP отправитель и получатель обмениваются последовательностью команд и ответов. Сначала получатель объявляет имя своего хоста. Затем отправитель:

- Объявляет имя своего хоста
- Идентифицирует источник сообщения
- Определяет одного или нескольких получателей
- Пересылает данные почтового сообщения
- Передает строку, содержащую ". <CR> <LF>", что указывает на завершение пересылки сообщения.

Отметим, что сообщение может быть доставлено нескольким получателям хоста в одной транзакции, поскольку в нем допустимо указывать нескольких получателей. В конце транзакции отправитель может:

- Начать следующую транзакцию
- Завершить работу и закрыть соединение

В стандарте определена команда *TURN* (возврат), позволяющая отправителю и получателю поменяться ролями. Однако эта команда редко (если вообще когда-либо) реализуется на практике.

16.6.1 Диалог при обмене почтой

В приведенном ниже диалоге отправитель пересылает сообщение получателю. Отправляющий сообщение хост работает и как шлюз системы почтового обмена для компьютеров подразделения компании. В доставляемом почтовом сообщении присутствуют следующие элементы:

```
Received: from PASCAL.MATH.YALE.EDU (MATH-GW.CS.YALE.EDU) by tigger.jvnc.net
with SMTP id AA08294
  (5.65c/IDA-1.4.4 for feit); Sun, 27 Aug 1995 08:02:55 -0400
Received: by PASCAL.MATH.YALE.EDU; Sun, 27 Aug 1995 08:01:44 -0400
Date: Sun, 27 Aug 1995 08:01:44 -0400
```

```
From: Sidnie Feit <feit-sidnie@MATH.YALE.EDU>
Message-Id: <199508271201.AA02330@PASCAL.MATH.YALE.EDU>
To: feit@tigger.jvnc.net
Subject: It's OK to talk to yourself!
Date: 08/26/95 1:29:59 PM

Hi there.
See you soon.
```

Элемент *Received* (получено) в верхней части сообщения был добавлен принимающим MTA в *tigger*. Остальная часть сообщения была передана на *tigger* от системы *pascal*.

Для пересылки сообщения отправитель открывает соединение с портом 25 получателя. Тогда получатель начинает диалог и объявляет имя своего домена.

Модель команда/ответ, которую мы видели в протоколе File Transfer Protocol (FTP), применяется и в данном случае; при этом выполняется сходное декодирование сообщения ответа. Следовательно, все сообщения от удаленного сервера электронной почты начинаются с номера ответа. Отметим, что почтовые идентификаторы выведены в угловых скобках (например, *<sfeit@pascal.math.yale.edu>*). Имена хостов не чувствительны к регистру и могут выводиться как в верхнем, так и в нижнем регистре. Однако в именах пользователей различаются регистры символов, хотя это и зависит от принятых соглашений для конкретной почтовой системы.

`220 tigger.jvnc.net 5.65c/IDA-1.4.4`	Идентификатор получателя и время его объявления.
`Sendmail is ready at Sun. 27 Aug 1995 08:02:55 -0400`	
`HELO MATH-GW.CS.YALE.EDU`	Идентификатор отправителя.
`250 Hello MATH-GW.CS.YALE.EDU, pleased to meet you`	
`MAIL FROM: <sfeit@pascal.math.yale.edu>`	Источник полученного почтового сообщения.
`250 <sfeit@pascal.math.yale.edu>.. Sender ok`	
`RCPT TO; <feit@tigger.jvnc.net>`	Получатель идентифицирован. Может присутствовать несколько операторов RCPT TO.
`250 <feit@tigger.jvnc.net>.. Receiver ok`	
`DATA`	Начало сообщения.
`354 Enter mail, end with "." on a line by itself`	
`Received: by PASCAL.MATH.YALE.EDU;`	Первым появляется заголовок.
`Sun, 27 Aug 1995 08:01:44 -0400`	
`Date: Sun, 27 Aug 1995 08:01:44 -0400`	
`From: Sidnie Feit <feit-sidnie@math.yale.edu>`	
`Message-Id: <199508271201.AA02330@PASCAL.MATH.YALE.EDU>`	
`To: feit@tigger.jvnc.net`	
`Subject: It's OK to talk to yourself!`	
`Date: 08/26/95 1:29:59 PM`	
	За заголовком следует пустая строка.
`Hi there.`	Это тело сообщения.
`See you soon.`	
`.`	Сообщение заканчивается .<CR><LF>
`250 Ok`	
`Quit`	До выхода из программы можно отправить другие сообщения.
`221 tigger.jvnc.net closing connection`	
`Connection closed by foreign host.`	

Обратите внимание, что конец сообщения отмечается строкой, содержащей только символ точки.

Предположим, что пользователю нужно послать такую строку внутри сообщения. Дополнительный символ точки будет вставлен отправителем SMTP и удален получателем SMTP.

16.7 Временная метка и идентификатор сообщения

При получении почты интересно узнать время ее отправления и получения. SMTP добавляет эту информацию к пересылаемому сообщению. Кроме того, этот протокол отслеживает все хосты, которые передавали почтовое сообщение, и время получения сообщения каждым из них.

Когда сообщение приходит к агенту пересылки SMTP, он вставляет в начало сообщения временную метку (timestamp). При каждой последующей пересылке вставляется дополнительная временная метка, содержащая:

- Идентификатор хоста, пославшего сообщение
- Идентификатор хоста, получившего сообщение
- Дату и время получения сообщения

Временные метки из заголовка сообщения обеспечивают неоценимую информацию для отладки, особенно когда возникают проблемы с пересылкой почты. Например, можно будет узнать, что сообщение оставалось на одном из промежуточных хостов в течение одного-двух дней.

Формат временной метки может различаться на различных системах, и разработчики могут включать в него дополнительные сведения. Новые реализации используют в метке значение местного времени, сопровождаемое смещением от *универсального времени* (Universal Time), которое ранее называлось временем по Гринвичу (Greenwich Mean Time).

Часы компьютера иногда установлены неточно, поэтому последовательности временных меток сообщения не всегда согласуются со здравым смыслом. Например, иногда кажется, что сообщение было получено раньше, чем было отправлено. Так как администраторы сети — единственные сотрудники, имеющие дело с установкой компьютерных часов, ошибки могут возникнуть из-за невнимательности.

Когда почта достигает точки назначения, пользовательский агент может самостоятельно добавить строку, указывающую на исходного отправителя.

Приведенный ниже пример поясняет причину добавления таких строк к сообщению. Верхняя строка была вставлена пользовательским агентом получателя. Она содержит сведения об источнике сообщения и о времени его поступления в почтовый ящик.

Идентификатор сообщения (Message-Id) в нижней части примера был добавлен первым почтовым агентом пересылки, который начал обрабатывать это сообщение.

Временные метки нужно анализировать снизу вверх, что позволит понять путь следования сообщения — от *dial131.mbnet.mb.ca* к *access.mbnet.mb.ca*, далее к *bulldog.cs.yale.edu* и наконец к *pascal.math.yale.edu*.

```
From vsankar@ForeTell.CA Thu Aug 17 14:36:19 1995
Received: from BULLDOG.CS.YALE.EDU by PASCAL.MATH.YALE.EDU via SMTP;
Thu, 17 Aug 1995 14:36:19 -0400
Received: from access.mbnet.mb.ca by bulldog.CS.YALE.EDU via SMTP;
Thu, 17 Aug 1995 14:31:47 -0400
Received: from ftl6 (dial131.mbnet.mb.ca) by access.mbnet.mb.ca with SMTP id
AA02060
(5.67b/IDA-1.4.4); Thu, 17 Aug 1995 14:31:33 -0500
Date: Thu, 17 Aug 1995 14:31:33 -0500
Message-Id: <199508171831.AA02060@access.mbnet.mb.ca>
```

16.8 Отброшенная почта

Иногда бывает невозможно переслать почту в точку назначения. Чаще всего это происходит из-за неправильного ввода идентификатора получателя. Почта, которая не может быть доставлена, отсылается назад отправителю и называется *отброшенной* (bounced mail).

16.9 Команды SMTP

Сценарий из раздела 16.6.1 содержал наиболее часто используемые команды SMTP. Полный набор команд SMTP представлен в таблице 16.1.

Таблица 16.1

Команды SMTP

Команда	Описание
HELO	Идентифицирует отправителя для получателя.
MAIL FROM	Начало почтовой транзакции и указание на источник сообщения.
RCPT TO	Идентифицирует отдельного получателя. Последовательность таких команд позволяет указать несколько получателей. Получатель по возможности проверяет правильность указанного имени и выводит результат проверки в ответном сообщении. Такая проверка не имеет смысла на промежуточных хостах. Если позже окажется, что некоторый получатель указан некорректно, обратно отправляется краткое сообщение об ошибке.
DATA	Отправитель готов передать строки текста. Каждая строка завершается <CR> <LF>. Максимальная длина строки, включая <CR> <LF>, составляет 1000 символов. Реализации SMTP должны обеспечивать отправку и получение сообщений длиной до 64 К/байт. Желателен максимальный размер, поскольку почта часто используется для копирования файлов.
RSET	Прерывает текущую почтовую транзакцию, удаляя всю информацию о ней у отправителя и получателя.
NOOP	Запрашивает у партнера положительный ответ.
QUIT	Запрашивает у партнера положительный ответ и закрытие соединения.
VRFY	Запрашивает у партнера подтверждение правильности указанного имени получателя.
EXPN	Запрашивает у партнера подтверждение соответствия имени получателя списку почтовой рассылки (mailing list). Если указанное имя находится в списке, нужно возвратить сведения о членстве в группе данного почтового списка.
HELP	Запрашивает у партнера информацию об используемой реализации, например о списке поддерживаемых команд.
Описанные в стандарте, но редко реализуемые или используемые команды	
TURN	Запрос смены ролей получателя и отправителя. Партнер может отказаться выполнить эту команду.
SEND	Если получатель зарегистрирован в системе назначения — направить сообщение прямо на терминал получателя.
SOML	Send or Mail — послать или отправить. Если получатель зарегистрирован в системе назначения — направить сообщение прямо на терминал получателя, иначе отправить сообщение как почту локальной системы.
SAML	Send and Mail — послать и отправить. Доставить в почтовый ящик получателя. Если пользователь зарегистрирован, то доставить и на его терминал.

Команды пересылаются как 4-символьные мнемонические названия. Многие команды сопровождаются параметрами.

Сеанс между партнерами SMTP напоминает соединение *telnet* в режиме NVT: используются те же самые правила, например пересылаются 7-битные символы ASCII в виде 8-разрядных байтов, а каждая строка оканчивается символами перевода строки и возврата каретки.

16.10 Коды ответов

Коды ответов SMTP имеют структуру, подобную кодам ответов FTP. Код состоит из трех цифр. Первая цифра указывает статус команды:

1yz	Положительный предварительный (Positive Preliminary) ответ (в настоящее время в SMTP не используется)
2yz	Положительный дополненный (Positive Completion) ответ
3yz	Положительный промежуточный (Positive Intermediate) ответ
4yz	Кратковременный отрицательный (Transient Negative) ответ ("повторить попытку")
5yz	Постоянный отрицательный (Permanent Negative) ответ

Вторая цифра классифицирует сам ответ:

x0z	В ответ на возникновение проблемы указывает на синтаксическую ошибку или неизвестную команду
x1z	Ответ на информационный запрос (например, *help*)
x2z	Ответ с информацией о соединении
x3z	В настоящее время не определен
x4z	В настоящее время не определен
x5z	В ответе указываются сведения о почтовой системе получателя

Значение третьей цифры меняется в зависимости от команды и первых двух цифр кода.

16.11 Формат сообщений Интернета

Стандарт для формата сообщений Интернета определен в RFC 822. Сообщение состоит из (в порядке списка):

- Набора полей заголовка (многие из них необязательны)
- Пустой строки
- Текста, или тела (body), сообщения

Поле заголовка имеет вид:

Имя_поля: Содержимое_поля

Имена полей и их содержимое записываются символами ASCII. Существуют разнообразные поля заголовка. К наиболее распространенным можно отнести:

```
Received (получено)
Date (дата)
From (от)
To (кому)
cc (система cc-Mail)
bcc (blind cc - неявный формат cc-Mail)
Message-Id (идентификатор сообщения)
Reply-To (кому ответить)
Sender (отправитель, если он не является автором сообщения)
In-Reply-To (в ответ на)
References (ссылка на идентификатор более раннего сообщения)
Keywords (ключевые слова для поиска)
Subject (тема)
Comments (комментарии)
Encrypted (шифровано)
```

Можно ожидать, что каждый заголовок сообщения содержит поля *Date*, *From* и *To*. *Добавленные* поля (received field) формируются на основе временных меток, собираемых при переходе через промежуточные почтовые агенты пересылки. По большей части почтовое программное обеспечение может создавать идентификатор, который вставляется в сообщение. Например:

```
Message-Id: <199508271201.AA02330@PASCAL.MATH.YALE.EDU>
```

Поле Message-Id должно быть уникально для сети. Для этого в поле наряду с уникальным буквенно-цифровым идентификатором обычно включается имя хоста отправителя. Отметим, что показанный выше идентификатор содержит дату (1995 08 27), универсальное время (12 01) и дополнительную строку, обеспечивающую уникальность идентификатора для данного хоста и времени отправки.

Поля *Resent* (пересылка) добавляются на промежуточных системах. Например: *Resent-To* (куда переслать), *Resent-From* (откуда переслать), *Resent-cc* (переслать в систему cc-Mail), *Resent-bcc* (переслать в blind cc-Mail), *Resent-Date* (когда переслать), *Resent-Sender* (от кого переслать), *Resent-Message-Id* (с каким идентификатором переслать) и *Resent-Reply-To* (переслать в ответ на что).

Очень важна пустая строка за заголовком сообщения. По ней пользовательский агент определяет, что заголовок завершился и начинается тело сообщения.

16.12 Почтовые расширения файлов и MIME

Простота SMTP и формата почты облегчает реализацию почтовых систем Интернета и приводит к широкому распространению этих средств. Однако пользователям неудобно работать с простыми и ограниченными по своим возможностям текстовыми сообщениями. Ясно, что SMTP нуждался в переработке. Но как это можно было сделать без изменения уже установленных базовых почтовых приложений?

Было выбрано очень практичное решение. Новые клиенты MIME должны создаваться с учетом возможности получать сообщения из нескольких частей, содержащих много полезных типов информации. Эти сообщения позволяют проводить обмен:

- Эффективно — через новый улучшенный агент пересылки сообщений SMTP.
- Менее эффективно — через старый стандарт SMTP. Перед пересылкой нетекстовой части сообщения старому агенту SMTP эта часть должна быть преобразована так, чтобы она выглядела как обычный текст NVT.

На рис. 16.7 показана работа такой архитектуры.

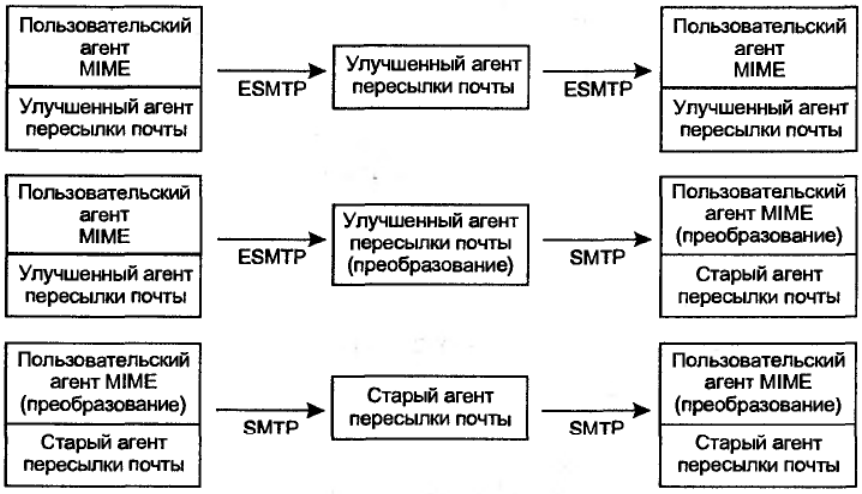

Рис. 16.7. Доставка сообщения MIME

16.12.1 Улучшенный агент пересылки почты

Улучшенный агент пересылки почты (Extended Message Transfer Agent) должен поддержать одну дополнительную команду. Вместо *HELO* он посылает сообщение-приветствие *EHLO.* Если ответ положителен, партнер также является улучшенным агентом пересылки почты (Extended MTA). Но если ответом будет сообщение об ошибке, значит MTA должен вернуться к протоколу SMTP и послать команду HELO.

Потребность поддержки MIME была основным поводом для улучшения агентов пересылки почты MTA. Кроме этого, можно добавить поддержку дополнительных служб посредством введения новых ключевых слов для EHLO. Для пересылки сообщения увеличенного размера имеется новая служба, позволяющая отправителю декларировать размер сообщения перед его отправкой. Приемник может указать, готов ли он принять сообщение такого размера. Он также может указать наибольший доступный для него размер.

Официальные расширения регистрируются в Internet Assigned Numbers Authority (IANA). Отдельные программы включают новые экспериментальные расширения, для которых используются временные названия, начинающиеся с X.

16.12.2 Диалог в улучшенной версии SMTP

Показанный ниже пример демонстрирует, как улучшенный агент пересылки почты формирует транзакцию для отправки сообщения MIME в 8-битном формате:

- Получатель объявляет о своих улучшенных возможностях, включая 8BITMIME.
- Команда MAIL FROM имеет параметр BODY = 8BITMIME.

```
EHLO MATH-GW.CS.YALE.EDU
250-Hello MATH-GW.CS.YALE.EDU, pleased to meet you
250-8BITMIME
250-HELP
250-SIZE
250-XONE
250-XVRB
250-XQUE
MAIL FROM:<feit-sidnie@math.yale.edu> BODY = 8BITMIME
250 <feit-sidniie@math.yale.edu>... Sender ok
RCPT TO:<Mary-Smith@jcn.com>
250 < Mary-Smith@jcn.com>... Recipient ok
DATA
354 Send 8BITMIME message, ending in CRLF.CRLF.
...
.
250 OK
QUIT
250 Goodbye
```

16.13 Формат сообщений MIME

Сообщение MIME содержит набор заголовков и одну или несколько *частей тела сообщения* (body part). Обычное сообщение почты Интернета начинается заголовками, подобными *From:*, *To:* и *Date:*. Сообщение MIME содержит дополнительные вводные заголовки, описывающие структуру и содержание сообщения.

Если сообщение состоит из нескольких частей, один из вводных заголовков определяет строку, используемую как *разделитель.* После такого разделителя будут стоять дополнительные заголовки, которые описывают следующую далее часть сообщения.

16.13.1 Заголовки описания типа содержания в MIME

Существует множество различных типов информации, которую можно разместить в сообщении. Общая структура сообщения и типы информации в каждой его части объявляются в заголовке *Content-Type* (тип содержания). Пример такого заголовка:

```
Content-Type: MULTIPART/MIXED; BOUNDARY ="xxxxxxxxx"
Content-Type: TEXT/PLAIN; charset=US-ASCII
Content-Type: image/gif
Content-Type: audio/basic
```

В основном заголовок Content-Type имеет форму:

Content-Type: *тип/подтип; param = значение; param = значение; ...*

Типы, подтипы и имена параметров нечувствительны к регистру символов. Они могут быть записаны в верхнем или нижнем регистре, равно как и в смешанном формате. Однако некоторые *значения* параметров зависят от регистра символов.

Хотя заголовки MIME записываются английскими фразами, параметр *charset* может объявить, что часть представлена в кодировке ISO-8859-1 или символами японского, еврейского, арабского языков или кириллицы.

16.13.2 Пример сообщения MIME

Показанное ниже сообщение MIME имеет несколько частей: одну текстовую часть и два подключенных текстовых файла. Первый заголовок Content-Type

```
Content-Type: MULTIPART/MIXED;
BOUNDARY = "plum.yale.edu:814898609:772210698:709846916:1916796928"
```

указывает, что сообщение состоит из нескольких частей. Параметр *BOUNDARY* (разделитель) маркирует начало и конец каждой части. Разделитель выбирается пользовательским агентом. В данном случае разделитель состоит из имени хоста и строки цифр, сгенерированных пользовательским агентом. Фактическая граница будет состоять из двух символов дефиса (--) и следующей далее строки-разделителя.

Заголовки MIME показаны в примере полужирным шрифтом. Справа добавлены комментарии. Отдельные строки сообщения свернуты, чтобы можно было вставить комментарий.

```
                                        Это стандартные почтовые заголовки.
Mime-Version: 1.0                       Указание на версию MIME.
Content-Type: MULTIPART/MIXED;
BOUNDARY = "plum.yale.edu:814898609:    В сообщении несколько частей.
772210698:709846916:1916796928"         Описание разделителя. Пустая строка,
                                        определяющая завершение заголовков.
-- plum.yale.edu:814898609:772210698:   Разделитель. Отметим наличие
709846916:1916796928                    начальных дефисов.
Content-Type: TEXT/PLAIN; charset=
US-ASCII                                Далее следует обычный текст.
                                        Пустая строка отмечает завершение
                                        заголовков первой части сообщения.
Подключаемая часть.                     Содержимое текстовой части.
-- plum.yale.edu: 814898609:772210698:
709846916:1916796928                    Следующий разделитель.
Content-Type: TEXT/plain; SizeOnDisk=28; Снова обычный текст. В параметрах
name="ATT.TXT"; CHARSET= US-ASCII       указана дополнительная информация.
Content-Description: ATT.TXT            Параметр задает имя файла.
                                        Конец заголовков данной части.
```

`** Первый подключенный фрагмент **`	Текстовое содержимое.
`-- plum.yale.edu: 814898609:772210698: 709846916:1916796928`	Следующий разделитель.
`Content-Type: TEXT/plain; SizeOnDisk =58368; name="NFSCAP.TXT"; CHARSET =US-ASCII`	Еще один обычный текстовый фрагмент.
`Content-Description: NFSCAP.TXT`	
	Конец заголовков данной части.
`Второй подключенный фрагмент. Далее следует текстовая часть сообщения:`	Текстовый фрагмент.
`. . .`	...
`. . .`	...
`-- plum.yale.edu:814898609:772210698: 709846916:1916796928--`	Заключительный разделитель.

16.13.3 Типы содержания MIME

В таблице 16.2 показаны главные типы и подтипы содержания фрагментов сообщения, определенные на момент выхода книги. Более свежую информацию можно получить в документе *Assigned Numbers*.

Таблица 16.2

Типы содержания (Content Types) для MIME

Тип	Подтип	Описание
text		Текст
	plain	Стандартное почтовое текстовое сообщение (неформатированное).
	richtext	Перемещаемый формат для текстовых процессоров.
	tab-separated values	Значения, разделенные табуляциями
multipart		Сообщение состоит из нескольких частей, отделенных друг от друга разделителями.
	mixed (смешанный)	
	alternative	Пользователь может выбирать из нескольких вариантов, например текст ASCII или Postscript.
	digest	Каждая часть сама представляет собой почтовое сообщение.
	parallel	Связанные между собой части, например видеоклип и соответствующий ему аудиоклип.
	appledouble	Двойной формат Apple
	header-set	Набор заголовков
message (сообщение)		Вложенное сообщение.
	rfc822	Классическое сообщение электронной почты.
	partial	Часть общего сообщения. Обеспечивает пересылку очень длинных сообщений.
	external-body	Содержит указатель на удаленный документ, но не сам документ.
	news	Содержит формат Usenet News.

Таблица 16.2 (*продолжение*)

Типы содержания (Content Types) для MIME

Тип	Подтип	Описание
application (приложение)		Неинтерпретируемое двоичное содержание либо формат определенного приложения.
	octet-stream	Поток октетов
	postscript	Форматировано для вывода или распечатки в формате Postscript.
	oda	Архитектура офисных документов (office document architecture).
	atomicmail	
	andrew-inset	
	slate	
	wita	Пересылка данных для компьютеров Wang (Wang information transfer).
	dec-dx	Формат документов DEC.
	dca-rft	Архитектура содержимого документов IBM, пересмотренный формат (Document Content Architecture, Revisable Format) для текстовых процессоров.
	activemessage	
	rtf	Формат документов Rich text format.
	applefile	Файлы Apple
	mac-binhex40	Файлы компьютеров Macintosh, преобразованные к пересылке (формат binhex40).
	news-message-id	Идентификатор сообщения сетевых новостей
	news-transmission	Пересылка сетевых новостей
	wordperfect5.1	Формат текстового процессора Word Perfect версии 5.1
	pdf	Формат Postscript для приложения Adobe Acrobat.
	zip	Сжатие данных.
	macwriteii	
	msword	Формат MS Word
	remote-printing	Удаленная печать
image		Данные графического изображения.
	jpeg	Формат Joint Photographic Experts Group, определяющий специфическую схему сжатия изображений.
	gif	Формат Graphics Interchange Format (для графики).
	ief	Формат Image exchange format.
	tiff	Формат Tag image file format.
audio		Аудиоданные
	basic	Основной аудиоформат
video		Видеоклипы.
	mpeg	
	quicktime	

16.13.4 Кодирование содержания

RFC 822 определил исходной формат для текстовых сообщений Интернета. Содержание почтового сообщения состоит из последовательности строк, завершающихся <CR> <LF>. Максимальная длина каждой строки (включая <CR> <LF>) определена в 1000 символов.

Как должны кодироваться для пересылки различные типы содержания сообщений MIME? Методы кодирования определены отдельно для каждого типа. Например, для SMTP можно использовать:

- Неэффективный способ кодирования, который представляет двоичные данные как текст, если можно будет доставить сообщение на принимающий агент пересылки почты только таким способом.
- Эффективный способ кодирования, когда получатель поддерживает такой способ.

Методы кодирования представлены в таблице 16.3. Если используется не обычный метод NVT USASCII, а другой, то он должен быть явным образом определен в заголовке Content-Transfer-Encoding. Например:

```
Content-Transfer-Encoding: base64
Content-Transfer-Encoding: Quoted-printable
```

Таблица 16.3

Методы кодирования

Метод	Описание
7bit	Обычные строки текста NVT USASCII.
quoted-printable	Содержимое по большей части представляет собой обычный текст ASCII, но дополнительно имеется несколько особых символов. Каждый такой символ представлен специальной последовательностью обычных текстовых символов.
base64	Все содержание отображается к виду, представленному обычными символами.
8bit	Сообщение организовано как последовательность строк, заканчивающихся на <CR> <LF> и имеющих длину не более 1000 символов. Однако могут быть включены 8-разрядные коды.
binary	Правильное представление двоичных данных.
x-token-name	Любой экспериментальный метод кодирования должен иметь название, начинающееся с "x".

16.13.5 Метод кодирования указанными печатными символами

Мётод кодирования указанными печатными символами (quoted-printable encoding method) используется для сообщений, содержащих только небольшое число символов, не принадлежащих основному множеству ASCII. Эти символы отображаются в специальные последовательности, в то время как большая часть сообщения остается в своей естественной форме. Кодирование выполняется как:

= шестнадцатеричный код для символа

Например, символ перевода формата (X'0C) будет закодирован как *=0C*.

16.13.6 Метод кодирования Base64

Метод кодирования Base64 преобразует любой тип данных к большему в 3 раза количеству текстовых символов. Данные разделяются на части по три 8-разрядных байта. Например:

```
10001000 00110011 11110001
```

Для преобразования эта последовательность сначала разделяется на четыре 6-разрядные группы:

```
100010 000011 001111 110001
```

Каждая группа интерпретируется как число:

```
34 3 15 49
```

Полученные числа заменяются соответствующими символами из таблицы 16.4.

Таблица 16.4

Кодирование Base64

Значение	Код	Значение	Код	Значение	Код	Значение	Код
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

Если общее число октетов не кратно трем, то в конце сообщения останутся 1 или 2 октета. Они дополняются нулевыми битами и кодируются. 1 октет транслируется в 2 символа со следующими далее двумя знаками равенства (==), 2 октета — в 3 символа со следующим далее одним знаком равенства (=).

16.14 Протокол POP

Протокол почтового офиса (Post Office Protocol POP) применяется для пересылки сообщений с почтового сервера на настольную или переносную компьютерную систему.

Спецификация POP определяет множество различных функций, например возможность просмотра списка поступивших сообщений (и их размеров) для извлечения или удаления некоторых из них. Однако многие реализации обычно просто пересылают всю поступившую почту на систему пользователя, которому предоставляется возможность сохранить копии всех сообщений на сервере или удалить их после загрузки на свою систему.

Настольные системы используют POP для загрузки почты, а SMTP для ее отправки. В большинстве случаев сервер загрузки почты одновременно является и входным почтовым шлюзом (см. рис. 16.8). Однако клиентское приложение позволяет применять различные системы для сервера POP и входного шлюза.

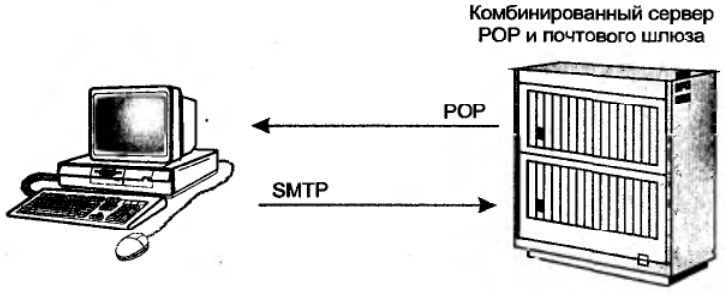

Рис. 16.8. Комбинация сервера POP и системы почтового шлюза

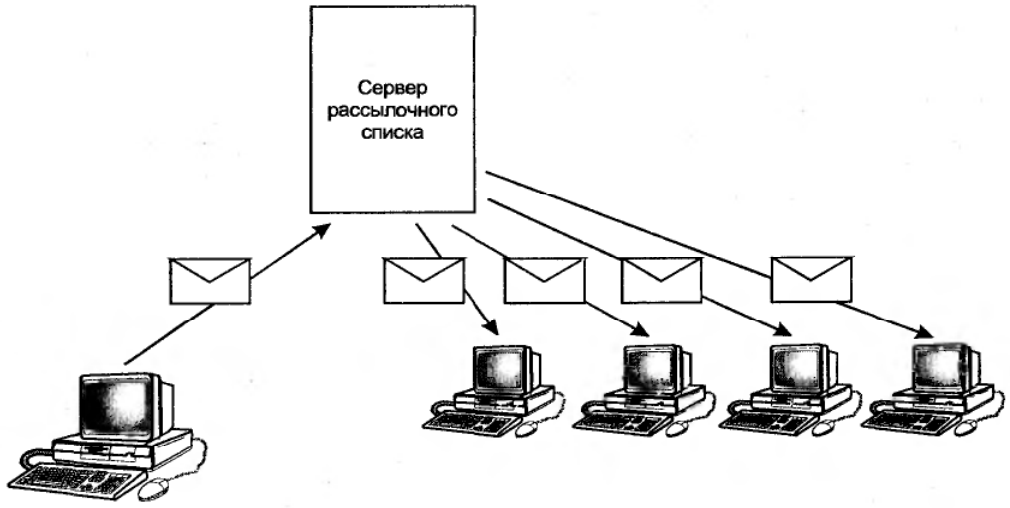

Рис. 16.9. Сервер рассылочного списка

16.15 Другие почтовые приложения

В Интернете существует множество *рассылочных списков* (mailing lists), которые обеспечивают своим подписчикам обмен вопросами и ответами, а также доступ к последним новостям по определенной тематике, будь то вакантные рабочие места, новые компакт-диски или проблемы компьютерной безопасности.

Пользователь подписывается на рассылочный список, отсылая запрос на объявленный почтовый ящик. Попавшие в этот ящик сообщения отправляются всем подписчикам рассылочного списка (см. рис. 16.9). Бесплатное программное обеспечение для работы с рассылочными списками доступно в Интернете, например, популярная программа *Majordomo* имеет версии для различных операционных платформ.

16.16 Производительность

Служба агента пересылки почты использует ресурсы диска, памяти, процессора и полосы пропускания сети. Почтовые службы интенсивно применяются, что создает достаточно напряженный сетевой трафик.

Сообщения должны быть сохранены на время ожидания пересылки. Обычно почтовые службы реализуются на серверах. Пользователям нужно зарегистрироваться в операционной системе такого сервера и получить доступ к своему почтовому ящику. Очень трудно заранее предвидеть объем дискового пространства, необходимого для работы почтового приложения.

Поскольку почтовая служба работает автоматически, некоторые сообщения могут навечно застрять в очереди агента пересылки почты. Администратору очень важно определить период тайм-аута для каждой почтовой операции, чтобы исключить непроизводительное использование компьютерных ресурсов.

16.17 Безопасность

16.17.1 Проблемы с программой sendmail

Из всех почтовых систем наиболее широко используется *sendmail*, большая и сложная программа, реализующая множество функций, включая трансляцию почтовых псевдонимов и расширение рассылочных списков.

Поскольку *sendmail* работает через SMTP, она выполняется поверх сеанса *telnet* в режиме NVT. Следовательно, пользователь легко может соединиться с *sendmail* через порт 25 и попытаться проникнуть в компьютер. К сожалению, *sendmail* не обеспечивает должного уровня безопасности системы.

Однако эта проблема имеет простое решение. Существуют гораздо более простые и надежные почтовые программы, обеспечивающие требуемый уровень защиты и работающие через SMTP. Их и следует применять при необходимости. Если же потребуются расширенные функции программы *sendmail*, то можно применить одну из простых программ контроля входящей почты для *sendmail*.

16.17.2 Безопасность электронной почты

Иногда злоумышленникам удается отследить выполнение операций по пересылке почты. К сожалению, при этом не менее легко фальсифицировать само почтовое сообщение. Поэтому рассмотренные в главе 3 методы повышения безопасности следует применить и к электронной почте, использовав аутентификацию и шифрование сообщений.

16.17.3 Secure MIME (S/MIME)

Secure MIME (MIME с защитой) предохраняет содержимое почтового сообщения с помощью общедоступных ключей и симметричных ключей сеансов. Общедоступные ключи позволяют организовать надежную защиту доступа для владельцев прав на электронные сообщения через иерархию цифровых сертификатов, формат которых определен в стандарте X.509 (см. раздел 16.19.1).

16.18 Обмен сообщениями через X.400

Всемирный телекоммуникационный союз (ITU) несет ответственность за поддержку международных коммуникаций и выпуск *рекомендаций* для обеспечения телеграфной, телефонной и факсимильной связи между странами.

Сектор стандартов этой организации (Telecommunication Standardization Sector of International Telecommunications Union — ITU-T) выпустил множество спецификаций для новейших технологий. Ранее этот сектор именовался International Telegraph and Telefone Consultative Committee (CCITT). Как уже отмечалось в главе 4, CCITT отвечал за коммуникационные стандарты для данных X.25.

В рамках CCITT за период 1981—1984 гг. была создана рабочая группа, разработавшая набор рекомендаций X.400 для управления службами международного обмена электронными сообщениями. Эти рекомендации были одобрены и утверждены ISO. В 1986 г. стандарты подверглись переработке, однако современные реализации основаны на спецификациях 1984 г. Приведем несколько характеристик X.400:

- Представлено определение служб "сохранить-и-переслать", используемых в электронной почте и называемых *межперсональными сообщениями* (interpersonal messaging), равно как и других подобных приложений.
- Специфицировано общее международное описание уровней вложенности пересылаемых сообщений и поддержки национальных алфавитов.
- Указаны возможности пересылки различных типов данных, а не только текстовых (например, двоичных данных, изображений или оцифрованных звуковых файлов).
- По желанию отправителя можно указать доставку на заданную систему получателя и представить подробное описание действий при невозможности доставки. Рассмотрена необязательная возможность для получателя послать сигнал конечному пользователю о прибытии почтового сообщения. Сознательно не специфицируется способ получения почты. Это может быть просмотр содержимого почтового ящика, чтение отдельных сообщений или нажатие на определенную функциональную клавишу для подтверждения приема.
- Поддержка приоритетов почтовых сообщений.
- Возможность преобразования к формату другого физического носителя, например для доставки почты по факсу или просмотра документа для последующей отправки как сообщения электронной почты.
- Описание удобных для пользователей идентификаторов отправителя и получателя.
- Использование формального обрамления, содержащего поля для отслеживания пути доставки сообщения и сбора управляющей информации о перемещении почты.

X.400 определяет стандарт для обмена почтовыми сообщениями между национальными правительственными учреждениями. Его можно рассматривать и как стандарт шлюза. Разработчики различных лицензионных почтовых систем реализуют программное обеспечение для преобразования своих сообщений в/из формата X.400, расширяя возможности своих приложений.

X.400 получил поддержку в Европе, а также был одобрен и правительственными агентствами США.

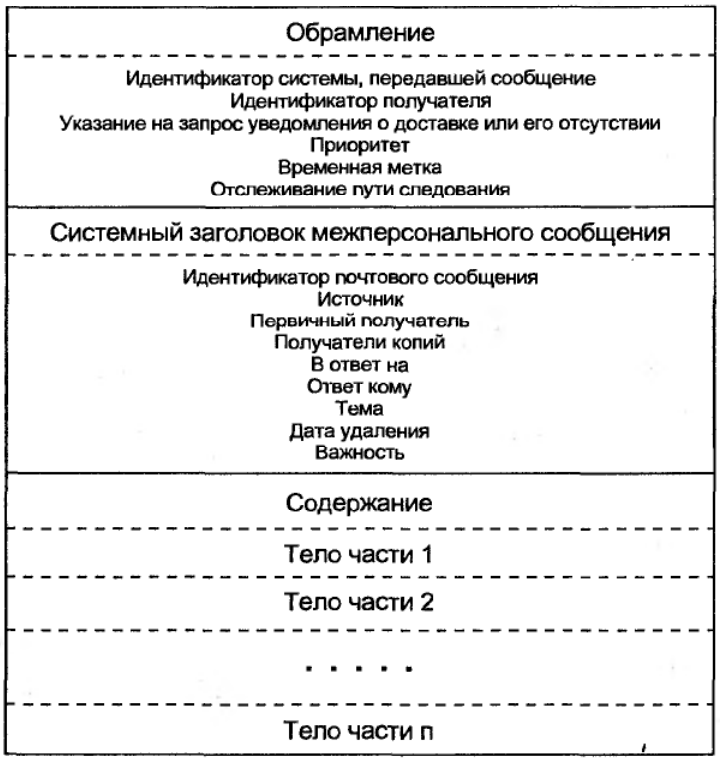

Рис. 16.10. Формат межперсонального сообщения в формате X.400

16.18.1 Пример сообщения X.400

В отличие от стандартов Интернета X.400 не требует 7-битного кода ASCII и взаимодействия по NVT. Поля сообщения форматируются в соответствии со спецификацией BER от ISO (см. главу 20), что предполагает для каждого поля шестнадцатеричный идентифицирующий код и значение длины поля. На рис. 16.10 показан обобщенный пример сообщения, иллюстрирующий основные возможности X.400.

16.18.2 Именование получателей в X.400

Как представляют людей в обычном разговоре? Можно сказать: "Мери Джонс, технический консультант Милуокского подразделения компании MCI", либо "Жак Брюн, который живет по адресу: Франция, Париж, авеню Сентраль, 10". Разработчики стандарта X.400 попытались создать универсальную систему именования, подобную обычному способу идентификации людей.

Имя отправителя и получателя имеют списки атрибутов. Стандарт определяет множество необязательных атрибутов, которые допустимо использовать в произвольной комбинации. Наиболее полезными для систем электронной связи можно считать следующие атрибуты:

- Название страны
- Имя административного домена
- Собственное имя (например, Джон X. Джонс III)
- Название организации
- Название подразделения организации
- Имя собственного домена
- Атрибуты домена

Собственные (личные) домены имеют службы коммерческой электронной почты или корпоративные почтовые системы на основе лицензированных почтовых программных продуктов.

Атрибуты домена позволяют использовать имена существующих почтовых систем, встраивая их в идентификатор X.400. Это очень важное свойство обеспечивает способность шлюзов X.400 переключаться между лицензированными почтовыми системами, а также между обычными лицензированными программами или совместимыми с X.400 системами.

16.18.3 Взаимодействие между X.400 и почтой Интернета

Поскольку оба эти протокола обеспечивают службы "сохранить-и-переслать" (store-and-forward), между ними возможно взаимодействие через специальные шлюзы. Несколько документов RFC специфицируют отображение почтовых сообщений Интернета в формат сообщений X.400.

16.19 Каталоги ISO/ITU-T

Создание правильного идентификатора для получателя системы X.400 может быть достаточно трудным. Выбранные атрибуты могут радикально меняться для различных пользователей. Сразу после завершения работ над X.400 стало ясно, что необходима специальная служба каталогов. Для этого в 1985–1988 гг. CCITT подготовила рекомендации X.500, определяющие службу каталогов и протоколов. Несколько исследовательских и коммерческих ассоциаций создали экспериментальные реализации X.500.

Стандарт каталогов охватывает многое. Каталог X.500 является распределенной базой данных, содержащей информацию различного типа. Например:

- Имена людей
- Почтовые адреса
- Идентификаторы пользователя для почты X.400
- Почтовые идентификаторы в стиле Интернета
- Номера телекса и факса
- Номера телефонов
- Имена и размещение принтеров

База данных X.500 в конечном счете будет содержать информацию, которая поможет пользователям найти сетевой ресурс любого типа.

16.19.1 Модель каталога

Информационная база каталога (Directory Information Base) распределена среди группы баз данных, управляемых *агентами обслуживания каталогов* (Directory Service Agent — DSA). Пользователи обращаются к каталогам через *пользовательский агент каталога* (Directory User Agent — DUA). DUA обеспечивает пользовательский интерфейс для интерактивных запросов и изменений, пересылая запросы пользователя в DSA.

Стандарты X.500 определяют комплексный формальный протокол управления взаимодействием между DUA и DSA. Облегченный протокол доступа к каталогам Интернета (Internet lightweight directory access protocol — LDAP) упрощает доступ к службе каталогов. Существует и протокол DSA-DSA, позволяющий DSA пересылать запросы пользователей или загружать копии отдельных частей информационной базы каталогов.

Существует множество структурных сходств между системой каталогов X.500 и Domain Name System. Обе представляют собой распределенные базы данных и имеют иерархическую древовидную структуру. Пользователи взаимодействуют с сервером через локальный клиент, а сервер может организовать распространение инициированного пользователем запроса.

Стандарт X.500 содержит метод проверки аутентификации записей каталогов. Запись проверяется на соответствие зашифрованным сертификатам, извлекаемым из доверенного источника. Формат сертификата определен в стандарте X.509.

16.20 Дополнительная литература

RFC 821 определяет протокол Simple Mail Transfer Protocol, а RFC 822 описывает формат сообщений Интернета. RFC 1939 специфицирует протокол Post Office Protocol, используемый для пересылки почты между настольными рабочими станциями и почтовым сервером.

RFC 1521 и 1522 посвящены MIME. Существует множество добавлений и изменений, определенных в других RFC. Типы MIME опубликованы в документе *Assigned Numbers*, а процедуры регистрации — в RFC 1590. RFC 1848 определяет структуру безопасности для MIME. Спецификация S/MIME разработана компанией RSA Data Security, Inc.

Улучшенные службы рассматриваются в RFC 1869, а RFC 1652 определяет SMTP Service Extension для транспорта 8BITMIME.

X.400 был первоначально издан как часть рекомендаций CCITT 1984 г. и затем обновлен в рекомендациях от 1988 г. ISO опубликовала собственную версию X.400 в ISO 10021, составленном из нескольких отдельных частей. X.500 появился в рекомендации CCITT от 1988 г.

В настоящее время RFC 1327 и 1495 определяют отображение между элементами X.400 и классическим форматом по документу RFC 822. Это отображение часто обновляется, поэтому лучше обратиться к текущей версии документа RFC. RFC 1496 обсуждает трансляцию в MIME, а RFC 1506 служит руководством по шлюзам между X.400 и почтой Интернета. Несколько других RFC обсуждают трансляции адреса получателя почты. К RFC 1777 и RFC 1798 можно обратиться за описанием облегченного протокола доступа к каталогам.

Глава

17

Сетевые новости

17.1 Введение

Ежедневно через *сетевые новости* (Usenet News) Интернета распространяется самая свежая информация о науке, технологии, компьютерах, экономике, спорте, музыке, образовании и т.д. *Группы новостей* (news group) подобны службам электронных досок объявления (bulletin-board). Новости доступны в форме *статей* (articles), которые *посылаются* в соответствующую группу.

Сегодня существуют тысячи общедоступных и личных групп новостей, содержащих сведения, которые трудно найти в других местах. Часто публикация в группе имеет список вопросов и ответов, связанных с тематикой группы. Иногда поток информации в группе следует в одном направлении — группы служат способом распространения отдельными лицами или организациями общедоступной информации.

Каждая группа новостей обслуживается администратором первичного сервера новостей. Если группа новостей личная, то все статьи могут полностью находиться на таком сервере, а пользователи получают доступ к информации. Публикация в общедоступной группе новостей обычно распространяется от первичного сервера на сотни других, расположенных по всему миру.

Приложения для работы с новостями обеспечивают возможности, далеко выходящие за рамки исходных досок объявления Интернета. Такие приложения часто используются организациями для публикации внутренней информации. Можно сказать, что такие программы изменили обычный издательский бизнес. Публикации на информационных серверах крупнейших агентств новостей, подобных AP, UPI или Рейтер, доставляются своим подписчикам через протокол работы с новостями Интернета.

17.2 Иерархия групп новостей Интернета

Уже созданы тысячи групп новостей Интернета. Каждая из них имеет имя, отражающее тематику группы. Имена групп организованы в древовидную структуру (см. рис. 17.1).

В отличие от других иерархических имен, с которыми мы уже познакомились в этой книге, имена групп новостей читаются *сверху вниз*. Например:

rec.sport.basketball.college

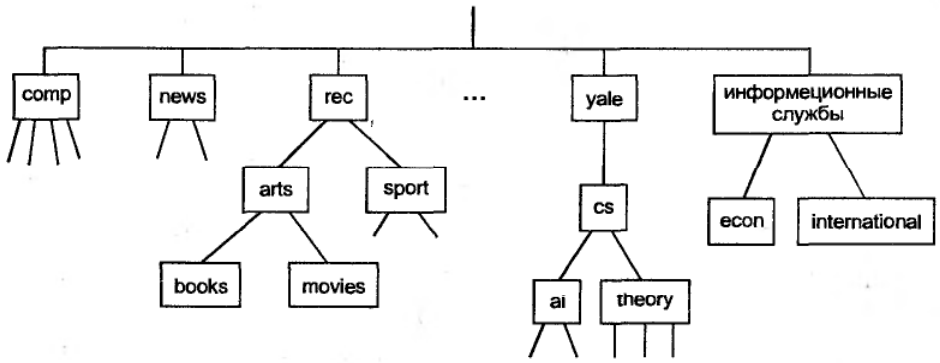

Рис. 17.1. Иерархия групп новостей

17.3 Агенты новостей

Как и пользовательские агенты, позволяющие получать и отправлять почтовые сообщения, *агенты новостей* (news agent) разрешают пользователям подписываться на группы новостей, читать статьи из групп и публиковать собственные статьи в группе.

17.4 Модель новостей

Клиентский процесс новостей взаимодействует с сервером сетевых новостей по *протоколу пересылки сетевых новостей* (Network News Transfer Protocol — NNTP). Клиентский процесс может размещаться в агенте новостей конечного пользователя или на сервере новостей того же уровня. Протокол NNTP обеспечивает следующие возможности:

- Сервер новостей может получать новости от другого сервера новостей.
- Клиентский агент новостей может получать новости от сервера новостей.
- Клиентский агент новостей может публиковать статьи на сервере новостей.

На рис. 17.2 показано, как клиент извлекает новости из сервера по протоколу NNTP, а серверы обмениваются новостями по этому же протоколу.

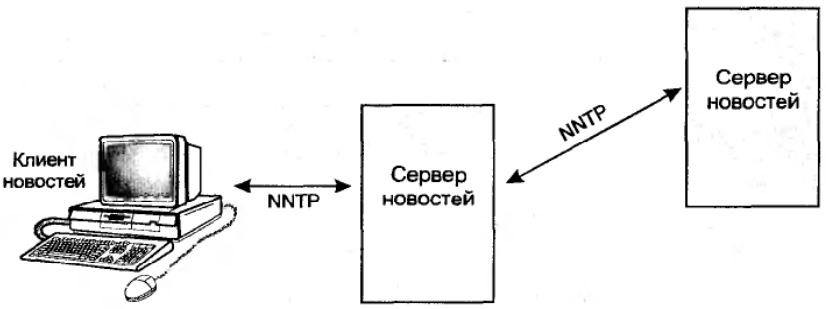

Рис. 17.2. Запрос и обмен новостями

17.5 Сценарий NNTP

Как и SMTP, протокол NNTP работает поверх сеанса *telnet* в режиме NVT. Показанный ниже диалог демонстрирует взаимодействие по пересылке новостей. В данном случае клиент:

- Соединяется с сервером
- Запрашивает у сервера список поддерживаемых команд
- Запрашивает список групп новостей, которые были созданы после 23 октября 1995 г.
- Обращается к группе новостей *news.answers*
- Читает статью из этой группы

```
200 yale InterNetNews NNRP server INN 1.4
22-Dec-93 ready (posting ok)
```

Сервер идентифицирует себя и указывает на возможность публикации статей.

```
help
100 Legal commands
  authinfo user Name|pass Password
  article [MessageID|Number]
  body [MessageID|Number]
  date
  group newsgroup
```

Поддерживаемые на сервере команды

```
  head [MessageID|Number]
  help
  ihave
  last
  list
  [active|newsgroups|distributions|schema]
  listgroup newsgroup
  mode reader
  newgroups yymmdd hhmmss ["GMT"]
      [<distributions>]
  newnews newsgroups yymmdd hhmmss ["GMT"]
      [<distributions>]
  next
  post
  slave
  stat [MessageID|Number]
  xgtitle [group_pattern]
  xhdr header [range|MessageID]
  xover [range]
  xpat header range|MessageID pat [morepat...]
  xpath xpath MessageID

Report problems to <usenet@cs.yale.edu>
.
```

newgroups 951023 010000

Эта команда запрашивает список *групп новостей*, созданных после 23 октября 1995 г. (с часу ночи)

```
231 New newsgroups follow.
rec.music.iranian 14 1 y
soc.atheism 0 1 m
soc.culture.jewish.parenting 1 1 m
soc.culture.rep-of-georgia 3 1 y
```

newnews news.answers 951020 110101

Документы FAQ (часто задаваемые вопросы) публикуются в *news.answers* и содержат сведения по различной тематике. Команда запрашивает список новых FAQ, опубликованных после 20 октября 1995 г. (от 11:01).

```
230 New news follows
<Unix-faq/faq/part2_814199602
@rtfm.mit.edu>
<Unix-faq/faq/part3_814199602
@rtfm.mit.edu>
<Unix-faq/faq/part4_814199602
@rtfm.mit.edu>
. . .
```

Выводится очень большой список.

Показывает подмножество списка.

```
<Skydiving-faq_814424705
@frc2.frc.ri.cmu.edu>
. . .
<Civil-war-usa/faq/part1_814453424
@rtfm.mit.edu>
<Civil-war-usa/faq/part2_814453424
@rtfm.mit.edu>
. . .
<461fkk$lt2@cst715.iac.honeywell.com>
<461flf$lt2@cst715.iac.honeywell.com>
. . .
.
group news.answers                    Переход к группе news.answers.
211 321 52807 53147 news.answers
Article                               Запрос просмотра статьи.
<461fkk$lt2@cst715.iac.honeywell.com> Это длинный заголовок.
220 0 article                         Домашним хостом для группы служит
                                      iac.honeywell.com.
<461fkk$lt2@cst715.iac.honeywell.com>
Path:
yale!yale.edu!spool.mu.edu!
  howland.reston.ans.net!newsfeed.
  internetmci.com
!in2.uu.net!news.iac.honeywell.com!dwe
From: dwe@eng.iac.honeywell.com (Dave Eaton)
. . .
Archive-name:
  sw-config-mgmt/cm-tools
Last-modified: 1995/10/25
Version: 2.5                          Наконец добрались до начала статьи.
Posting-Frequency: monthly
.=.=.=.=.=.=.=.=.=.=.=.=.=.=.=.=.=.=.
Configuration Management Tools Summary
.=.=.=.=.=.=.=.=.=.=.=.=.=.=.=.=.=.=.
This is the newsgroup comp.
  Software.config-mgmt
  "Frequently Asked Questions"
(FAQ) posting of a Software
  Configuration Management tools summary.
  This is part 2 of the 3 part FAQ.
( ... и т.д.)

                                      Конец статьи обозначен строкой,
                                      содержащей только символ точки.
Quit                                  Конец сеанса.
205
Connection closed by foreign host.
```

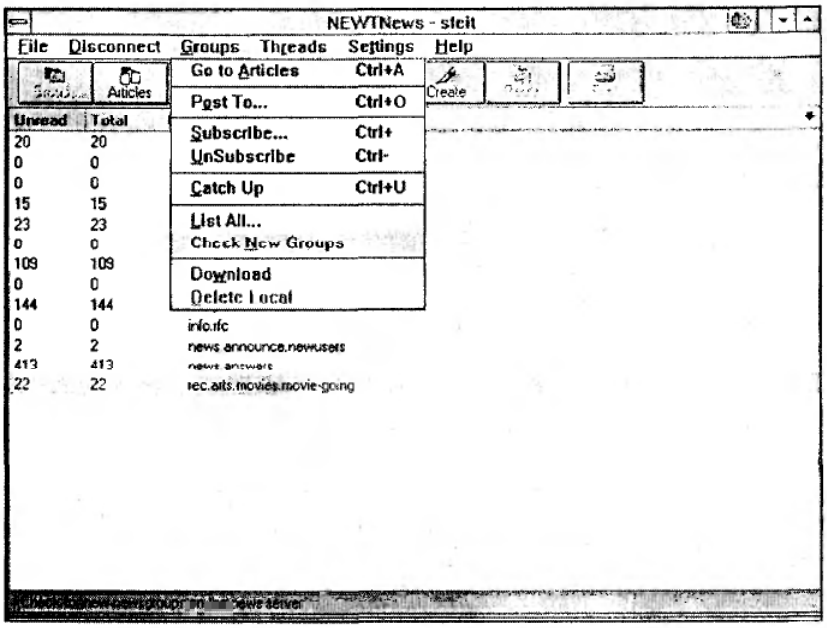

Рис. 17.3. Пункты меню для групп новостей

17.6 Применение агентов новостей для настольных систем

Рассмотрим, как будет выглядеть аналогичный диалог для агента настольной системы. На рис. 17.3 показан вывод новостей в *Chameleon*. Список новых групп новостей выводится щелчком мыши на соответствующем пункте меню.

На рис. 17.4 показан отслеживаемый набор групп новостей (на которые *подписался* пользователь).

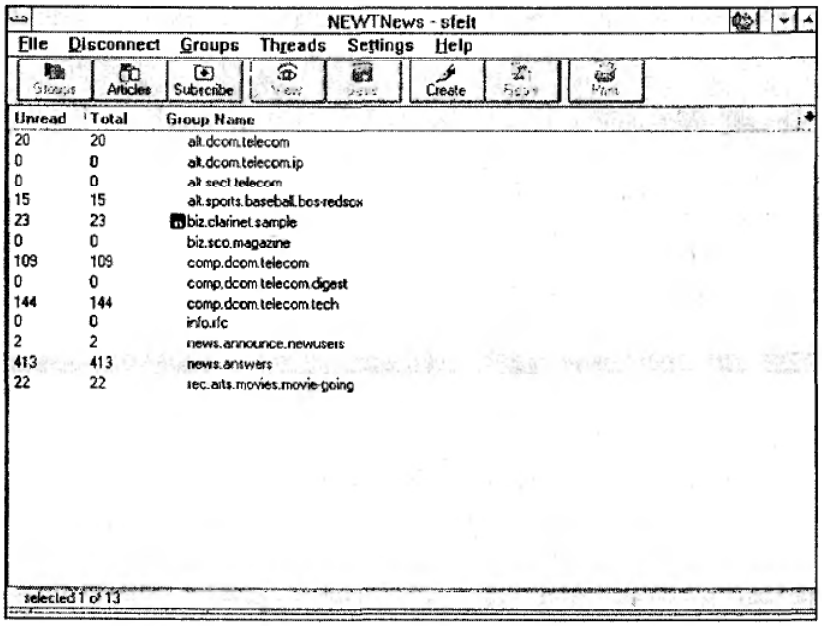

Рис. 17.4. Просмотр групп, на которые подписался пользователь

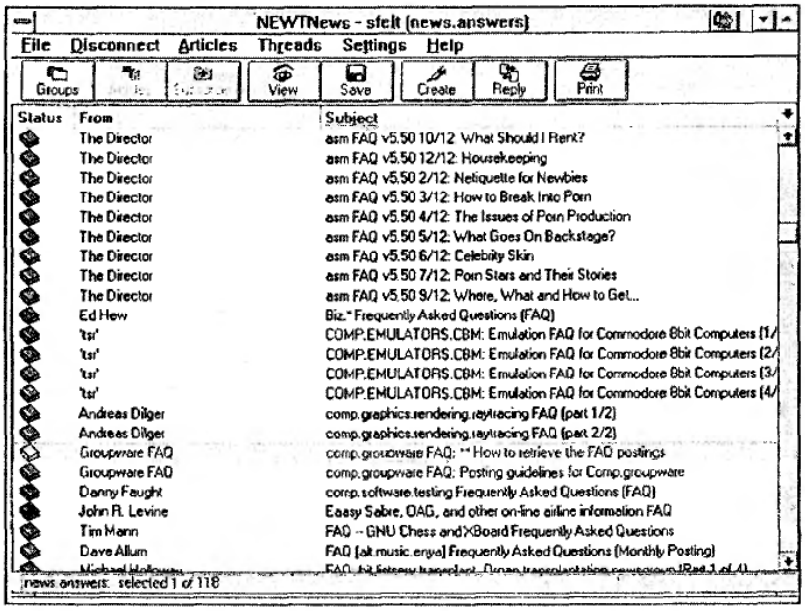

Рис. 17.5. Список непрочитанных статей из группы *news.answers*

Список непрочитанных статей из популярной группы *news.answers* запрашивается двойным щелчком мыши на строке *news.answers*. Результат этой операции представлен на рис. 17.5, а сама статья — на рис. 17.6. Длинный заголовок статьи можно не выводить, если только этого не захочет сам пользователь.

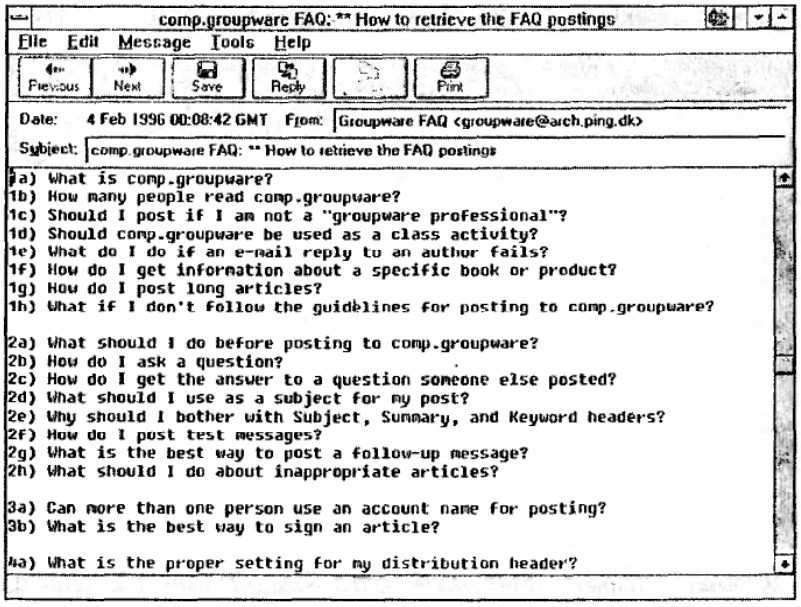

Рис. 17.6. Вывод выбранной статьи

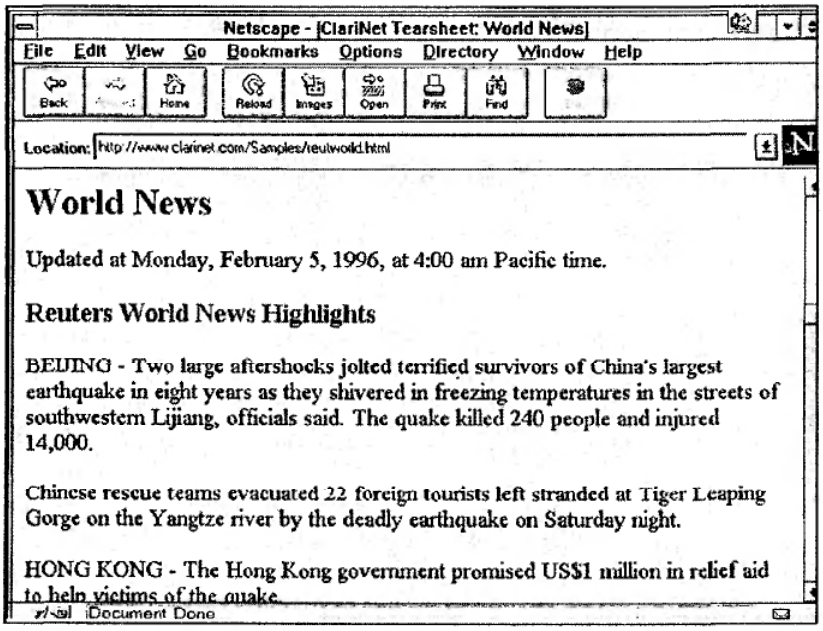

Рис. 17.7. Статья сетевых новостей

На рис. 17.7 показан вывод статьи из группы новостей в браузере WWW (в данном случае — *Netscape Navigator*), применяющемся для чтения статей. Сама статья была написана информационным агентством Рейтер и опубликована в электронном виде через службу новостей *Clarinet*.

17.7 Протокол NNTP

17.7.1 Команды NNTP

Для доступа к статье группы новостей клиентский процесс соединяется с портом 119 сервера новостей. Клиент отправляет серию команд и получает на них ответы. Команды не чувствительны к регистру символов.

Существуют команды для запроса:

- Списка всех групп
- Выбора конкретной группы
- Выбора определенной статьи

Указатель на текущую статью (current article pointer) сервера сохраняет свою позицию на время сеанса пользователя. Команды NNTP перечислены в таблице 17.1.

Необязательный параметр *<распространитель>* (distributions) разрешает пользователю выбрать список категорий высокого уровня, например *comp* или *news*. Список должен заключаться в угловые скобки, а его элементы разделяться запятыми. Например, ниже показан список новых групп новостей, расположенных под *sci*:

```
newsgroup 950601 010000 <sci>
231 New newsgroups follow.
sci.physics.cond-matter 552 1 y
sci.techniques.mass-spec 279 1 m
sci.psychology.consciousness 164 1 m
. . .
```

Таблица 17.1

Команды и параметры NNTP

Команда	Параметры	Описание
article	"<Идентификатор сообщения>", номер статьи или ничего	Извлечение статьи по идентификатору или номеру либо извлечение текущей статьи
body		Извлечение содержимого текущей статьи
group	Имя группы	Переход к указанной группе новостей
head		Вывод заголовка текущей статьи
help		Запрос списка поддерживаемых сервером команд
ihave	<Идентификатор сообщения>	Сервер указывает другому серверу на наличие статьи. При необходимости копия статьи может быть затребована другим сервером.
last		Перемещение указателя текущей статьи на одну статью назад в списке текущей группы
list		Запрос списка групп новостей и количества доступных в них статей
newgroup	Дата, время и необязательный параметр <распространитель>	Запрос списка групп новостей (при необходимости по категориям), созданных после указанной даты и времени
newnews	Группа новостей, дата, время и необязательный параметр <распространитель>	Запрос списка новых статей группы, опубликованных после указанной даты и времени
next		Перемещение указателя текущей статьи на одну статью вперед в списке текущей группы
post		Опубликовать новую статью в группе новостей
quit		Выход
slave		Указывает на запрос от почтового сервера, а не от отдельного клиента
stat	Номер сообщения	Выбор статьи

17.7.2 Коды состояния NNTP

В диалоге из раздела 17.5 видно, что каждый ответ сервера NNTP начинается с числового кода состояния. При этом используются одинаковые для серверов SMTP и FTP правила:

1xx Информационное сообщение

2xx Успешная команда

3xx Команда пока была успешна, нужно послать ее остаток

4xx Команда корректна, но не выполнена по некоторым причинам

5xx Команда не реализована или неверна, либо возникла серьезная ошибка в программе

Как и ранее, вторая цифра кода представляет более специфичную информацию:

x0x Соединение, установка или дополнительное сообщение

x1x Выбор новой группы новостей

x2x Выбор статьи

x3x Функция распространения

x4x Публикация

x8x Нестандартное расширение

x9x Отладочный вывод

17.8 Различия между новостями и рассылочным списком

Приложения для сетевых новостей более эффективны, чем рассылочные списки. Новости хранятся на центральном сервере и доступны для многих пользователей. Несколько пользователей могут одновременно читать новости из совместно используемой базы данных.

Рассылочный список заполняет почтовый ящик ненужной информацией, затрудняя извлечение реально важных сведений. Однако доступ к новостям полностью определяется требованиями пользователя, а более изощренные возможности просмотра новостей на экране (встроенные в агента новостей) делают их более удобными в применении.

Нет необходимости в подписке на группу новостей для чтения или публикации статей. Реально подписка выполняется самим агентом новостей и позволяет отслеживать изменение информации в группе, помечая уже прочитанные статьи.

Многие рассылочные списки автоматически публикуют свои сообщения в группах новостей.

17.9 Дополнительная литература

Протокол NNTP определен в RFC 977.

Глава

18

Службы Gopher и WAIS

18.1 Введение

Система *gopher* была разработана в 1991 г. Центром микрокомпьютеров, рабочих станций и сетей Миннесотского университета. Сотрудники этого центра столкнулись с необходимостью обеспечить поддержку тысяч пользователей, которым понадобилась помощь в изучении компьютеров и общеуниверситетских сетевых ресурсов.

Именно эта тематика и определила основные тенденции в формировании информации службы gopher. Однако требовалось упростить студентам поиск нужного материала среди огромного объема информации. Решением этой проблемы стала служба gopher — иерархическая структура простых меню в архитектуре клиент/сервер.

Gopher обеспечивает простое перемещение в огромном объеме информации и позволяет:

- Находить информацию на локальных сайтах
- Обеспечивать прозрачный доступ к удаленным сайтам
- Извлекать необходимые данные

Возможности gopher по организации и распространению информации были первоначально оценены в колледжах и университетах по всему миру. Далее службы gopher распространились и в правительственных учреждениях.

Впоследствии службы gopher были вытеснены более совершенными и мощными инструментами *WWW* (см. главу 19). Однако еще многие сайты обеспечивают доступ к информации через gopher. Браузеры WWW также способны обеспечивать доступ к серверам gopher, и при этом пользователи могут даже не знать о том, как это выполняется.

18.2 Применение Gopher

Лучший способ знакомства с gopher — применение этой службы на практике. Если пользователь зарегистрировался на многопользовательском хосте и может применять текстовый пользовательский интерфейс, то для запуска локального клиента gopher достаточно ввести команду *gopher*. На рис. 18.1 такой клиент запущен в системе *tigger*, и доступ производится к серверу gopher по умолчанию (в данном случае это сервер компании Global Enterprise Services).

Как показано на рисунке, служба gopher выводит меню. Пункты меню могут приводить к переходу на:

- Текстовый документ
- Изображение
- Следующее меню
- Приложение для поиска
- Сеанс *telnet* с приложением, расположенном на удаленном хосте
- Другому приложению (например, FTP)

Некоторые пункты меню выполняют переход на сервер gopher или другое приложение, которые могут размещаться не на тех компьютерах, где был выполнен запуск клиента gopher.

```
>gopher

       Root gopher server: gopher.jvnc.net

->     1. About this gopher.
       2. Search GES Gopher Tree <?>
       3. GES/
       4. Educational Services/
       5. Internet Resources/
       6. Medical Resources/
       7. Gophers Hosted by GES/
       8. Other Interesting Gophers/
       9. Publishers Online/
      10. WAIS Based Information/
      11. InterNIC/

Press ? for Help, q to Quit, u to go up a menu
```

Рис. 18.1. Доступ к серверу gopher из текстового клиента

Клиенты gopher включены в состав браузеров WWW. На сегодняшний день это наиболее популярный способ доступа к серверам gopher. На рис. 18.2 показан *Netscape Navigator*, выводящий то же самое меню службы gopher, что и на рис. 18.1.

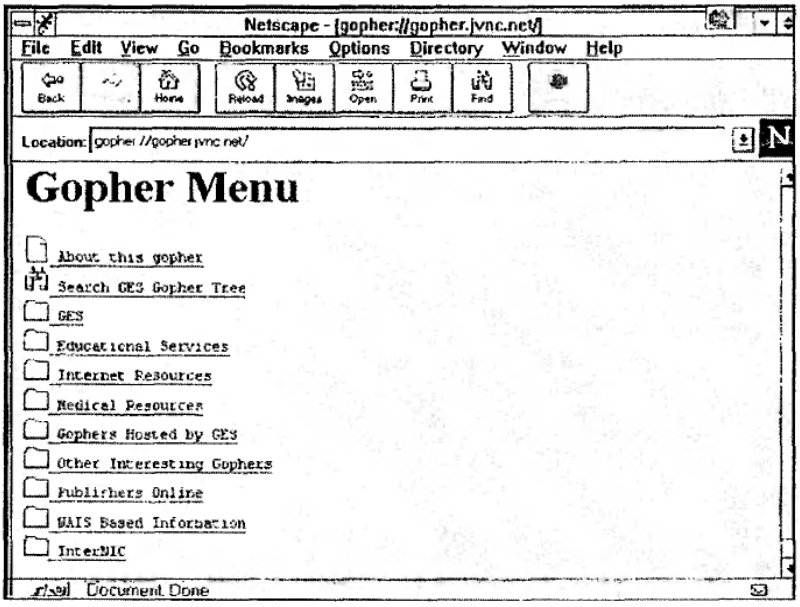

Рис. 18.2. Доступ к серверу gopher из браузера

18.3 Типы информации в gopher

Пункты меню gopher могут содержать различные типы информации. Каждому типу присвоен идентификационный код. Текстовые клиенты gopher указывают на тип информации пункта меню, выводя в конце строки этого пункта специальный тег (tag). Типы, соответствующие им коды и теги

перечислены в таблице 18.1. Графические клиенты gopher отображают типы информации специальными значками.

Таблица 18.1

Типы данных, коды и теги в gopher

Идентификационный код	Тип	Тег	Комментарии
0	Файл	. или пробел	
1	Меню	/	
2	Служба телефонной книги (названа по имени организации компьютерного обслуживания Иллинойского университета — Computer Services Organization of tne University of Illinois).	<CSO>	Простое приложение для базы данных телефонных номеров, адресов электронной почты, почтовых адресов организаций и т.д.
3	Ошибка		
4	Файлы Macintosh в формате BinHexed		
5	Двоичные файлы PC	<PC Bin>	Клиент должен выполнять операцию чтения, пока не будет закрыто соединение TCP.
6	Файлы формата uuencoded операционной системы UNIX		
7	Служба индексного поиска	<?>	
8	Текстовый сеанс *telnet*	<TEL>	При выборе этого пункта можно получить доступ к сеансу *telnet*.
9	Двоичный файл	<Bin>	Клиент *должен* выполнять операцию чтения, пока не будет закрыто соединение TCP.
s	Звуковой файл	<)>	
e	Событие		
c	Приложение для работы с календарем		
T	Текстовый сеанс с устройством 3270	<3270>	При выборе пункта запускается сеанс с терминалом.
g	Графический файл (в формате стандарта "GIF")	<Picture>	
I	Графический файл определенного формата	<Picture>	Способ отображения файла выбирается клиентом.
M	Сообщение MIME	Пустое место или <MIME>	
h	Гипертекстовый документ World Wide Web	Пустое место или <MIME>	

18.4 Иерархия меню Gopher

Меню gopher организовано в виде иерархического дерева. Пункт меню может указывать на следующее меню, которое, возможно, размещается на совершенно другом сайте. Листьями дерева меню являются документы и приложения.

Далее будет видно, что меню gopher реально соответствует каталогам, поэтому применение символа косой черты (/) для указания на следующее меню не случайно. Домашний каталог сервера gopher указывается в его конфигурационных параметрах загрузки. Список пунктов меню по умолчанию формируется из файлов и подкаталогов домашнего каталога сервера.

18.5 Архитектура gopher

Внутренняя структура gopher очень проста. На рис. 18.3, показано, как клиент соединяется с сервером gopher, извлекает меню или файл и закрывает соединение. Выбранный элемент выводится на монитор пользователя. При работе с меню или файлом пользователь уже не соединен с сервером.

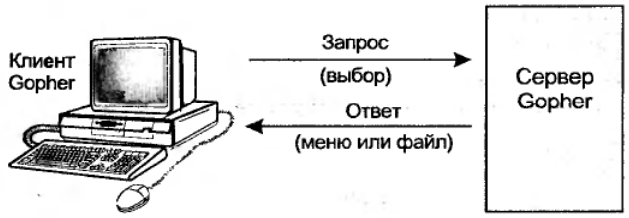

Рис. 18.3. Клиент извлекает информацию из сервера gopher

Сервер gopher *не сохраняет* сведений о клиенте. Клиент соединяется с сервером и запрашивает выполнение некоторой операции. Сервер отвечает на запрос и забывает о нем. Именно это делает gopher простым для запуска и очень надежным. Кроме того, сервер gopher поддерживает одновременно значительно большее число клиентов, чем *telnet* или пересылка файлов. Аналогичные принципы применяются для увеличения эффективности сервера WWW.

18.6 Отличия gopher от FTP

Разработка gopher проводилась для обеспечения удобного и эффективного доступа к архивам пересылки файлов. Каждое меню gopher соответствует некоторому каталогу сервера. В каталоге имеется специальный файл, который:

- Присваивает пунктам меню файлы или подкаталоги
- Определяет ссылки на файлы и каталоги удаленного хоста
- Описывает ссылки на приложения

Несколько примеров будет приведено ниже.

18.7 Протокол gopher

Сеанс gopher выполняется поверх соединения TCP. Обычно используется порт 70 и некоторые правила для соединений *telnet* в режиме NVT. Для получения информации с сервера клиент gopher должен:

- Соединиться с необходимым портом хоста сервера gopher
- Послать на сервер *селекторную строку*, заканчивающуюся на <CR><LF>

Селекторная строка (selector string) определяет выбранный пользователем пункт меню или текстовый документ (а также данные другого типа, например сценарии, исполняемые программы или запросы к базам данных). Пустая селекторная строка, содержащая только <CR><LF>, приводит к возвращению от сервера корневого меню по умолчанию.

Если сервер отошлет меню назад, клиент выводит пользователю список пунктов меню. Однако в меню содержится намного больше информации, чем просто названия пунктов. Каждый посланный сервером пункт меню состоит из последовательности полей, разделенных знаками табуляции. В этих полях содержится:

- Тип пункта меню и его название
- Селекторная строка, которую нужно послать на сервер, чтобы получить этот пункт меню (обычно указывается тип пункта вместе с именем файла или каталога)
- Имя хоста, содержащего данный пункт меню
- Номер порта для доступа к хосту

Содержимое отдельных полей можно увидеть самостоятельно. Ниже показан пример сырого, или необработанного, взаимодействия с сервером gopher компании GES. Обращение происходит по *telnet* к порту 70 сервера, а далее, после установки соединения, просто нажимается клавиша ENTER:

```
> telnet gopher.jvnc.net 70
Trying 128.121.50.10 ...
Connected to nicol.jvnc.net
Escape character is '^}' .
      (Нажатие на ENTER приводит к отправке <CR><LF>)
0About this gopher              0/0about                    nicol.jvnc.net 70
7Search GES Gopher Tree         7/ts                        nicol.jvnc.net 70
1GES                            1/GES                       nicol.jvnc.net 70
1Educational Services           1/Educational_Services      nicol.jvnc.net 70
1Internet Resources             1/Internet_Resources        nicol.jvnc.net 70
1Medical Resources              1/Medical_Resources         nicol.jvnc.net 70
1Gophers Hosted by GES          1/Hosted                    nicol.jvnc.net 70
1Other Interesting Gophers      1/Other_Interesting_Gophers nicol.jvnc.net 70
1Publishers Online              1/Publishers_Online         nicol.jvnc.net 70
1WAIS Based Information         1/WAIS_Based_Information    nicol.jvnc.net 70
1InterNIC                       /                           internic.net 70
.
Connection closed by foreign host
```

Рассмотрим первый элемент списка. 0About this gopher указывает, что данный пункт — это текстовый файл, и определяет его название, *About this gopher*, которое и будет выведено пользователю. Селекторная строка 0/0about повторяет описание типа (0) и ссылается на файл по имени *0about* из домашнего каталога сервера. Если пользователь выберет этот пункт меню, клиент gopher пошлет заданную селекторную строку серверу.

Следующий столбец определяет хост, хранящий данный пункт меню. Мы соединились с *gopher.jvnc.net*, что является псевдонимом для *nicol.jvnc.net*. Наконец последний столбец указывает, что должен использоваться стандартный порт gopher (70). Каждый элемент завершается <CR><LF>.

Следующие несколько элементов описывают подкаталоги домашнего каталога сервера gopher системы *nicol*. Последний элемент указывает на меню по умолчанию (на сервере gopher в InterNIC).

Отметим, что сервер gopher сообщает о завершении пересылки меню, посылая строку, которая содержит только символ точки. Когда пересылается текстовый файл, символ точки используется также для указания на конец файла.

18.8 Файл .names

Простейший сервер gopher можно организовать, сконфигурировав в программе сервера расположение домашнего каталога и запустив эту программу. Главное меню сервера будет содержать список имен файлов и подкаталогов домашнего каталога. Если будет выбран один из подкаталогов, то соответствующий список также будет хранить имена файлов и подкаталогов.

Чтобы заменить созданные имена файлов и каталогов на более содержательные названия, администратор сервера создает в каждом каталоге сервера gopher специальный файл *.names*. Ниже показано несколько элементов такого файла (из домашнего каталога gopher компании GES):

```
# Каталог верхнего уровня
Path = 0/0about
Name = About this gopher
Numb = 1

Path = 1/GES
Name = GES
Numb = 3

Path = 1/Educational_Services
Name = Educational Services
Numb = 4

Path = 1/Internet_Resources
Name = Internet Resources
Numb = 5
```

Пункты меню для соединения с удаленным сервером gopher или для запуска приложений перечислены в файле *.Links*. Элементы такого файла содержат дополнительную информацию: формальное описание типа информации, имя хоста и порт для доступа. Примеры типичных элементов файла *.Links*:

```
Type = 7
Name = Search GES Gopher Tree
Path = 7/ts
Host = nicol.jvnc.net
Port = 70
Numb = 2

Type = 1
Name = InterNIC
Path = /
Host = internic.net
Port = 70
Numb = 11
```

Как показано на рис. 18.4, меню *Internet Resources* (ресурсы Интернета) имеет много ссылок на сеансы *telnet*. Типичный элемент файла *.Links* для сеанса *telnet* имеет вид:

```
Type = 8
Name = CARL System
Path = CARL
Host = pac.carl.org
Port = 23
Numb = 2
```

Tun 8 означает *telnet*, и в этом случае параметр *Path* (путь) определяет идентификатор пользователя (userid), который должен использоваться для регистрации в *telnet*.

```
                    Internet Resources

->      1. Area Code lnfo/
        2. CARL System <TEL>
        3. FreeNet (USA Today) <TEL>
        4. Ftp/
        5. Geographic Server <TEL>
        6. Libraries/
        7. Netfind (Internet White Pages) <TEL>
        8. News /
        9. Pilot Weather Service. [Airplane Pilot] <TEL>
       10. RFC/
       11. Sun Managers/
       12. Sunergy/
       13. Weather By State/
       14. Weather Service <TEL>
       15. World Wide Web <TEL>


Press ? for Help, q to Quit, u to go up a menu
```

Рис. 18.4. Меню Internet Resources

18.9 Служба WAIS

Gopher делает доступными для пользователей множество файлов. Однако пользователи нуждаются в инструменте для поиска в архиве полезных для себя текстовых документов. Большинство серверов gopher имеет поисковое средство — *региональную информационную службу* (Wide Area Information Service — *WAIS*), обеспечивающую, кроме прочего, полномасштабную индексацию текста. Существуют бесплатные и коммерческие версии WAIS (в настоящее время это торговая марка компании WAIS, Inc).

Кроме WAIS, были разработаны другие средства для индексации и поиска. Поисковые приложения очень важны. Они постоянно совершенствуются в конкурентной борьбе за создание эффективной методологии индексации, наиболее функциональных возможностей и ускорения работы.

18.10 Дополнительная литература

Протокол gopher описан в RFC 1436. Бесплатные справочные материалы и программное обеспечение для gopher доступно на сервере Миннесотского университета (*gopher.tc.umn.edu*).

Глава
19

WWW

19.1 Введение

19.1.1 Гипертекст

Идея *гипертекста* (hypertext) известна уже многие годы. Она основана на следующих положениях:

- Выделенные в документе фразы связаны с указателями на другие документы.
- Пользователь может *перейти* на другой документ, щелкнув мышью на выделенной фразе.

Пользователи Microsoft Windows или Macintosh хорошо знакомы с гипертекстом по справочным системам, хотя могли и не слышать о самом термине. Например, меню справки может выглядеть так:

> Saving Files
> Finding and Replacing
> Cutting and Pasting
> Page Formats

Для получения более подробных сведений по каждой из этих тем следует щелкнуть мышью на соответствующем заголовке. В данном случае каждая из выделенных фраз заголовка обеспечивает гипертекстовую ссылку на другой документ. В иных пользовательских интерфейсах такие фразы могут отличаться цветом или иным способом выделения.

19.1.2 Гипермедиа

Идея гипертекста расширяется до понятия *гипермедиа* (hypermedia), когда выделенная фраза указывает на изображение, звуковой файл, видеоклип или иные виды двоичных данных. Изображение может также содержать элементы, щелчок на которых мышью вызывает ссылки на документы, изображения, звуковые файлы или видеоклипы. Такой способ доступа к информации уже давно и успешно используется на компакт-дисках. (Однако наиболее общим свойством гипермедиа-гипертекста следует считать не возможность перехода по ссылкам и не встраивание различных типов информации, а нелинейную структуру самого гипертекста. В отличие от обычного текста, который является линейным и состоит из последовательных строк, гипертекст состоит из отдельных фрагментов, объединенных ссылками. Структура такого текста может быть не только не линейной, но даже и не древовидной. Вместе с множеством достоинств в гипертексте есть один недостаток: чтобы просмотреть последовательно весь документ от начала до конца, придется отслеживать все переходы по ссылкам. Разумеется, этот процесс автоматизирован в современных браузерах WWW, которые выделяют в тексте не только сами ссылки, но и специальным образом отмечают уже просмотренные пользователем ссылки вместе с реализацией функции возврата по последней ссылке.— *Прим. пер.*)

19.1.3 Гипермедиа и WWW

Использование гипермедиа расширяется на сетевую информацию через службу Интернета *World Wide Web* (WWW). В этом случае выделенные фразы могут указывать не только на локальный элемент, но и на любой элемент данных любого удаленного компьютера. Именно эта простая идея лежит в основе пользовательского интерфейса, существенно упрощающего перемещение по Интернету.

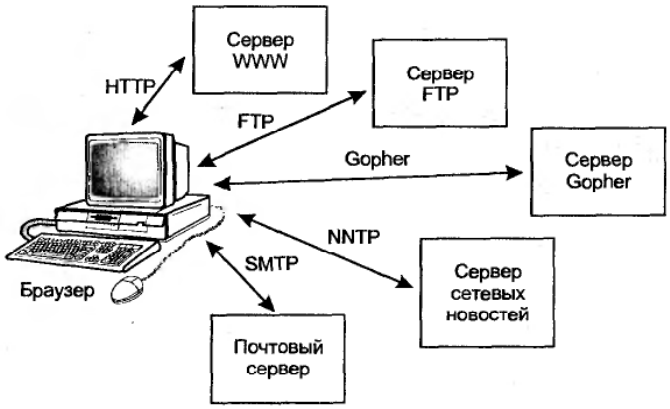

Рис. 19.1. Браузер может работать по нескольким протоколам

19.2 История WWW

Идея WWW возникла среди физиков. Теоретические основы были заложены Тимом Бернерс-Ли (Tim Berners-Lee) из швейцарского центра физических исследований ЦЕРН.

19.3 Браузеры WWW

Толчком к распространению WWW послужило создание Марком Андрессеном в 1992 г. клиента WWW под названием *Mosaic*. В то время Андрессен был аспирантом Иллинойского университета и сотрудником университетского центра по применению суперкомпьютеров (National Center for Supercomputing Applications — NCSA). Mosaic был первым *браузером* (browser) для Интернета, т.е. программой доступа к данным из различных источников, включая гипертекстовые архивы, серверы gopher, поисковые средства баз данных, сайты пересылки файлов и группы новостей.

Как показано на рис. 19.1, браузер может работать по нескольким протоколам, которые требуются для доступа к различной информации. На основе Mosaic был создан мощный коммерческий браузер *Netscape Navigator*, распространяемый компанией Netscape Communications Corporation. На рис. 19.2 представлена домашняя страница этой компании в браузере Netscape.

Использование браузеров и серверов WWW расширяется, равно как и происходит ускоренное совершенствование технологий и протоколов.

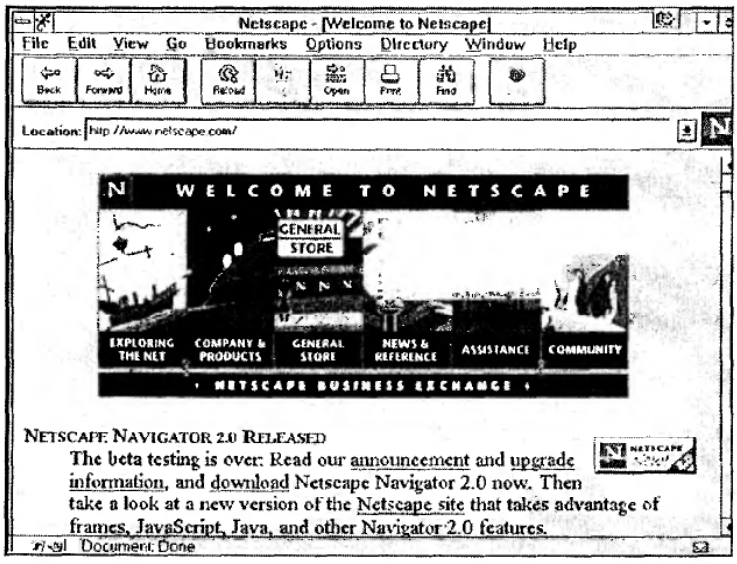

Рис. 19.2. Домашняя страница компании *Netscape* в браузере этой компании.

19.4 URL

Успех WWW обеспечивается и очень важной концепцией унификации. Каждый информационный ресурс WWW идентифицирован *унифицированным указателем ресурсов* (Uniform Resourse Locator — *URL*), иногда называемым и универсальным указателем ;есурсов (Universal Resourse Locator). URL определяет:

- Имя ресурса
- Местоположение ресурса
- Используемый для доступа к ресурсу протокол

URL является частным случаем *универсального идентификатора ресурса* (Universal Resourse Identifier — URI), обеспечивающего единообразный способ именования любых информационных ресурсов.

19.4.1 URL для гипертекста

Если в браузере WWW ввести значение URL гипертекстового документа, браузер извлечет этот документ по *протоколу пересылки гипертекста* (Hypertext Transfer Protocol — *HTTP*). Формат URL для гипертекста:

http://имя-системы/имя-файла

Например:

http://www.ibm.com/index.html

Если указать только:

http://имя-системы

то браузер WWW возвратит по умолчанию домашнюю страницу (*home page*), которая обычно именуется *home.html* или *index.html*. Более общий формат URL для протокола HTTP имеет вид:

http://хост:порт/путь?путь_поиска

Не менее проста структура URL для других протоколов.

19.4.2 URL для gopher

Если в браузере ввести URL:

gopher://gopher.jvnc.net/

то браузер будет работать как клиент gopher и соединится с сервером gopher по имени *gopher.jvnc.net*. Если сервер недоступен на обычном порту (70), но использует другой порт, например 3333, то нужно указать URL в виде:

gopher://gopher.somewhere.edu:3333/

19.4.3 URL для FTP

Пересылка файлов по протоколу FTP может быть выполнена по URL:

ftp://ds.internic.net/

или с указанием определенного файла

file://ds.internic.net/rfc/rfc1738.txt

Для доступа по FTP к сайту с вводом пароля и идентификатора пользователя применяется:

ftp://имя_пользователя:пароль@идентификатор_хоста

Хост можно указать через IP-адрес или имя домена.

Для доступа к файлу URL должен быть похож на:

```
file://ds.internic.net/rfc/rfc1738.txt
```

Отметим, что протокол не указан, однако по умолчанию используется FTP.

19.4.4 URL для telnet

Соединиться по *telnet* поможет:

telnet://ds.internic.net/

Или в более общей форме:

telnet://имя_пользователя:пароль@идентификатор_хоста/

19.4.5 URL для сетевых новостей

URL для группы новостей имеет вид *news.имя_группы*, например:

news:rec.airplane

Сервер новостей не идентифицирован в URL. Вместо этого его название (или адрес) указывается в параметрах конфигурации браузера.

19.4.6 URL электронной почты

URL для отправки электронной почты:

mailto:пользователь@размещение_почты

Как и для новостей, имя или адрес почтового шлюза указывается в конфигурационной информации браузера.

19.4.7 URL для WAIS

Хотя и редко используемый (если вообще когда-либо), URL был определен для доступа к базам данных WAIS по протоколу Z39.50. Например, интерфейс для каталога общедоступного сервера WAIS имеет форму:

wais://cnidr.org/каталог_сервера

В общем случае URL для WAIS имеют формат:

wais:/ /хост:порт/база_данных

wais:/ /хост:порт/база_данных?search

wais:/ /хост:порт/база_данных/тип/путь

На момент выхода книги немногие (если вообще какие-нибудь) браузеры поддерживали протокол доступа к WAIS. Поиск в базах данных обычно выполняется путем заполнения форм и отправки их на сервер WWW, который должен запустить соответствующее поисковое средство.

19.5 Обобщенный формат URL

Обобщая вышесказанное, отметим, что:

- URL начинается с указания используемого протокола доступа.
- Для всех приложений, кроме сетевых новостей и электронной почты, далее следует разделитель ://.
- Затем указывается имя хоста сервера.
- Наконец определяется ресурс (иначе будет извлечен файл по умолчанию).

Для сетевых новостей и электронной почты местоположение нужного сервера новостей и почтового шлюза определяется конфигурационной информацией браузера. Применяется только часть разделителя (:), и в URL не указывается никакой серверный хост.

19.5.1 Специальные символы

Иногда идентификатор ресурса содержит пробелы или иные специальные символы (например, слэш или двоеточие), которые применяются в URL как разделители. Например, имена файлов Macintosh и Windows 95 могут содержать пробелы и другие необычные символы.

Специальные символы в именах ресурсов записываются строкой, начинающейся с символа процентов (%). Такое отображение показано в таблице 19.1.

Таблица 19.1

Отображение специальных символов

Специальный символ	Пробел	/	#	=	;	?	:	~
Представление	%20	%2F	%23	%3D	%3B	%3F	%3A	%7E

19.6 Введение в HTML

Документы WWW с гипертекстовыми ссылками записываются на *языке разметки гипертекста* (Hypertext Markup Language — *HTML*). Гипертекстовые файлы, совместимые с версиями 1 и 2 HTML, обычно имеют имена в формате:

имя_файла.html

Файл, содержащий расширенные возможности версии 3, именуется как:

имя_файла.html3

На компьютерах DOS и Windows применяется суффикс *htm* или *ht3*.

HTML основан на обобщенном *стандарте разметки гипертекста* (Standard Generalized Markup Language — SGML). Основная идея состоит в размещении в документе специальных тегов для идентификации таких элементов, как заголовки, подзаголовки, границы параграфов, маркированные списки, графические символы и т.д.

HTML должен быть независим от платформы, чтобы обеспечить просмотр гипертекстового документа любыми клиентскими устройствами: от неинтеллектуальных терминалов до мощных рабочих станций. Клиенты должны уметь выводить документы на экранах любого размера и использовать локально выбранные шрифты.

Далее мы рассмотрим основы HTML, следуя спецификации HTML версии 3. HTML становится очень большим по объему языком и имеет массу возможностей.

Например, можно не указывать описание структуры сложных форм при записи пересылаемых от клиента на сервер данных. Такие формы могут использоваться для ввода запросов в базу данных или заказов товаров в интерактивных магазинах.

Другая важная способность — это построение изображений с областями для щелчка мышью. Конечный пользователь может щелкать на области в изображении, чтобы выбрать связанный с этой областью документ.

19.6.1 Создание документа на HTML

Некоторые детали отображения документа оставлены клиенту. Браузер настольной системы обычно разрешает конечному пользователю выбрать шрифты для выводимого текста. Текст HTML-документа будет переформатирован согласно размеру окна экрана и выбранного шрифта.

Автор документа HTML может определить следующие элементы:

- Заголовки
- Подзаголовки
- Абзацы
- Ссылки с помощью URL
- Списки
- Предварительно отформатированный текст
- Форматирование символов
- Специальные символы
- Встроенные изображения
- Внешние графические изображения
- Формы для ввода данных
- Карту областей щелчка мышью
- Таблицы и формулы

Включенный в HTML-документ элемент определяется соответствующим *тегом*. Например, тег <TITLE> вводит заголовок документа.

Гипертекстовый документ можно создать, используя обычный текстовый редактор. Однако популярные программы текстовых процессоров обеспечивают подключаемые модули для автоматизации создания тегов и позволяют проводить работу в режиме "Что видим, то и получаем". Существуют специальные программные продукты для создания гипертекстовых документов. В них автоматизировано построение различных элементов и по желанию можно скрыть от пользователя примененные теги.

Хороший способ создания документа HTML состоит в том, чтобы отформатировать документ в обычном текстовом процессоре, а затем применить конвертер для автоматического преобразования в HTML.

Общее понимание принципов работы HTML полезно при рассмотрении способов наиболее эффективного использования любых его средств. Кроме того, постоянно появляются новые возможности в самом языке, которые еще не реализованы в соответствующих инструментах, и такого рода данные могут вводиться только вручную. К счастью, HTML достаточно прост для изучения.

19.6.2 Теги HTML

Тег состоит из названия элемента и параметров, заключенных в угловые скобки (<...>). Ниже мы рассмотрим наиболее широко используемые теги. Символы тегов не чувствительны к регистру, но для постоянства мы будем записывать их только в верхнем регистре.

Большинство тегов применяется парами, показывая начало и конец элемента. Заключительный тег имеет то же самое имя, что и начальный, но начинается с символа слэша < / ... >. Например:

```
<TITLE>Welcome To The Web</TITLE>
```

19.6.3 Общий формат HTML-документа

Несколько тегов служат для определения начала и конца HTML-документа или выделяют в нем заголовок и тело. Например:

`<HTML>`	Начало гипертекстового документа.
`<HEAD>`	Начало заголовка.
`<!--Last Modified on October 21, 1995-->`	Комментарий.
`<BASE HREF = "http://www.abc.com/ind.html3">`	Указывает размещение данного документа.
`<TITLE>Welcome to the Web</TITLE>`	Заголовок, обычно выводимый вверху клиентского экрана.
`</HEAD>`	Конец заголовка.
`<BODY>`	Начало тела документа.
...	
`</BODY>`	Конец тела документа.
`</HTML>`	Конец гипертекстового документа.

19.6.4 Заголовки HTML

Главы, разделы и подразделы документа начинаются заголовками. Можно использовать шесть уровней заголовков, и каждый будет выведен собственным форматом. Например, заголовки первого уровня обычно представлены жирным шрифтом большого размера:

```
<H1>Это заголовок первого уровня — самый главный</H1>
<H2>Заголовок второго уровня можно применять для разделов</H2>
<H3>Существуют еще заголовки уровней с третьего по шестой</H3>
```

19.6.5 Абзацы и разрывы

Автор должен указывать границы абзацев, иначе весь выводимый текст сольется вместе. Клиентская программа обычно объединяет повторяющиеся пробелы и пустые строки в один пробел или пустую строку, если не указано иное форматирование.

Старые версии HTML выделяли абзацы, помещая тег <P> в начале каждого нового абзаца:

```
<P>Это абзац.
<P>Это следующий абзац.
```

Это справедливо и для версии 3, но в ней можно применять и пару тегов, отмечающих начало и конец абзаца:

```
<P> Это абзац.</P>
```

По умолчанию большинство браузеров вставляет между абзацами пустую строку (в версии 3 есть теги для описания другого стиля абзацев, например, для отступа в первой строке). Если нужно начать новую строку, но не новый абзац, используют разрыв:

```
Розы — красные,<BR>
Фиалки — голубые.<BR>
```

19.6.6 Неупорядоченные списки

Неупорядоченный список выводится как последовательность помеченных элементов. Например:

```
<UL>
<LI> Яблоко
<LI> Груша
</UL>
```

В версии 3 определен необязательный заголовок списка и тег конца элемента:

```
<UL>
<LH>Виды фруктов</LH>
<LI>Яблоко</LI>
<LI>Груша</LI>
</UL>
```

19.6.7 Упорядоченные списки

Упорядоченные списки имеют такую же структуру, но элементы нумеруются:

```
<OL>
<LH>Это упорядоченный список.</LH>
<LI>Первый элемент.
<LI>Следующий элемент.
</OL>
```

Как и раньше, тег конца элемента списка (</LI>) и заголовок списка (<LH> ... </LH>) необязательны.

19.6.8 Список определений

Список определений является последовательностью терминов и их определений:

```
<DL>
<LH>Терминология WWW</LH>
<DT>Язык разметки гипертекста (HTML)
<DD>Язык форматирования для записи гипертекстовых документов. Теги документа
идентифицируют такие элементы, как заголовки, абзацы или списки.
<DT>Протокол пересылки гипертекста (HTTP)
<DD>Протокол для запроса и пересылки гипертекстовых документов.
</DL>
```

При выводе это будет выглядеть как:

Терминология WWW
Язык разметки гипертекста
  Язык форматирования для записи гипертекстовых документов. Теги документа идентифицируют такие элементы, как заголовки, абзацы или списки.
Протокол пересылки гипертекста.
  Протокол для запроса и пересылки гипертекстовых документов.

Списки любого типа могут быть вложенными.

19.6.9 Дополнительные теги

Для выделения отдельных частей документа можно воспользоваться горизонтальным разделителем, который пересекает всю ширину выводимой страницы:

```
<P><HR></P>
```

Иногда нужно получить текст, размещенный точно так же, как он был введен. Тег предформатирования (<PRE>) указывает браузеру на вывод текста "как есть":

```
<PRE>
  Этот текст будет показан так,
  как написан, включая отступы.
</PRE>
```

Цитируемый блок текста (block quote) — еще один способ выделения фрагмента в тексте. Обычно это делается сдвигом вправо всего блока. В версии 2 применяется тег <BLOCKQUOTE>.

```
<BLOCKQUOTE>
  Это — цитируемый блок.
  Возможно, он будет выведен пользователю сдвинутым вправо.
</BLOCKQUOTE>
```

В версии 3 название тега сокращено до <BQ>.

19.6.10 Выделение в тексте

Иногда требуется выделить фрагмент текста особым образом, например полужирным шрифтом или курсивом. Это можно сделать двумя способами:

1. Оставить детали вывода на усмотрение браузера

```
<EM> Обычно выводится курсивом. </EM>
<STRONG> Обычно выводится полужирным шрифтом. </STRONG>
<CODE> Обычно отображается моноширинным шрифтом. </CODE>
```

2. Явно указать способ изображения текста:

```
<I> Вывести курсивом. </I>
<B> Вывести полужирным шрифтом. </B>
<U> Подчеркнуть текст. </U>
<S> Перечеркнуть текст. </S>
<TT> Вывести моноширинным шрифтом (как на пишущей машинке). </TT>
<SUB> Подстрочными символами. </SUB>
<SUP> Надстрочными символами. </SUP>
```

Версия 3 имеет много дополнительных свойств, обеспечивая автору разнообразные возможности по управлению выводом текста клиенту.

19.6.11 Ссылки

Чтобы включить в документ ссылку, нужно:

- Использовать теги начала и конца ссылки
- Указать URL связанного со ссылкой документа
- Обеспечить метку для щелчка мышью (обычно выводится подчеркиванием или голубым цветом).

Ниже показан пример ссылки. Символ A определяет название тега, именуемого точкой привязки, или *якорем*. Параметр *HREF* идентифицирует элемент, через который выполняется ссылка. Текст перед разделителем </A> становится меткой для щелчка мышью на этой ссылке:

```
<A HREF= "http://www.abc.com/wwwdocs/showme.html">Щелкните здесь для вывода
дополнительных сведений</A>
```

Не всегда нужно записывать полный URL для связанного документа. Предположим, что документ *showme.html* содержит ссылку на файл *more.html* из того же каталога. Тогда можно записать ссылку как:

```
<A HREF = "more.html">дополнительные сведения</A>
```

Такой способ называется *указанием относительного пути*. Его можно применять и для подкаталогов текущего каталога.

19.6.12 Ссылки на локальные документы

Можно создать ссылку на документ локального хоста. Например, ссылка на локальный документ DOS выглядит как:

```
<A HREF = "file:///c:\webdocs\home.htm">Документ локального хоста</A>
```

Для извлечения такого документа нет надобности в протоколе HTTP. Отметим, что имя хоста не указано — между косыми чертами (///) ничего нет.

Допустимо ссылаться на отдельные места того же самого документа. Сначала маркируется нужное место. В версии 2 это выполняется вставкой точки привязки с использованием параметра NAME:

```
<A NAME = "Раздел3"> 3. Самолеты </A>
```

Затем создается ссылка на это место документа путем указания перед его именем символа диез:

```
См. <A HREF = "#Раздел3"> обратитесь к разделу три </A> за дополнительной
информацией.
```

Если пользователь щелкнет мышью на подчеркнутой фразе (обратитесь к разделу три), клиент "перескочит" на заданное место документа.

В версии 3, вместо маркировки позиции в документе специальным тегом, можно добавить идентификатор к любому уже существующему тегу. Например, ниже мы добавляем идентификатор для тега H2:

```
<H2 ID = "Раздел3"> 3. Самолеты </H2>
```

19.6.13 Изображения

Тег IMG служит для вставки изображения в документ. Тег содержит параметр *SRC*, который определяет URL для файла, имеющего изображение. URL изображений выглядит как любые другие URL. Ссылка на изображение будет выглядеть как:

```
<IMG SRC = "http://www.abc.com/wwwdocs/ourlogo.gif">
<IMG SRC = "bigpic.jpeg">
<IMG SRC = "file:///c:\webdocs\building.gif">
```

На WWW-страницах часто используются изображения в *формате для обмена графикой* (Graphics Interchange Format — GIF). Для сжатия точечных (растровых) изображений служит формат перемещаемой сетевой графики (Portable Network Graphics — PNG). Еще одним популярным форматом является формат *объединенной экспертной группы по фотографии* (Joint Photographic Experts Group — JPEG). Он был разработан для сжатия фотографических изображений, но иногда используется и для других типов графики.

Не имеющие графических возможностей браузеры будут игнорировать теги IMG, если только в них не указан параметр ALT. Например:

```
<IMG SRC = "bigpic.jpeg" ALT = "Памятник Вашингтону">
```

Вместо изображения текстовый браузер выведет строку "Памятник Вашингтону".

19.6.14 Просмотр исходного кода HTML

Чтобы хорошо изучить HTML, нужно познакомиться с исходными кодами документов. Обычно браузер имеет для этого специальный режим, иначе придется сохранить документ на диске и затем просмотреть его в обычном текстовом редакторе.

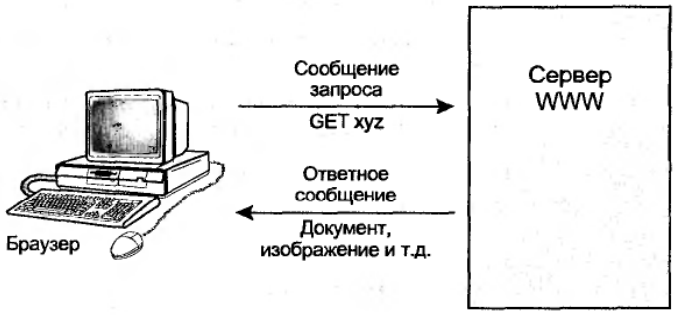

Рис. 19.3. Браузер извлекает страницу гипертекста с сервера WWW.

19.7 Архитектура HTTP

Как и в gopher, извлечение гипертекстового документа достаточно просто. Как показано на рис. 19.3, клиент соединяется с сервером WWW, извлекает часть документа (обычно ее называют страницей.— *Прим. пер.*) и закрывает соединение. Браузер выводит извлеченную страницу, а пользователь может выполнять следующую операцию.

Сервер WWW, предоставляющий только текстовые документы, работает очень эффективно и может поддерживать множество пользователей. Однако объем информации резко увеличивается при работе и перемещении графических изображений или звуковых файлов. Эти объекты имеют значительный размер, и для их пересылки требуется большее количество ресурсов сети и центрального процессора, чем для обмена обычными текстовыми файлами. Более того, некоторые запросы вызывают программы, формирующие ответную информацию. Для этого нужно еще больше системных ресурсов.

19.7.1 Прокси-сервер

Прокси-сервер WWW используется для доступа к внешним серверам WWW клиентов, расположенных в пределах зоны безопасности сети. В этом случае браузер клиента конфигурируется для отправки всех запросов прокси-серверу, который, в свою очередь, взаимодействует с реальным сервером WWW и возвращает клиенту полученный результат. На рис. 19.4 показан клиент, обращающийся к серверу WWW через прокси.

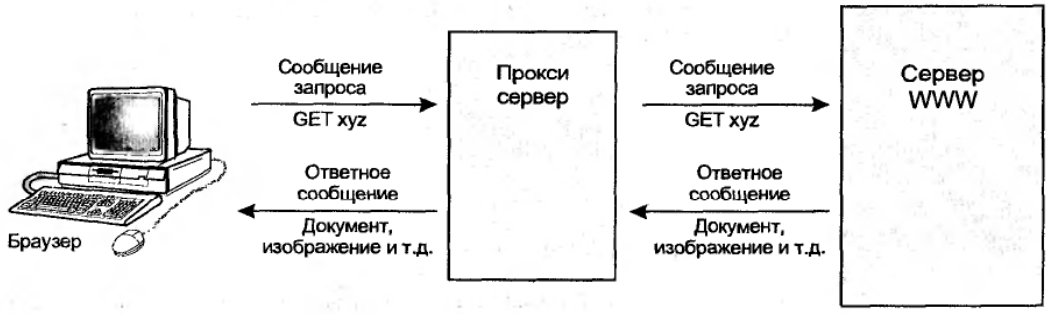

Рис. 19.4. Извлечение информации с сервера WWW через прокси

Некоторые прокси кешируют пересылаемые документы и могут самостоятельно отвечать на повторные запросы.

19.8 Протокол HTTP

Служба WWW реализуется поверх соединений TCP (хотя можно применять и другие транспорты) и разрастается вместе с Интернетом. Работа сервера WWW заключается в следующем:

- Клиент соединяется с сервером.
- Клиент посылает запрос, например:

```
GET /home.html HTTP/1.0
ACCEPT: text/html
```

- Сервер отвечает на запрос, указывая тип пересылаемой информации и передавая затребованный документ.

Сервер может взаимодействовать с различными видами клиентов благодаря подстройке отправляемых данных под возможности конкретного клиента. Клиент может объявлять о своих возможностях в операторе *Accept:*, отправляемом на сервер в запросе на извлечение документа. Один клиент может указать, что способен принимать только тексты в формате HTML, а другой — о своих возможностях по обработке текстов, изображений и звуковых файлов.

Обычно сервер WWW работает через общеизвестный порт TCP с номером 80. Иногда серверы конфигурируются для работы через другие порты.

В объектно-ориентированном языке (HTTP) вместо терминов "*команда*" или "*запрос*" используется термин "*метод*". Клиент может запрашивать три стандартных метода:

GET	Извлечение страницы документа
HEAD	Запрос на вывод заголовка запрашиваемого документа
POST	Отправка страницы на сервер, например ввод данных в форму

Метод GET извлекает *страницу*. Страница — это документ, содержащий любые изображения или звуковые файлы. Она может размещаться на одном листе или иметь размер целой книги.

Команда HEAD позволяет клиенту до начала пересылки определить длину и тип данных извлекаемого элемента, равно как и дату последнего изменения и текущую версию. Если браузер уже кешировал на локальном диске последнюю версию документа, то документ будет извлечен локально.

19.8.1 Пример типичного диалога HTTP

Один из доводов в пользу быстрого развития протокола WWW состоит в том, что разработчики не тратили время на повторное изобретение колеса, а заимствовали форматы заголовков и типов данных из классической электронной почты и стандартов MIME.

Представленный ниже диалог показывает, насколько просто выполняется взаимодействие в *HTTP*. Запрос GET/HTTP/1.0 требует извлечения с сервера документа по умолчанию и объявляет, что клиент работает по версии 1.0 протокола HTTP. Клиент также указывает, что способен принимать только текстовые документы HTML.

Ответ сервера объявляет об используемой версии HTTP (1.0) и коде статуса; 200 — означает успешное выполнение запроса. Далее следует серия подобных MIME заголовков. Пустая строка (<CR> <LF>) сообщает о конце раздела заголовков и начале тела документа.

```
GET/HTTP/1.0
ACCEPT: text/html

HTTP/1.0 200 Document follows
Date: Sat, 28 Oct 1995 14:07:25 GMT
Server: NCSA/1.5.1
Content-type: text/html
Last-modified: Tue, 09 May 1995 01:22:41 GMT
Content-length: 1563
```

```
<TITLE>InterNIC Directory and Database Services Home Page</TITLE>
<IMG src = "/Pics/logo.gif" alt = "">
<a href = ds/dspg01.html>
<H1>InterNIC Directory and Database Services</H1></a>
<P>
Welcome to InterNIC Directory and Database Services provided by AT&amp;amp;T.
These services are partially supported through a cooperative agreement with
the National Science Foundation.
. . .
```

Сервер закроет соединение, когда будет завершена пересылка.

19.8.2 Заголовки сообщения

В таблицах 19.2—19.5 представлены краткие описания заголовков в запросах и ответах.

Таблица 19.2

Главные заголовки HTTP

Главные заголовки	Описание
Date: *дата*	Дата в формате универсального времени, например: *Date: Sun, 29 Oct 1995 15:15:23 GMT*
MIME-Version: *версия*	Версия MIME заголовка, например: *MIME-Version: 1.0*
Pragma: *директива*	Реализация конкретной директивы, например: *Pragma: no-cache* (указывает прокси на извлечение более новой версии страницы, если данная страница уже кеширована).

Таблица 19.3

Заголовки запросов HTTP

Заголовки запросов	Описание
Authorization: *мандат*	Содержит информацию об аутентификации клиента для доступа к защищенным ресурсам.
From: *идентификатор электронной почты*	Подобен соответствующему полю в электронной почте.
If-Modified-Since: *дата*	Служит для организации условия в GET. Если затребованный документ не был изменен после указанной даты, в ответе будет содержаться только код 304, но не будет тела документа.
Referer: *URL*	Указание на источник получения ссылки на документ, например: *Referrer: http://www.abc.com/index.html.*
User-Agent: *программа*	Идентифицирует программное обеспечение клиента.

Таблица 19.4

Заголовки ответов HTTP

Заголовки ответов	Описание
Location: *URL*	Предпочитаемое сервером размещение документа.
Server: *программа*	Идентифицирует программное обеспечение сервера.
WWW-Authenticate: *исследование*	Предоставляет параметры для указания на схему аутентификации и необходимость аутентификации самого клиента.

Таблица 19.5

Заголовки элементов HTTP

Заголовки элементов	Описание
Allow: *метод*	Перечисление поддерживаемых ресурсом методов, например: *Allow: GET, HEAD*
Content-Encoding: *кодирование содержимого*	Для сжатого или зашифрованного содержимого; указывает на использованный алгоритм, например: *Content-Encoding: x-gzip*
Content-Length: *длина*	Описывает длину тела пересылаемого документа, например: *Content-Length: 2048*
Content-Type: *тип носителя*	Определены IANA, например: *Content-Type: text/html*
Expires: *дата*	Элемент недостоверен после указанной даты.
Last-Modified: *дата*	Время последней модификации элемента.

- В сообщении первыми стоят *главные* заголовки как в запросах, так и в ответах (таблица 19.2).
- Затем следуют заголовки, специфичные для запросов (таблица 19.3) или ответов (таблица 19.4).
- Наконец последними стоят заголовки *элементов*, которые обеспечивают детальное описание данного элемента (таблица 19.5).

Нужно помнить, что запрос POST приводит к пересылке от клиента к серверу определенных элементов, например данных формы. Поэтому заголовки элементов могут появляться в запросах и ответах.

19.8.3 Коды состояния

Коды состояния используются подобно электронной почте и пересылке файлов (FTP). Наиболее распространенные значения кодов:

1xx	Информация. Не используется, но зарезервирован для применения в будущем.
2xx	Успешно. Запрошенная операция была успешно получена, понята и принята для исполнения.
3xx	Перенаправление. Для полного завершения требуются дополнительные действия.
4xx	Ошибка клиента. Запрос имеет синтаксическую ошибку или не может быть выполнен.
5xx	Ошибка сервера. Сервер не смог выполнить корректный запрос.

Более детальные сведения обозначаются дополнительными кодами.

19.9 Продолжение совершенствования

В ответ на требования пользователей по обеспечению больших функциональных возможностей HTTP и HTML постоянно совершенствуются. На момент написания книги шла разработка стандартов для обеспечения безопасности взаимодействий клиент/сервер и для создания действительно защищенных коммерческих систем. Других достижений можно ожидать в области определения и реализации независимой от размещения *ресурсов схемы именования* (Uniform Resource Names — URN), поскольку существует проблема потери ссылки при перемещении документа на другой компьютер или в другой каталог.

URN делает доступным извлечение нужного документа из другого места сети. Можно указать несколько мест размещения документа с выбором оптимального варианта для извлечения.

19.10 Дополнительная литература

RFC 1738 содержит описание URL. RFC 1630 — это техническое руководство по Universal Resource Identifiers.

Спецификация HTTP 1.0 была опубликована в RFC 1945. Отдельные документы по HTML существуют в Интернете в форме проектов, к которым можно обратиться по *ftp://ftp.internic.net/internet-drafts*.

Информация о безопасности в HTTP, HTML и WWW доступна на сайте консорциума W3 (*http://www.w3.org/*).

Глава

20

SNMP

20.1 Введение

Сетевое управление далеко отстало по своим возможностям от других сетевых средств. Очень большие сети TCP/IP прекрасно работают, однако их администрирование и обслуживание требуют много времени и наличия квалифицированного технического персонала.

Особенно это справедливо для сети Интернет, которая быстро разрастается и усложняется. В конце 80-х гг. Совет по архитектуре Интернета (Internet Architecture Board — IAB) столкнулся с необходимостью определения технической политики Интернета, наиболее критичной частью которой являлось установление основ управления сетью и создание набора стандартов для соответствующих рабочих инструментов. Сделать это нужно было как можно быстрее.

Хотя большая часть работы уже была выполнена комитетом по созданию стандартов сетевого управления OSI, предстояло разработать реальные стандарты для инструментов управления сетями TCP/IP.

Рабочая группа Интернета создала простой протокол сетевого управления (Simple Network Management Protocol — SNMP), который решал сиюминутные потребности TCP/IP. Архитектура SNMP разрабатывалась в соответствии с моделью OSI. Была надежда, что стандарт сетевого управления OSI — общая информационная служба управления/общий протокол управляющей информации (Common Management Information Services/Common Management Information Protocol — CMIS/CMIP) — просуществует долго. Однако за несколько месяцев стало ясно, что SNMP требует независимой разработки и должен помочь в создании средств для управления сетями.

20.1.1 Результат одобрения SNMP в IAB

Первая спецификация SNMP стала начальной точкой. Эксперты из IAB быстро внесли необходимые изменения. Как указано в RFC 1052 (рекомендации по разработке стандартов сетевого управления для Интернета), служба сетевого управления должна:

(a) поддерживать сети максимально большого размера
(b) как можно полнее охватывать реализации
(c) работать поверх наибольшего количества протоколов различного уровня
(d) охватывать как можно больший круг задач администрирования

Результаты политики IAB превзошли все первоначальные ожидания. Как только была опубликована спецификация SNMP и в Интернете появились первые примеры исходных кодов, протокол был реализован в сотнях продуктов, начиная от сложных хостов на больших ЭВМ — до простейших коммуникационных устройств, и этот процесс расширялся и углублялся.

Разработчики получили возможность создавать сетевые станции управления с использованием хорошо известных протоколов для взаимодействия с широким диапазоном различных устройств. Расширяющийся рынок позволил создавать все более совершенные управляющие станции с графическим пользовательским интерфейсом, регистрацией изменений баз данных и возможностью генерации отчетов.

Продолжающаяся разработка RFC позволила охватить протоколом большое количество различных устройств.

В 1996 г. была опубликована вторая версия SNMP. Некоторые ее возможности рассматриваются в этой главе.

20.2 Модель SNMP

20.2.1 Логическая база данных

В SNMP используется модель базы данных. Каждая сетевая система содержит информацию о конфигурации, текущем состоянии, ошибках и производительности. К этой информации может получить доступ сетевой администратор. Она рассматривается как расположенная в *логической базе данных.*

20.2.2 Агенты

Для обеспечения доступа к информации управляемая система должна иметь программный *агент*, который отвечает на запросы, выполняет изменения и выводит отчет о возникших проблемах. Одна или несколько *станций управления* (management station) посылают запросы и сообщения об изменениях к агентам, получая от них ответы и сообщения об ошибках.

20.2.3 Диспетчеры

Как видно на рис. 20.1, станция управления имеет программный *диспетчер*, который посылает и получает сообщения SNMP. Кроме того, она может иметь различные *приложения для управления* сетью, взаимодействующие с устройствами сети через диспетчера.

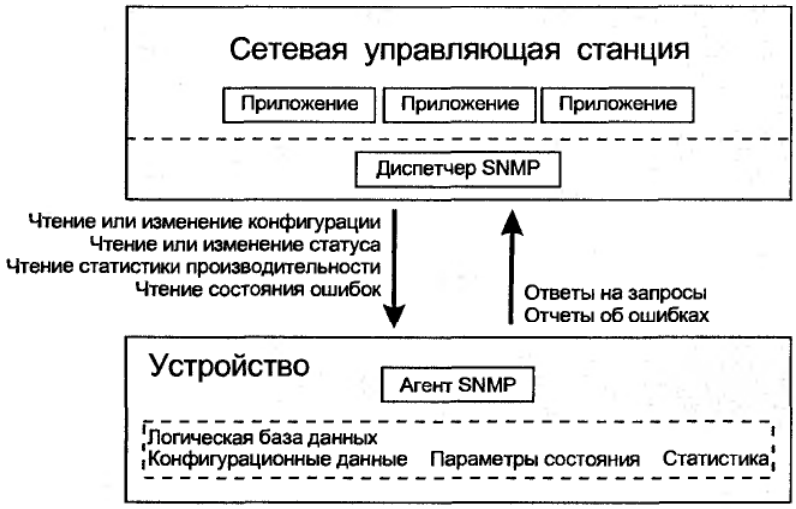

Рис. 20.1. Модель SNMP

20.2.4 Управляющая информационная база

Управляющая информационная база (Management Information Base — MIB) является логическим описанием всех управляющих данных. Существует много документов RFC, присваивающих набор соответствующих переменных. Каждый из этих документов описывает *модуль* MIB. Кроме того, имеются документы MIB, разработанные производителями оборудования и определяющие переменные, специфичные для данного типа продуктов.

Описание переменных MIB не связано со способом хранения этих переменных в устройстве, но определяет:

- Смысл переменной
- Способ измерения переменной
- Имя для доступа к значению переменной при чтении или изменении содержимого базы данных

Хотя MIB формально состоит только из набора определений, этого достаточно для запроса необходимых данных, хранящихся в *базе данных MIB* определенного устройства (иногда говорят: в *MIB устройства*). Типичная база данных MIB содержит:

- Информацию о системе и ее состоянии

- Статистику производительности
- Конфигурационные параметры

MIB устройства хранит только те переменные, которые необходимы для данного устройства. Например, простейшему мосту локальной сети нет смысла хранить переменные для оценки статистики TCP.

20.3 Назначение диспетчера и агента

Приложение для управления сетью обеспечивает оператору пользовательский интерфейс, реализующий функции обслуживания сети, просмотра состояния ее отдельных компонентов и анализ данных различных сетевых узлов.

Диспетчер осуществляет *общее руководство* сетью, запрашивая у агентов сетевых устройств значения переменных из их баз данных MIB. Типичным значением такой переменной может служить тип физических сетевых интерфейсов устройства и оценка напряженности трафика, проходящего через каждый интерфейс.

Диспетчер *управляет* системой, запрашивая у агента изменение состояния MIB или конфигурационных параметров из этой базы данных. Изменение параметра может быть связано с выполнением определенной операции. Например, можно отключить один из сетевых интерфейсов устройства, установив переменную статуса в состояние "*выключено*".

Новые возможности по управлению и обслуживанию сети реализуются через введение новых переменных в MIB.

Современные системы мониторинга позволяют отслеживать работу многих разнообразных устройств. Существуют программные мониторы для работы на различных платформах: от больших ЭВМ до персональных компьютеров. В следующих разделах этой главы будут приведены примеры экранов систем мониторинга *HP Open View for Windows Workgroup Node Manager*, работающего в среде Windows.

20.3.1 Прокси-агенты

В основе модели SNMP лежит принцип сосуществования агента и базы данных MIB в одном устройстве, которое может управляться и контролироваться из удаленной точки сети. *Прокси-агент* несколько расширяет эту модель, разрешая косвенный доступ к устройству. Станция управления будет взаимодействовать с прокси, когда станет необходим доступ или изменение информации. Прокси-агент обменивается информацией с устройством с помощью отдельного соединения (см. рис. 20.2).

В версии 2 SNMP прокси служат для пересылки информации между окружениями версий 1 и 2.

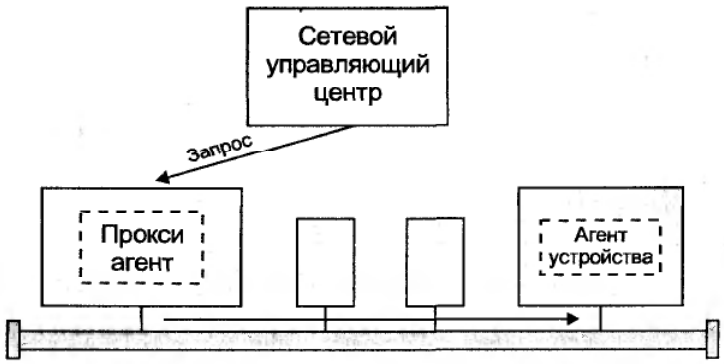

Рис. 20.2. Прокси-агент

20.4 Сущность управляющей информации

Описание управляющих *переменных* полностью независимо от спецификации *протокола* для обмена между программным монитором и агентом. Это наиболее важное свойство сетевой архитектуры.

Описание переменных возложено на комиссии экспертов по каждой из сетевых технологий. Отдельные комиссии разрабатывают MIB для мостов, хостов, телефонных интерфейсов и т.д.

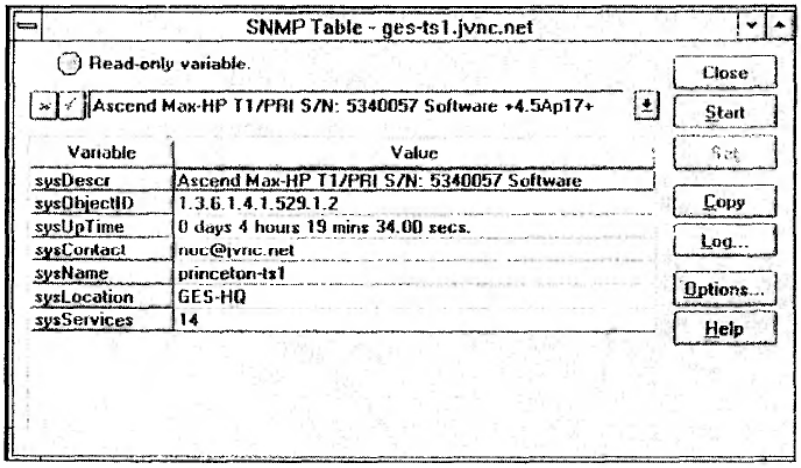

Рис. 20.3. Извлеченная из устройства системная информация

Главным документом MIB является спецификация управления в сетях TCP/IP, которая содержит следующие сведения:

- Тип данной системы
- Имя и физическое размещение системы
- Типы сетевых интерфейсов системы
- Число полученных и отправленных кадров, сегментов или датаграмм TCP.

На рис. 20.3 показана системная информация маршрутизатора, извлеченная в HP *Openview*.

20.5 Структура управляющей информации

Для описания переменных сетевого управления необходимы:

- *Административная структура.* Работа по описанию переменных MIB для различных типов сетевых устройств возложена на специалистов в данной области. Административная структура необходима для описания и отслеживания разделения работ и делегирования полномочий на их проведение.
- *Информационная структура.* Сетевая информация не остается статичной, следовательно, она должна быть структурирована для упрощения расширения или изменения старых, либо ввода новых технологий.
- *Структура именования.* Существуют сотни переменных, описывающих управление в сети. Необходим единый метод описания, определения и именования этих переменных.

Всем этим требованиям удовлетворяет древовидная структура, называемая *структурой управляющей информации* (Structure of Management Information — SMI).

20.5.1 Дерево SMI

Вспомним, что первоначально SNMP предполагался как временное решение до выпуска стандартов управления ISO. На рис. 20.4 дерево администрирования/именования отражает первичные попытки согласования с ISO.

Узлы вверху этого дерева предполагают ответственность определенных организаций за разработку требований для нижестоящих узлов (см. таблицу 20.1). Однако многое в этом дереве уже устарело. Стандарты SNMP уже давно не координируются ISO (в дереве — iso), а Министерство обороны США не управляет работой Интернета (в дереве — dod).

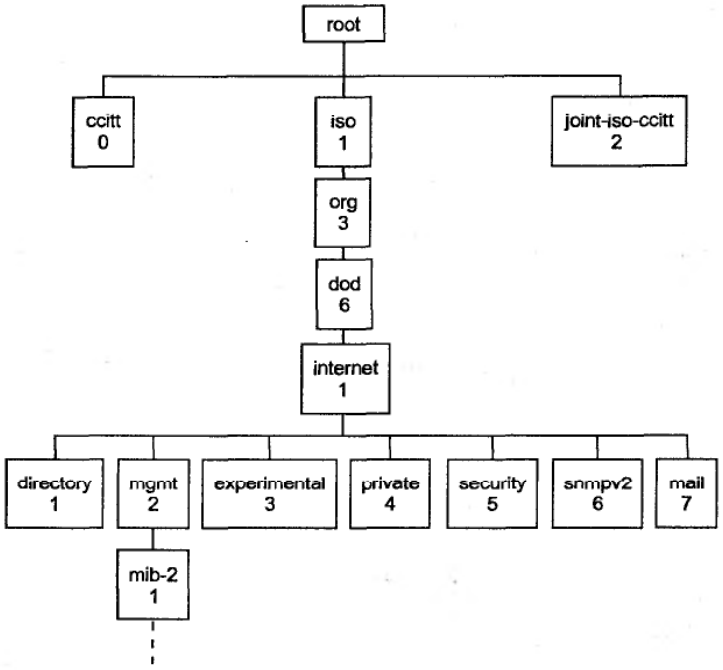

Рис. 20.4. Дерево администрирования и именования SMI

Таблица 20.1

Узлы дерева SMI

Метка	Описание
iso(1)	Международная организация по стандартизации (ISO)
org(3)	Национальные и международные организации
dod(6)	Министерство обороны США (DOD)
internet(1)	IAB

Однако дерево продолжает выполнять свою основную функцию — определение имен MIB. Древовидная структура очень полезна: как только в сетевом окружении возникает новая технология, создается специальный комитет, а на дереве появляется новый узел. Комитет ведет разработку переменных и добавляет их к общему дереву в виде поддерева.

20.6 Имена идентификаторов объектов

На рис. 20.5 показаны наиболее важные части дерева SMI, которые применяются для присвоения управляющим переменным имен, называемых *идентификаторами объектов* (object identifiers).

Идентификаторы объектов создаются путем приписывания числовых идентификаторов каждому узлу, начиная с вершины дерева. Каждый узел имеет и текстовую метку, помогающую пользователям и разработчикам понять назначение переменной. Например:

Идентификатор объекта	**1.3.6.1.2.1.1.1**
Текстовое имя	**iso.org.dod.internet.mgmt.system.sysDescr**

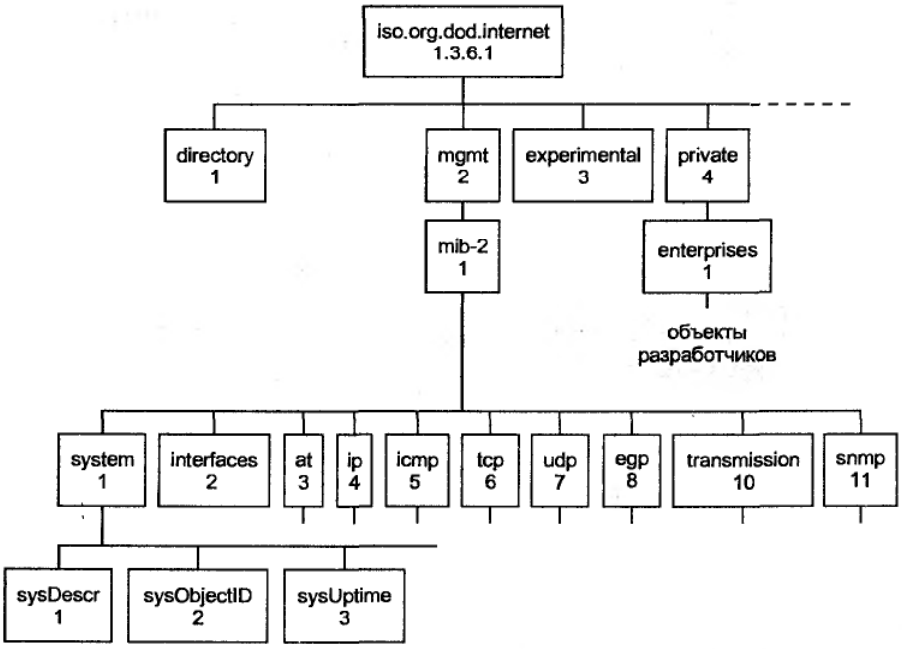

Рис. 20.5. *Дерево именования объектов MIB*

20.6.1 Идентификация значений в базе данных MIB

Для описания реального значения в базе данных устройства в конец идентификатора объекта добавляется еще одно число. Например, если информация обо всех интерфейсах устройства хранится в таблице, а идентификатор объекта для таблицы *ifType* имеет значение 1.3.6.1.2.1.2.2.1.3, то для указания на четвертый интерфейс данного маршрутизатора нужно использовать идентификатор:

1.3.6.1.2.1.2.2.1.3.*4*

Соглашение по добавлению индексов применяется и для переменных с единственным значением, например *sysDescr* или *sysUpTime*. В этом случае в конец идентификатора добавляется 0. Например, полное описание переменной *sysDescr*:

1.3.6.1.2.1.1.1.*0*

20.6.2 Лексикографический порядок

Переменные в MIB упорядочены лексикографически. Для сравнения двух идентификаторов:

1.3.6.1.2.1.2.2.1.**19**.3

1.3.6.1.2.1.2.2.1.**21**.2

нужно выполнить:

- Начать слева.
- Сравнивать значения, пока не будет найдено первое отличие.
- Число с большим значением определяет больший элемент.

В приведенном примере второй идентификатор больше первого. Однако что делать в следующем случае:

1.3.6.1.2.1.2.2.1

1.3.6.1.2.1.2.2.1.21.2

Большим будет более длинный идентификатор.

Таким образом, для просмотра таблицы в лексикографическом порядке нужно сначала пройти по столбцу сверху вниз, а затем перейти в верхнюю часть следующего столбца (см. рис. 20.6).

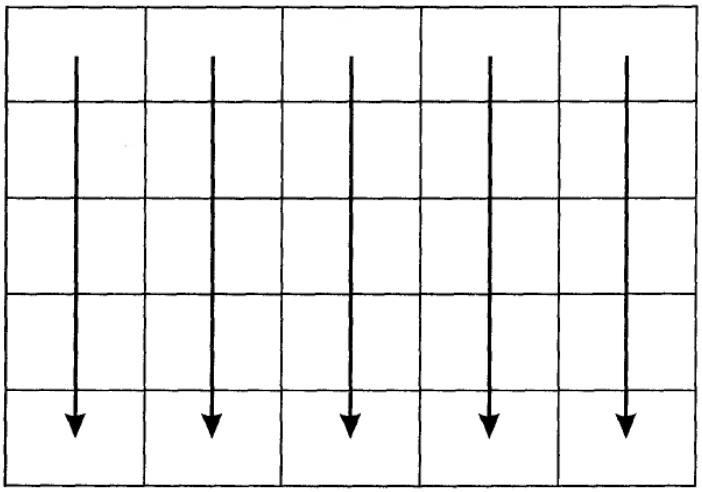

Рис. 20.6. Лексикографический порядок для таблиц

20.7 Наиболее важные модули MIB

Разработаны десятки модулей MIB, описывающих все: от интерфейса RS-232 до серверов электронной почты. В этом разделе мы рассмотрим наиболее важные модули MIB.

20.7.1 MIB-II

Данная группа переменных изображена на рис. 20.5 (*system*, *interfaces* и т.д.). Все эти переменные были определены в первом документе MIB, описывающем переменные сетей TCP/IP. После проверки и исследования модуль был переработан и получил название *MIB-II*. В последней версии модуля описываются определения переменных для SNMP версии 1. Затем были опубликованы небольшие изменения, связанные с версией 2.

20.7.2 Модули пересылки

Создано множество модулей, описывающих технологии локальных и региональных сетей. Несколько поддеревьев создано от узла *transmission* (см. рис. 20.7). Полный их список приведен в документе *Assigned Numbers*.

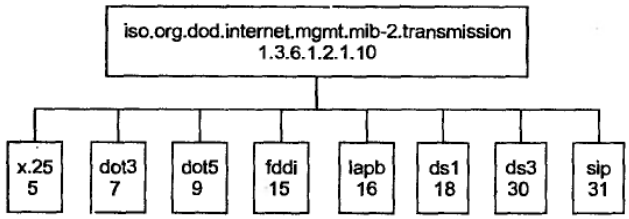

Рис. 20.7. Модули пересылки MIB

20.7.3 RMON MIB

Сетевой *монитор* (*зонд*) — это устройство, пассивно наблюдающее за трафиком связи. Оно может быть сконфигурировано для сбора данных об этом трафике с использованием шаблонов и отображением статистики о производительности сети. Мониторы конфигурируются на определенный уровень ошибок, что позволяет узнать о проблемах до того, как они станут критическими.

Удаленный мониторинг сети MIB (Remote Network Monitoring MIB — RMON MIB) интегрирует ценную информацию, собранную мониторами из структур SNMP. Это дает существенное расширение возможностей станций управления SNMP.

Удаленный монитор может самостоятельно собирать локальные данные, проводить диагностику оборудования и обнаруживать опасные состояния. Так как о проблемах становится известно при их возникновении, сетевые станции управления могут регулировать частоту запросов данных MIB от отдельных устройств. Для RMON MIB определены девять групп данных (см. таблицу 20.2).

Таблица 20.2

Группы переменных RMON MIB

Переменная	Описание
statistics	Статистика работы определенного типа интерфейса, например для Ethernet — коллизии или ошибки, а для Token-Ring — сигналы ошибок или потерянные маркеры.
history	Статистические оценки за указанный временной интервал выборки значений.
alarm	Генерация события при превышении переменной заданного граничного значения.
host	Отчет хоста монитору, содержащий сопутствующую статистическую информацию, например число переданных кадров.
hostTopN	Статистический отчет хоста, содержащий список отсортированных значений о производительности или количестве ошибок.
matrix	Статистический отчет об обмене между двумя сетевыми адресами.
filter	Определение критерия для выделения набора кадров с целью более подробного анализа.
packet capture	Позволяет регистрировать кадры, соответствующие установленному критерию.
event	Управление генерацией и выводом сведений о событиях. Событие может иметь локальную природу, например превышение граничного значения переменной. Оно позволяет переключиться на выполнение локальной операции, например на запись сообщения в файл регистрации, инициализацию захвата пакетов или на вывод сообщения *trap* на станцию управления.

20.7.4 Реализация MIB от разработчиков оборудования

С самого начала на дереве объектов MIB было отведено место для объектов от разработчиков. Для получения ветви дерева разработчик (компания, организация или правительственное агентство) регистрируется в IANA. На рис. 20.8 показана часть дерева MIB компании Cabletron, которой был присвоен идентификатор объекта:

1 . 3 . 6 . 1 . 4 . 1 . 52.

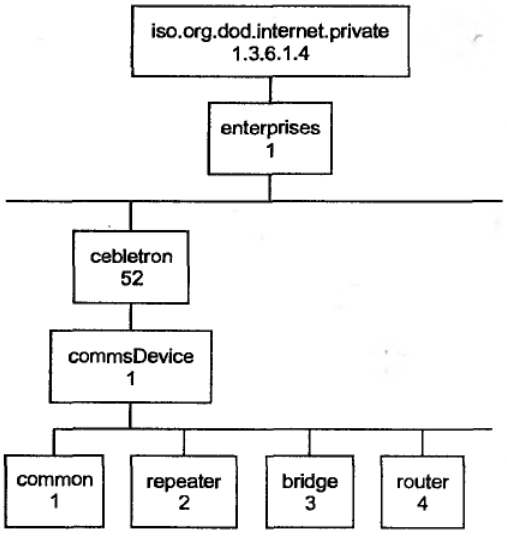

Рис. 20.8. Часть дерева MIB компании Cabletron

20.8 Протокол сообщений SNMP

Рассмотрим протокол сообщений, обеспечивающий взаимодействие диспетчеров с агентами. SNMP основан на двух принципах:

- Выбирается очень нетребовательный транспортный протокол для пересылки данных, но допустимо использовать SNMP и в сетях, не имеющих протокола TCP/IP.
- Используется очень мало типов сообщений.

20.8.1 Типы сообщений SNMP версии 1

Диспетчеры и агенты взаимодействуют друг с другом, обмениваясь сообщениями SNMP. Как показано на рис. 20.9, для версии 1 протокола SNMP существует только пять типов сообщений:

get-request	Запрос значений одной или нескольких переменных из системы управления MIB
get-next-request	Разрешение диспетчеру на последовательное извлечение значений переменных (используется для просмотра таблиц или всей базы данных MIB)
set-request	Разрешение диспетчеру изменить значения переменных
get-response	Получить ответ на сообщения *get*, *get-next* и *set* (в версии 2 называется *response*)
trap	Разрешение агенту на отчет о важных событиях или проблемах

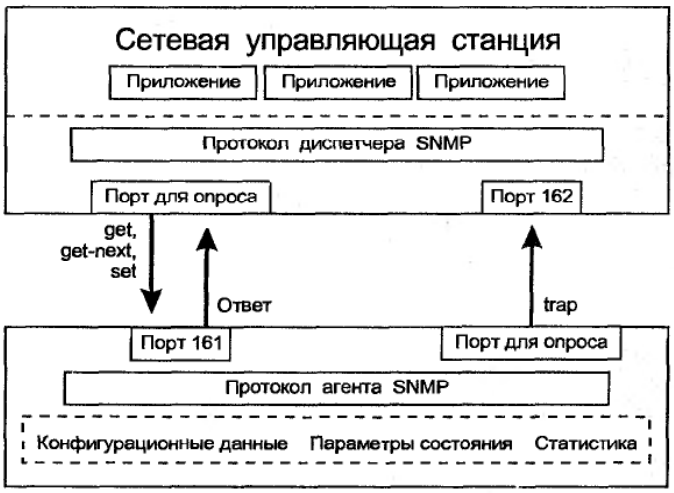

Рис. 20.9. Типы сообщений SNMP версии 1

Ограничение обмена пятью типами сообщений сохранило простоту реализации и при этом обеспечило множество функциональных возможностей.

Обычно сетевые администраторы конфигурируют станцию управления на чтение статистики через регулярные интервалы, например через каждые 15 мин. Полученные значения могут быть сохранены и проанализированы, чтобы обнаружить пиковые нагрузки и определить необычные состояния.

Сообщение *trap* используется для отчета о наиболее общих событиях:

- Самостоятельная переинициализация
- Локальный отказ связи
- Восстановление связи

Комитеты стандартов MIB определили дополнительные сообщения *trap* для данных коммуникационных технологий. Кроме того, разработчики определяют *trap* для вывода информации о критических проблемах, связанных с работой их оборудования.

Частью концепции SNMP является то, что число пересылаемых сообщений *trap* должно быть относительно невелико. Сетевые администраторы часто сталкиваются с ситуацией, когда одна проблема приводит к появлению других. Наводнение сети потоком сообщений о проблемах может препятствовать выполнению операций по восстановлению нормальной работы.

20.8.2 Транспортные протоколы

В качестве наиболее предпочтительного транспорта был выбран протокол UDP. Это объясняется его простотой и реализацией с помощью очень небольшого кода. Такой выбор особенно подходит для работы устройств в режиме перегрузки или при неисправности. Однако для SNMP могут использоваться и другие транспортные протоколы. Например, в окружении NetWare протокол SNMP может работать поверх IPX.

Когда используется UDP, каждое сообщение SNMP вкладывается в одну датаграмму UDP и доставляется через IP. Как показано на рис. 20.9, запросы направляются на порт 161 от любого удобного порта UDP. *Ответы* возвращаются на запрашивающий порт. Сообщения *trap* исходят из любого удобного порта UDP и направляются на порт 162.

Все реализации версии 1 должны быть способны обрабатывать сообщения длиной, по крайней мере, в 484 октета.

20.9 Форматы сообщений SNMP

Сообщение SNMP версии 1 состоит из некоторого вводного материала — "обертки",— сопровождаемого сообщением Protocol Data Unit одного из пяти типов: *get-request, get-next-request, get-response, set-request* или *trap*. Вводный материал содержит:

Версию протокола	0 для SNMP версии 1 и 1 для версии 2
Имя сообщества	используется как пароль

Агент конфигурируется на ограничение (по имени сообщества) доступа к информации по чтению или записи. Кроме того, можно указать IP-адрес станции управления, которой разрешен доступ по чтению или записи информации MIB.

К сожалению, имя сообщества в сообщении можно легко подглядеть с помощью любого сетевого анализатора, а IP-адрес иногда можно сфальсифицировать. Одним из решений является доступ к важным устройствам (например, маршрутизаторам) через отдельную, безопасную линию связи, особенно при изменении конфигурации или статуса системы.

20.9.1 Формат сообщений gets, sets и responses в версии 1

Главное информационное содержимое во всех этих сообщениях одинаково. Оно состоит из списка (пары этого списка обычно называют "*связыванием переменной*"):

Имя переменной	Значение
Имя переменной	Значение
...	...

Идентификатор объекта используются как имя переменной. В сообщениях *get* и *get-next* поля значений пустые. В них агент разместит необходимые значения.

Элемент данных протокола (Protocol Data Unit) для сообщений *get-request, get-next-request, set-request* или *response* включает:

Идентификатор запроса	Служит для согласования запроса и ответа на него.
Поле статуса ошибки	0 в запросах. Ненулевые значения в ответах означают различные ошибки.
Поле индекса ошибки	0 в запросах. В ответах указывает переменную, создавшую ошибку.
Список идентификаторов объектов и значений	В *get* или *get-next* — пустые, но заполнены в *set* или *response*.

20.9.2 Запрос get и ответ на него

На рис. 20.10 показаны запрос *get-request* и *ответ* на него (response), полученные в анализаторе *Sniffer* компании Network General. Запрос содержит список из пяти переменных, значения которых нужно получить. После каждого идентификатора переменной стоит заполнитель NULL. Чтобы создать ответ, агент должен только заполнять пробелы и заменять пустые поля фактическими значениями.

```
SNMP: Version = 0
SNMP: Community = public
SNMP: Command = Get request
SNMP: Request ID = 112
SNMP: Error status = 0 (No error)
SNMP: Error index = 0
SNMP:
SNMP: Object = {1.3.6.1.2.1.1.3.0} (sysUpTime.0)
SNMP: Value = NULL
SNMP:
SNMP: Object = {1.3.6.1.2.1.5.1.0} (icmpInMsgs.0)
SNMP: Value = NULL
SNMP:
SNMP: Object = {1.3.6.1.2.1.5.2.0} (icmpInErrors.0)
SNMP: Value = NULL
SNMP:
SNMP: Object = {1.3.6.1.2.1.5.3.0} (icmpInDestUnreachs.0)
SMMP: Value = NULL
SNMP:
SNMP: Object = {1.3.6.1.2.1.5.4.0} (icmpInTimeExcds.0)
SNMP: Value = NULL
```

```
SNMP: Version = 0
SNMP: Community = public
SNMP: Command = Get response
SNMP: Request ID = 112
SNMP: Error status = 0 (No error)
SNMP: Error index = 0
SNMP:
SNMP: Object = {1.3.6.1.2.1.1.3.0} (sysUpTime.0)
SNMP: Value = 1037388 hundredths of a second
SNMP:
SNMP: Object = {1.3.6.1.2.1.5.1.0} (icmpInMsgs.0)
SNMP: Value = 1 messages
SNMP:
SNMP: Object = {1.3.6.1.2.1.5.2.0} (icmpInErrors.0)
SNMP: Value = 0 messages
SNMP:
SNMP: Object = {1.3.6.1.2.1.5.3.0} (icmpInDestUnreachs.0)
SNMP: Value = 0 messages
SNMP:
SNMP: Object = {1.3.6.1.2.1.5.4.0} (icmpInTimeExcds.0)
SNMP: Value = 0 messages
```

Рис. 20. 1. Пример *get-request* и *response*

20.9.3 Запрос get-next и ответ на него

Сообщение *get-next* работает по-другому. Когда отсылается идентификатор объекта, возвращается значение *следующего* объекта. Например, если послать запрос:

```
SNMP: Object = {1.3.6.1.2.1.5.1.0} (icmpInMsgs.0)
SNMP: Value = NULL
```

ответ будет содержать имя и значение для следующей переменной:

```
SNMP: Object = {1.3.6.1.2.1.5.2.0} (icmpInErrors.0)
SNMP: Value = 0 messages
```

Такой запрос позволяет просматривать значения MIB или перемещаться на следующую строку таблицы.

20.9.4 Запрос set

Запрос *set* позволяет записывать информацию в базу данных агента. Формат сообщения очень прост, он выглядит как *get-request*, но приводит к изменению указанных в запросе переменных. На рис. 20.11 показано отслеживание запроса *set*.

```
SNMP: Version = 0
SNMP: Community = xyz
SNMP: Command = Set request
SNMP: Request ID = 0
SNMP: Error status = 0 (No error)
SNMP: Error index = 0
SNMP:
SNMP: Object = {1.3.6.1.2.1.4.1.0} {ipForwarding.0}
SNMP: Value = 2
SNMP: Object = {1.3.6.1.2.1.4.2.0} {ipDefaultTTL.0}
SNMP: Value = 70
```

Рис. 20.11. Изменение значений MIB запросом *set*

Успешными должны быть все изменения запроса, иначе будет отклонен весь запрос. Поскольку часто нужно изменять одновременно несколько переменных, либо ни одну из них. Это бескомпромиссное правило для *set* сохраняется и в версии 2.

Ответ на *set* выглядит как запрос, за исключением того, что при возникновении проблем заполняются поля статуса ошибки и индекса ошибки.

20.9.5 Сообщения trap

Агент использует сообщения *trap* для указания диспетчеру на серьезные проблемы.

В стандарте SNMP определено очень немного таких сообщений. Описание *trap* оставлено в ведении комитетов по технологическим стандартам и разработчиков — с предупреждением о снижении количества таких сообщений. Когда сеть перегружена, нежелательно получать десятки сообщений от каждого из сетевых устройств с указаниями на их проблемы.

Сообщения *trap* в версии 1 были более сложными, чем следовало бы. Такое положение было исправлено в версии 2. Рассмотрим сначала сообщения *trap* версии 1. В них имеется *общее поле* (generic trap), значение которого определяет тип прерывания в соответствии со следующим списком:

coldStart (*0*)	Отправитель проводит переинициализацию, и его конфигурационные параметры могут измениться.
warmStart (*1*)	Отправитель проводит переинициализацию, и его конфигурационные параметры не будут изменяться.

linkDown (2)	Смежная связь нарушена.
linkUp (3)	Смежная связь восстановлена.
autentication Failure (4)	Кто-то послал агенту запрос, который не был аутентифицирован (например, в сообщении было использовано неправильное имя сообщества).
egpNeighbor Loss (5)	Сосед по протоколу Exterior Gateway Protocol выключен.
enterprise Specific (6)	Другие запросы, определенные комитетом стандартов, разработчиком или иным заинтересованным лицом.

На рис. 20.12 показано очень простое сообщение *trap*, указывающее на выполнение холодного старта (перезапуска с выключением питания.— *Прим. пер.*).

- Поле *Enterprise* указывает, что это сообщение отправлено системой, выполняющей программный продукт FTP для TCP/IP.
- Поскольку значение *общего поля trap* равно 0, это сообщение свидетельствует о холодном старте.
- Поле *счетчика времени* (time ticks) содержит *sysUpTime*, которое равно 0, поскольку система только что выполнила инициализацию по холодному старту.

```
SNMP: Version = 0
SNMP: Comunity = public
SNMP: Command = Trap
SNMP: Enterprise = {1.3.6.1.4.1.121.1.1}
SNMP: Network address = [198.207.177.10]
SNMP: Generic trap = 0 (Cold start)
SNMP: Specific trap = 0
SNMP: Time ticks = 0
```

Рис. 20.12. Сообщение *trap* версии 1 протокола SNMP

Любые сообщения trap, определенные комитетом MIB или разработчиком, имеют в *общем поле* значение 6. В данном случае поле *enterprise* комбинируется с полем *specific trap* (специальное прерывание), определяющим смысл сообщения.

Как видим, структура сообщения достаточно сложна. В версии 2 она была проще.

20.9.6 Проблемы версии 1, исправленные в версии 2

Следующие свойства SNMP версии 1 были не слишком удачны:

- Если одна из переменных в запросе *get* или *get-next* была некорректна, то отбрасывалось все сообщение.
- Если *запрашивались* значения нескольких переменных и агент не мог разместить ответ в самом большом по размеру сообщении, отбрасывалось все сообщение.
- Сообщения *trap* реализовывали простые функции, но имели сложную структуру.

Все эти проблемы были решены в версии 2. Теперь агент может помещать код ошибки в поле *значения переменной*, которая не может быть обработана. Появился новый запрос *get-bulk*, требующий от агента возврата максимально возможного объема информации. Сообщения *trap* стали иметь такой же простой формат, как и все другие сообщения.

В версии 2 расширен список поддерживаемых кодов ошибок, что позволяет диспетчерам лучше проанализировать причину неисправности, когда отклоняется запрос.

20.9.7 Сообщение get-bulk версии 2

Сообщение *get-bulk* ведет себя подобно *get-next*. Агент возвратит переменные, чьи идентификаторы объектов следуют за идентификаторами объектов, указанными в запросе.

Сообщение *get-bulk* имеет параметры, указывающие:

- Количество начальных автономных неповторяющихся (nonrepeater) запрошенных переменных
- Количество требуемых повторений для оставшихся повторяющихся (repeater) переменных

Например, можно запросить две неповторяющиеся автономные переменные:

sysDescr

sysUpTime

и затем еще десять строк табличных переменных: *ifIndex*, *ifDescr*, *ifType*, *ifMTU* и *ifSpeed*. В этом случае:

- В списке будет 7 переменных
- 2 неповторяющиеся переменные
- Максимальное число повторений будет равно 10

В ответе будет упаковано столько затребованной информации, сколько возможно. Приложение легко сможет послать еще один запрос *get-bulk* за данными, не поместившимися в сообщении ответа.

Так как поля статуса ошибки и индекса ошибки не используются в запросах, они задействованы в запросе *get-bulk* для хранения *неповторяющихся* параметров и *максимального значения повторений*. Это означает, что базовый формат не изменился при реализации нового сообщения *get-bulk*.

20.9.8 Сообщение trap в версии 2

В версии 2 сообщение *trap* имеет тот же самый формат, что и ответ на него. Сообщение начинается стандартной информацией заголовка, далее следует список переменных:

```
Идентификатор объекта      Значение
        ...                   ...
        ...                   ...
```

В начале списка переменных размещается *SysUpTime* и уникальный идентификатор *trap*. Могут быть включены дополнительные переменные, помогающие определить причину возникновения проблемы.

20.9.9 Сообщение inform версии 2

В версии 2 реализована идея *информационного* сообщения, подтверждающего получение *trap*. Такие сообщения полезны при взаимодействии диспетчеров, когда отправителю нужно точно знать о получении сообщения в принимающем диспетчере. Для подтверждения приема информационного сообщения (inform) используется обычный ответ.

20.9.10 Другие усовершенствования в версии 2

Насколько точно реализация модуля должна соответствовать определению MIB от разработчика для обеспечения требований совместимости? И как разработчик может объявить о несоответствии спецификации, которое, скорее всего, было необходимо из-за некоторых ограничений в возможностях оборудования?

Решить эти вопросы в версии 2 помогают следующие средства:

- Описание совместимости (compliance statement), определяющее фактические минимальные требования для модуля
- Описание возможностей (capability statement), предоставляемое разработчиком для пояснения реальных возможностей агента

Эти описания позволяют клиенту при выборе узнать о продукте немного больше, чем "мы поддерживаем SNMP".

20.10 Документы MIB

Документы, определяющие переменные MIB, содержат полезную информацию. Они точно описывают, как каждая переменная определена и измеряется. Имеется и дополнительный материал, описывающий технологию, условия возникновения ошибок и типичные конфигурации.

В следующих разделах мы обсудим некоторые концепции, знание которых будет полезно при чтении документов MIB.

20.10.1 Управляемые объекты

До сих пор мы использовали неформальный термин "*переменная MIB*". Но стандарты MIB реально определяют *управляемые объекты* (managed objects). Переменная имеет название и значение, а определение управляемого объекта включает:

- Имя — идентификатор объекта
- Набор атрибутов, в частности:
  1. Тип данных
  2. Описание деталей реализации
  3. Информацию о статусе
- Набор операций, которые могут быть выполнены над объектом

Рассмотрим типичное определение MIB:

```
sysDescr OBJECT-TYPE
  SYNTAX DisplayString (SIZE (0. .255))
  ACCESS read-only
  STATUS mandatory
  DESCRIPTION
      "Текстовое описание элемента, которое должно содержать полное имя и
      номер версии, типа аппаратного обеспечения системы, операционной
      системы и сетевых средств. Подтверждается (mandatory), что вся
      информация содержит только воспроизводимые символы ASCII."
  :: = { system 1 }
```

Определение начинается с обозначения текстовой метки объекта — *sysDescr* — и заканчивается { *system 1* }, что означает "поместить этот узел ниже узла *system* и присвоить ему номер 1". Такая запись позволяет построить полный идентификатор объекта, который будет выглядеть как:

1.3.6.1.2.1.1.1

Остальная часть определения состоит из ряда *конструкций* (clauses) — SYNTAX (синтаксис), ACCESS (доступ), STATUS (статус) и DESCRIPTION (описание).

В данном случае *SYNTAX* (datatype) — это выводимая строка, т.е. ряд символов не длиннее 255 знаков.

ACCESS определяет действие(я), которое может быть выполнено. В данном случае ACCESS задан как "чтение/запись", а диспетчер может читать или изменять значения переменных.

В ранних документах MIB условие *STATUS* могло иметь значения: *mandatory* (обязательно), *optional* (необязательно), *obsolete* (устарело) или *deprecated* (отменено). Однако значения mandatory и optional были бесполезны. Более новые MIB не включают переменных, значение которых столь мало, что нет смысла помечать их специальным значением. STATUS теперь может указать на *current* (текущее значение), *deprecated* (отменено) или *obsolete* (устарело).

20.10.2 Первая абстрактная синтаксическая нотация (ASN.1)

Определения MIB написаны на стандартном языке *первой абстрактной синтаксической нотации* (Abstract Syntax Notation 1 — ASN.1), разработанном в ISO. ASN.1 похож на компьютерные языки. Существуют и *основные правила кодирования* (Basic Encoding Rules — BER), также от ISO, определяющие формат пересылки значений, определенных с помощью ASN.1.

Станция управления анализирует переменные MIB, *компилируя* определения MIB в записи ASN.1. Хорошие станции управления позволяют компилировать столько MIB, сколько нужно.

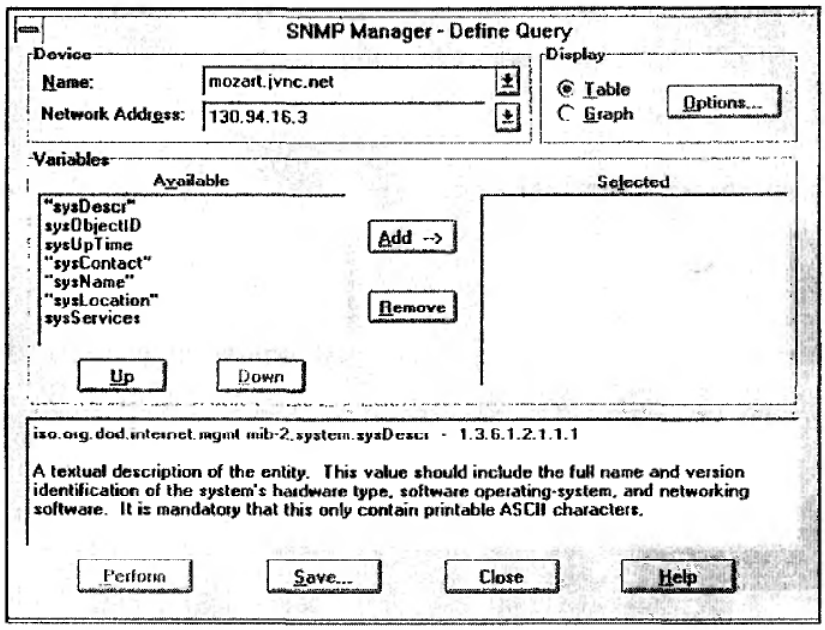

Рис. 20.13. Вывод описаний переменных на экране диспетчера SNMP

После компиляции станция управления готова посылать и получить сообщения SNMP, содержащие любую из скомпилированных переменных. Хорошие станции могут также выводить описания переменных. На рис. 20.13 показан вывод в HP *OpenView* условия DESCRIPTION описания *sysDescr*.

20.10.3 Типы данных MIB

Причиной широкого распространения SNMP стало то, что проектировщики придерживались правила "Будь проще!"

- Все данные MIB состоят из простых скалярных переменных, хотя отдельные части MIB могут быть *логически* организованы в таблицы.
- Только небольшое число типов данных (например, целые числа или строки октетов) используется выражения значений всех переменных MIB.

Фактически основные типы данных — это INTEGER (целое), OCTET STRING (строка октетов) и OBJECT IDENTIFIER (идентификатор объекта).

20.10.4 Целые числа

Целые числа используются в двух случаях:

- Для ответа на вопрос "сколько?"
- Для перечисления списка вариантов, например 1 = включено, 2 = выключено, 3 = тестирование.

Ниже приведено определение, иллюстрирующее использование различных типов данных. Заметьте, что в первом определении формулировка SYNTAX ограничивает амплитуду значений.

```
tcpConnLocalPort OBJECT-TYPE
  SYNTAX INTEGER (0..65535)
  ACCESS read-only
  STATUS mandatory
```

```
    DESCRIPTION
        "Номер локального порта для данного
         соединения TCP."
    :: = { tcpConnEntry 3 }

ifAdminStatus OBJECT-TYPE
    SYNTAX INTEGER {
        up ( 1), - готов к пересылке пакета
        down ( 2 ) ,
        testing (3) - режим тестирования
            }
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        "Требуемое состояние интерфейса. Тестирование (3) указывает
         на отмену пересылки пакетов."
    :: = { ifEntry 7 }
```

20.10.5 Счетчики

Счетчик — это положительное целое число, которое увеличивается до максимального значения и затем сбрасывается в ноль. Известно, что 32-разрядный счетчик может увеличиваться до $2^{32} - 1$ (4 294 967 295) и затем сбрасывается в 0. В версии 2 добавлен 64-разрядный счетчик, который может увеличиваться до 18 446 744 073 709 551 615.

Значение счетчика само по себе не используется. Регистрируется текущее значение счетчика, а затем сравнивается с его предыдущим значением. Смысл имеет *разность* этих значений. Пример переменной со счетчиком:

```
ifInOctets OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Общее количество полученных интерфейсом октетов,
         включая символы обрамления кадров."
    :: = { ifEntry 10 }
```

20.10.6 Масштаб

Масштаб (gauge) — это целое число, которое ведет себя по-разному. Значения масштаба увеличиваются и уменьшаются. Масштабы используются для количественного описания, например длины очереди. Иногда значение масштаба растет, а иногда уменьшается.

32-разрядный масштаб может увеличиваться до $2^{32} - 1$ (4 294 967 295). Если измеряемая величина превышает масштаб, то она фиксируется в этом максимуме, пока значение снова не уменьшится (см. рис. 20.14).



Рис. 20.14. Поведение значения масштаба

Пример переменной масштаба:

```
ifOutQLen OBJECT-TYPE
  SYNTAX Gauge
  ACCESS read-only
  STATUS mandatory
  DESCRIPTION
      "Длина выходной очереди пакетов
       (в пакетах)."
  :: = { ifEntry 21 }
```

20.10.7 TimeTicks

Интервалы времени измеряются в *TimeTicks*, размер которого выражается в сотых долях секунды. Значение TimeTick — неотрицательное целое число в пределах от 1 до $2^{32} - 1$. Для переполнения счетчика TimeTick потребуется 497 дней.

SysUptime, измеряющая время от инициализации программного обеспечения агента,— это наиболее часто используемая переменная TimeTick.

```
sysUpTime OBJECT-TYPE
  SYNTAX TimeTicks
  ACCESS read-only
  STATUS mandatory
  DESCRIPTION
      "Время (в сотых долях секунды) от последней инициализации
       части системы для сетевого управления."
  :: = { system 3 }
```

20.10.8 Строки октетов

OCTET STRING (строки октетов) — это последовательность байт. Почти любые данные можно представить строкой октетов.

20.10.9 Текстовые соглашения

Вместо определения новых типов данных в определении MIB применяются *текстовые соглашения* (textual conventions), позволяющие указать, что информация пакетирована в строки октетов, и описать способ ее вывода пользователям.

Тип данных, определенный через текстовые соглашения, представляется для пересылки неформатированными значениями строки октетов. Однако реальный смысл типа данных определен в описании текстового соглашения. Существуют шаблоны MIB, используемые для создания текстовых соглашений. Приведем пример описания *Display String*:

```
DisplayString :: = TEXTUAL-CONVENTION
  DISPLAY-HINT "255a"
  STATUS current
  DESCRIPTION
      "Представление текстовой информации, заимствованное из набора
       символов NVT ASCII, как определено на стр.4 и 10-11 документа RFC 854."
```

Следует помнить, что в сообщении значению всегда предшествует идентификатор объекта. Приложение станции управления могло бы использовать определение MIB, которое соответствует такому идентификатору, и применять описание текстового соглашения для выбора варианта отображения, хранения и использования полученного значения строки октетов.

20.10.10 Кодирование типов данных в BER

Наряду с языком описания типов данных ASN.1 ISO специфицировала *базовые правила кодирования* (Basic Encoding Rules — BER), которые используются для кодирования значения данных SNMP при пересылке. Кодирование BER для значений данных имеет вид:

[*Идентификатор*] [*длина содержания*] [*содержание*]

Например, идентификатор X'02 используется для INTEGER, X'04 — для строки октетов, а X'06 — для идентификаторов объектов.

Фактически все сообщение SNMP представляет собой последовательность значений ASN.1, и каждое сообщение полностью кодируется по BER.

20.11 Что дальше?

Наиболее важная часть работы, которая не была реализована в текущей версии SNMP,— это определение новой административной структуры и спецификация условий аутентификации и стандартов шифрования для безопасного удаленного конфигурирования устройств. Однако уже предложены механизмы аутентификации и шифрования трафика на уровне IP (см. главу 24).

Разработчики и авторы стандартов ведут активные поиски возможностей расширения и улучшения определений MIB, в результате чего в сетях в изобилии стала доступна сырая управляющая информация.

Необходимы хорошие приложения, чтобы эффективно использовать информацию SNMP для обнаружения сетевых проблем и долгосрочного планирования необходимой емкости ресурсов. Производители оборудования нуждаются в стандартизированных прикладных комплектах программных инструментов разработки, чтобы можно было перемещать полученные новые средства между различными станциями управления.

Интеллектуальные системы, подобные маршрутизаторам и хостам, наиболее подходят для самоконтроля. Некоторые интересные результаты были получены при встраивании в системы управляющих приложений и общения с ними через браузеры WWW и протокол HTTP.

20.12 Дополнительная литература

Существует длинный и все разрастающийся список RFC, относящихся к SNMP и MIB. Архив RFC в InterNIC содержит самые последние версии этих документов.

Наша другая книга — *SNMP: Guide to Network Management* — содержит описание концепции и структуры SNMP и детальный разбор некоторых модулей MIB.

Глава
21

Программный интерфейс socket

21.1 Введение

Коммуникационные стандарты определяют все правила для обмена информацией в сети. Однако до некоторого момента игнорировалась необходимость стандартизации интерфейса программирования приложений (Application Programming Interface — API). Как же тогда программист должен создавать приложения клиент/сервер, если программы на каждом из компьютеров совершенно различны?

20.1.1 Программный интерфейс Berkeley

К счастью, большинство реализаций TCP/IP обеспечивает программный интерфейс, следующий очень простой модели *программного интерфейса socket*, который впервые был предложен в 1982 г. в версии 4.1c операционной системы Unix университета Беркли (Berkeley Software Distribution — BSD). Со временем в исходный интерфейс было внесено несколько усовершенствований.

Программный интерфейс socket разрабатывался для применения с различными коммуникационными протоколами, а не только для TCP/IP. Однако, когда была закончена спецификация транспортного уровня OSI, стало ясно, что этот интерфейс не согласуется с требованиями OSI.

В 1986 г. компания AT&T предложила спецификацию протокола интерфейса транспортного уровня (Transport Layer Interface — TLI) для операционной системы Unix System V. Интерфейс TLI мог применяться для транспортного уровня OSI, TCP и других протоколов.

Еще одним важным событием в истории socket стал программный интерфейс socket для Windows (WinSock), позволивший приложениям Windows функционировать поверх стеков TCP/IP, созданных разными производителями. В Windows 95 обеспечивается поддержка многопротокольного интерфейса.

Интерфейс socket стал стандартом де-факто благодаря широкому распространению и универсальности доступа. В этой главе мы рассмотрим общие принципы работы этого интерфейса. На компьютерах могут существовать незначительные отличия в API, связанные с тем, что коммуникационные службы в операционных системах реализуются по-разному. Детальную информацию по программированию в конкретной системе можно найти в технических описаниях.

21.1.2 Ориентация на Unix

Исходный вариант интерфейса socket был разработан для Unix. Архитектура этой операционной системы позволяет единообразно обращаться к файлам, терминалам и вводу/выводу. Операции с файлами предполагают использование одного из следующих вызовов:

```
descriptor = open(filename, readwritemode)
read(descriptor, buffer, length)
write(descriptor, buffer, length)
close(descriptor)
```

Когда программа открывает файл, вызов создает в памяти область, называемую *управляющим блоком файла* (file control block) и содержащую сведения о данном файле (например, имя, атрибуты и место размещения).

Вызов возвращает небольшое целое число, именуемое *дескриптором файла* (file descriptor). Дескриптор используется в программе для идентификации файла в последующих операциях. При чтении или записи в файле специальный указатель из дескриптора отслеживает текущее положение внутри файла.

Похожие методы используются в socket для TCP/IP. Главным отличием между программным интерфейсом socket и файловой системой Unix является то, что в socket применяется несколько дополнительных предварительных вызовов, необходимых для сбора всех сведений перед формированием соединения. Не считая дополнительной работы при запуске, для чтения или записи, в сети применяются те же самые операции.

21.2 Службы socket

Программный интерфейс socket обеспечивает работу трех служб TCP/IP: *потокового* обмена, обмена *датаграммами* в UDP и *пересылки* необработанных данных непосредственно на уровень IP. Все эти службы показаны на рис. 21.1.

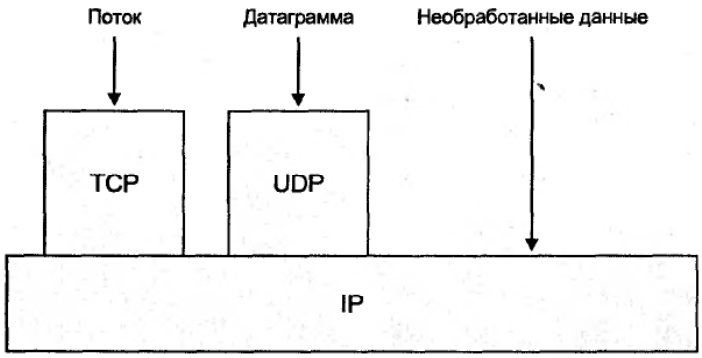

Рис. 21.1. Программный интерфейс socket

Вспомним, что API интерфейса socket разрабатывался не только для TCP/IP. Исходная цель заключалась в создании единого интерфейса для различных коммуникационных протоколов, в том числе и для XNS (Xerox Network Systems).

Результат получился несколько странным. Например, некоторые вызовы socket содержат необязательные параметры, не имеющие никакого отношения к TCP/IP — они необходимы в других протоколах. Кроме того, иногда программист обязан указывать длину для параметров фиксированного размера, например для адресов IP версии 4. Смысл этого в том, что, хотя длина адреса в IP версии 4 всегда равна 4 байт, в программных интерфейсах для других протоколов могут использоваться адреса другой длины.

21.3 Блокированные и неблокированные вызовы

Когда программа читает данные из сетевого соединения, трудно предсказать заранее, как долго будет продолжаться эта операция. Программист может только дождаться полного завершения чтения или перейти на другое место в программе и периодически проверять значение переменной статуса соединения, либо разрешить программное прерывание по окончании операции.

- Вызов с последующим ожиданием называется блокированным (blocking) или синхронным (synchronous).
- Вызов с переходом на выполнение других операций называется неблокированным (nonblocking) или асинхронным (asynchronous).

В программном интерфейсе socket вызовы могут быть блокированными или неблокированными, а программист способен управлять поведением вызова.

21.4 Вызовы socket

Вызовы *socket* подготавливают сетевое взаимодействие путем создания *блоков управления пересылкой* (Transmission Control Block — TCB). В некоторых изданиях процесс создания TCB называется созданием socket. Вызов *socket* возвращает небольшое целое число, называемое *дескриптором* и используемое для идентификации соединения во всех последующих запросах.

В TCB используется множество параметров. Перечисленные ниже параметры предоставляют информацию, необходимую для создания сеанса TCP:

- Локальный IP-адрес
- Локальный порт
- Протокол (например, TCP или UDP)
- Удаленный IP-адрес
- Удаленный порт
- Размер выходного буфера
- Размер приемного буфера
- Текущее состояние TCP
- Усредненное время цикла пересылка-получение
- Отклонение от усредненного времени цикла пересылка-получение
- Текущее время тайм-аута повторной пересылки
- Количество выполняемых повторных пересылок
- Текущий размер окна отправки
- Максимальный размер отправляемого сегмента
- Порядковый номер последнего подтвержденного по ACK байта
- Максимальный размер получаемого сегмента
- Порядковый номер следующего отправляемого байта
- Разрешение/запрещение отслеживания

21.5 Программирование работы TCP socket

Рассмотрим вызовы из программ к socket, используемые при взаимодействии с TCP. Для упрощения не будем указывать в вызовах параметры ввода/вывода и сконцентрируемся на более важных функциях и их взаимоотношениях. Детали формирования параметров описаны ниже.

21.5.1 Модель сервера TCP

Типичный сценарий для взаимодействия с сервером TCP предполагает наличие главного процесса, который большую часть времени отслеживает запросы от клиентов. Когда клиент соединяется с сервером, сервер обычно создает новый дочерний процесс, который будет реально выполнять всю работу для клиента. Сервер передает клиента этому дочернему процессу и снова возвращается к отслеживанию запросов от других клиентов.

Иногда клиенты появляются быстрее, чем их может обслужить главный процесс. Как поступить в этом случае? Стандартный механизм заключается в том, что при запуске главного процесса в TCP создается очередь, которая способна хранить несколько запросов на соединение. Запросы клиентов, которые нельзя обслужить сразу, помещаются в очередь и обрабатываются в порядке этой очереди. Предположим, что очередь заполнена до конца и поступает запрос от очередного клиента. В этом случае соединение с новым клиентом не будет создано.

21.5.2 Пассивное открытие сервера TCP

Сервер готовится к принятию запроса на соединение и пассивно ожидает обращения клиентов. При подготовке он выполняет ряд запросов:

socket ( ) — Сервер идентифицирует тип связи (в данном случае TCP). Локальная система создает соответствующую структуру данных TCB для взаимодействия с сервером и возвращает *дескриптор socket.*

bind ( ) — Сервер устанавливает локальный IP-адрес и порт, которыми он будет пользоваться. Вспомним, что хост может иметь несколько IP-адресов. Сервер может применять один IP-адрес или указать, что желает принимать соединения от любого локального IP-адреса. Он может запросить определенный порт или разрешить связывание запроса с одним из доступных свободных портов.

listen ( ) Сервер устанавливает длину очереди для клиентов.

accept ( ) Сервер готов принимать соединения от клиентов. Если очередь не пуста, принимается первый полученный клиентский запрос. Запрос *accept* ( ) создает *новый TCB*, который будет использоваться для соединения этого клиента и возвращать *новый дескриптор* соединения серверу.

Обычно применяется синхронная форма *приема* запросов, чтобы при пустой очереди *accept* ( ) ожидал появления следующего клиента до ответа на полученный запрос.

21.5.3 Активное открытие клиента TCP

Открытый клиент активно запрашивает соединение через два запроса:

socket ( ) Клиент идентифицирует тип связи (в данном случае TCP). Локальная система создает соответствующую структуру данных TCB для соединения и возвращает локальный дескриптор socket.

connect ( ) Клиент указывает IP-адрес и порт сервера. TCP попытается установить соединение с сервером.

Если клиент желает явно определить применяемый далее локальный порт, он должен вызвать *bind* ( ) перед выдачей запроса *connect* ( ). Если порт доступен, он присваивается клиенту.

Если клиент запросил порт не через *bind* ( ), ему присваивается один из неиспользованных портов. Номер порта вводится в TCB.

21.5.4 Другие запросы

Оставшиеся запросы используются клиентом и сервером аналогичным способом. Данные могут быть переданы и получены через обычные запросы *записи* и *чтения*. Соединение может быть закрыто по запросу *close*. Существуют также специальные запросы *send* и *recv*, поддерживающие отправку и получение как срочных, так и обычных данных:

send ( ) Запись буфера данных в socket. Как альтернативу можно применить *write* ( ).

sendv ( ) Пересылка в socket последовательности буферов. Как альтернативу можно применить *writev* ( ).

recu ( ) Получение буфера данных из socket либо из *read* ( ).

recvmsg ( ) Получение последовательности буферов из socket либо из *readv* ( ).

Иногда программе нужна информация, хранящаяся в TCB:

getsockopt ( ) Чтение выбранной информации из TCB. Иногда система обеспечивает необязательные системные запросы ввода/вывода, которые позволяют читать различные части TCB.

Проверка входных параметров запросов на открытие, отправку или получение показывает, что этих параметров очень мало. Причина в том, что обычно для большинства параметров TCB используются значения по умолчанию, содержащие важную информацию об окружении, например о размере приемного буфера, разрешении регистрации событий либо об использовании блокированной или неблокированной обработки в запросах, подобных *recv*. Некоторые значения по умолчанию можно изменить с помощью функций:

setsockopt ( ) Устанавливает значения нескольких параметров TCB, например размеры приемного и выходного буферов, пересылку срочных данных в общем порядке оправки информации либо блокировку закрытия соединения до благополучной отправки всех данных.

iocntl ( ) или *fcntl* ( ) Устанавливает ввод/вывод в socket в режим блокирования или снимает блокирование.

На рис. 21.2 демонстрируется последовательность вызовов в типичном сеансе TCP. Вызовы *socket* ( ), *bind* ( ) и *listen* ( ) обрабатываются очень быстро, и на них немедленно возвращается ответ.

Вызовы *accept* ( ), *send* ( ) и *recv* ( ) предполагаются в режиме блокирования (что является их обычным значением по умолчанию). Вызов *send* блокируется и при переполнении выходного буфера TCP. Вызовы *write* ( ) и *read* ( ) можно использовать вместо *send* ( ) и *recv* ( ).

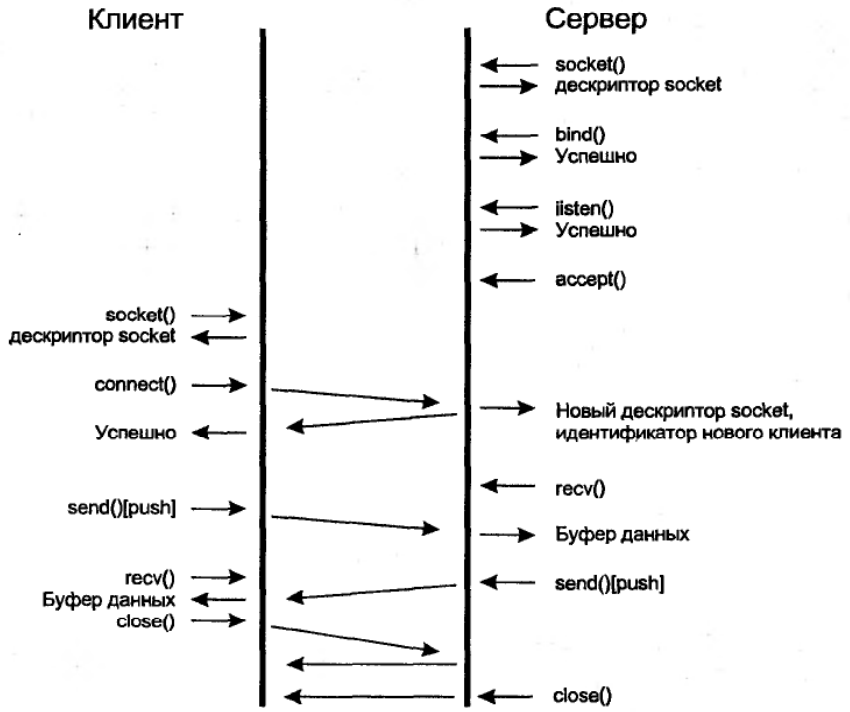

Рис. 21.2. Последовательность программных вызовов в socket TCP

21.6 Серверная программа TCP

Рассмотрим подробно пример серверной программы. Сервер предназначен для непрерывной работы. Он будет выполнять следующие действия:

1. Запрашивать у *socket* создание главного TCB и возвращать значение дескриптора socket, который будет идентифицировать этот TCB в последующих вызовах.
2. Вводить локальный адрес сервера socket в структуру данных программы.
3. Запрашивать *связывание*, при котором в TCB копируется локальный адрес socket.
4. Создавать очередь, которая сможет хранить сведения о пяти клиентах.

Оставшиеся шаги повторяются многократно:

5. Ожидать запросов от клиентов. Когда появляется клиент, создавать для него новый TCB на основе копии главного TCB и записи в него адреса socket клиента и других параметров.
6. Создавать дочерний процесс для обслуживания клиента. Дочерний процесс будет наследовать новый TCB и обрабатывать все дальнейшие операции по связи с клиентом (ожидать сообщений от клиента, записывать их и завершать работу).

Каждый шаг в программе объясняется в следующем разделе.

```
/* tcpserv.c
 * Для запуска программ ввести "tcpserv".
 */

/* Сначала включить набор стандартных заголовочных файлов.
*/
#include <sys/types.h>
#include <sys/socket.h>
```

```
#include <stdio.h>
#include <netinet/in.h>
#include <netdb.h>
#include <errno.h>

main()
{
int sockMain, sockClient, length, child;
struct sockaddr_in servAddr;

/* 1. Создать главный блок управления пересылкой.
 */

if ( (sockMain = socket(AF_INET, SOCK_STREAM, 0)) < 0)
   { perror ("Сервер не может открыть главный socket.");
     exit (1);
   }

/* 2. Создать структуру данных для хранения локальных IP-адресов
 *    и портов, которые будут использованы. Предполагается прием
 *    клиентских соединений от любых локальных IP-адресов
 *    (INADDR_ANY). Поскольку данный сервер не применяет
 *    общеизвестный порт, установить port = 0. Это позволит
 *    связать вызов с присвоением порта серверу и записать
 *    порт в TCB.
 */
bzero( (char *) &servAddr, sizeof(servAddr));
servAddr.sin_family = AF_INET;
servAddr.sin_addr.s_addr = htonl (INADDR_ANY);
servAddr.sin_port = 0;

/* 3. Связать запрос, выбор номера порта и
 *    запись его в TCB.
 */
if ( bind(sockMain, &servAddr, sizeof(servAddr)) )
   { perror("Связывание сервера неудачно.");
     exit (1) ;
   }

/* Чтобы увидеть номер порта, следует использовать
 * функцию getsockname(), чтобы скопировать порт в servAddr.
 */
length = sizeof(servAddr);
if ( getsockname(sockMain, &servAddr, &length) )
   { perror("Вызов getsockname неудачен.");
     exit (1) ;
   }

printf ("СЕРВЕР: номер порта - % d\n", ntohs(servAddr.sin_port) );

/* 4. Создать очередь для хранения пяти клиентов.
 */
listen (sockMain, 5) ;
```

```
/* 5. Ожидать клиента. При разрешении возвратить новый
 *     дескриптор socket, который должен использоваться клиентом.
 */
for ( ; ; ) {

if ( (sockClient = accept(sockMain, 0, 0)) < 0)
   { perror ("Неверный socket для клиента.");
     exit (1) ;
   }

/* 6. Создать дочерний процесс для обслуживания клиента.
 */
if ( (child = fork()) < 0)
   { perror("Ошибка создания дочернего процесса.");
     exit (1);
   }
else if (child == 0)        /* Это код для исполнения дочернего процесса. */
      { Close(sockMain);    /* Дочерний процесс неинтересен для sockMain.*/
        childWork(sockClient);
        close(sockClient);
        exit (0);
      }

/* 7. Это родительский процесс. Его более не интересует
 *     socket клиента, поскольку его обслуживание.передано
 *     дочернему процессу. Родительский процесс закрывает свой элемент для
 *     socket клиента и переходит на цикл приема новых accept().
 */
  close(sockClient);
    }
}

/* Дочерний процесс читает один поступивший буфер, распечатывает
 * сообщение и завершается.
 */
#define BUFLEN 81
int childWork(sockClient)
int sockClient;
{
char buf [BUFLEN];
int msgLength;

/* 8. Опустошить буфер. Затем вывести recv для получения сообщения от клиента.
 */
bzero(buf, BUFLEN);
if ( (msgLength = recv(sockClient, buf, BUFLEN, 0)) < 0)
   { perror("Плохое получение дочерним процессом.");
     exit (1);
   }
printf ("SERVER: Socket для клиента - %d\n", sockClient);
printf ("SERVER: Длина сообщения - %d\n", msgLength);
printf ("SERVER: Сообщение: %s\n\n", buf);
}
```

21.6.1 Вызовы в серверной программе TCP

1. *sockMain = socket (AF_INET, SOCK_STREAM, 0);*

   Вызов socket имеет форму:

```
дескриптор_socket = socket (адрес_домена, тип_коммуникации, протокол)
```

Напомним, что интерфейс socket может использоваться для других видов коммуникаций, например XNS. *AF_INET* указывает на семейство адресов Интернета. *SOCK_STREAM* запрашивает socket TCP. Эта переменная должна иметь значение *SOCK_DGRAM*, чтобы создать socket UDP, а *SOCK_RAW* служит для непосредственного обращения к IP.

Не нужно явно определять никакую другую информацию протокола для TCP (или для UDP). Однако параметр *protocol* необходим для интерфейса с необработанными данными, а также для некоторых протоколов из других семейств, использующих socket.

2. *struct sockaddr_in servAddr;*

   ...

   *bzero ((char *) &servAddr, sizeof (servAddr));*

   servAddr.sin_family = AF_INET;

   servAddr.sin_addr.s_addr = htonl (INADDR_ANY);

   servAddr.sin_port = 0;

Программная структура *servAddr* используется для хранения адресной информации сервера. Вызов *bzero ( )* инициализирует *servAddr*, помещая нули во все параметры. Первая переменная в структуре *servAddr* указывает, что остальная часть значений содержит данные семейства адресов Интернета.

Следующая переменная хранит локальный IP-адрес сервера. Например, если сервер подключен к локальной сети Ethernet и к сети X.25, может потребоваться ограничить доступ клиентов через интерфейс Ethernet. В данной программе об этом можно не беспокоится. *INADDR_ANY* означает, что клиенты могут соединяться через любой интерфейс.

Функция *htonl ( )* имеет полное название host-to-network-long. Она применяется для преобразования 32-разрядных целых чисел локального компьютера в формат Интернета для 32-разрядного адреса IP. Стандарты Интернета предполагают представление целых чисел с наиболее значимым байтом слева. Такой стиль именуется Big Endian (стиль "тупоконечников"). Некоторые компьютеры хранят данные, располагая слева менее значимые байты, т.е. в стиле Little Endian ("остроконечников"). Если локальный компьютер использует стиль Big Endian, *htonl ( )* не будет выполнять никакой работы.

Если сервер взаимодействует через общеизвестный порт, номер этого порта нужно записать в следующую переменную. Поскольку мы хотим, чтобы операционная система сама присвоила порт для нашей тестовой программы, мы вводим нулевое значение.

3. *bind (sockMain, &servAddr, sizeof (servAddr));*
   getsockname (sockMain, &servAddr, &length);

   Вызов *bind* имеет форму:

```
возвращаемый_код = bind(дескриптор_socket, адресная_структура,
длина_адресной_структуры)
```

Если адресная структура идентифицирует нужный порт, *bind* попытается получить его на сервере. Если переменная порта имеет значение 0, *bind* получит один из неиспользованных портов. Функция *bind* позволяет ввести номер порта и IP-адрес в TCB. Вызов *getsockname* имеет форму:

```
возвращаемый_код = getsockname(дескриптор_socket, адресная_структура,
длина_адресной_структуры)
```

Мы запросили у *bind* выделение порта, но эта функция не сообщает нам, какой именно порт был предоставлен. Для выяснения этого нужно прочитать соответствующие данные из TCB. Функция *getsockname ( )* извлекает информацию из TCB и копирует ее в адресную структуру, где можно будет прочитать эти сведения. Номер порта извлекается и выводится следующим оператором:

```
printf ("SERVER: Номер порта %d\n", ntohs(servAddr.sin_port) );
```

Функция *ntohs ( )* имеет полное название network-to-host-short и служит для преобразования номера порта из порядка следования байт в сети в локальный порядок следования байт на хосте.

4. *listen* (*sockMain,5*);

Вызов *listen* применяется для ориентированных на соединение серверов и имеет форму:

```
возвращаемый_код = listen(дескриптор_socket, размер_очереди)
```

Вызов *listen* указывает, что это будет пассивный socket, и создает очередь требуемого размера для хранения поступающих запросов на соединения.

5. *sockClient* = *accept* (*sockMain, 0, 0*);

Вызов *accept* имеет форму:

```
новый_дескриптор_socket = accept(дескриптор_socket,
клиентская_адресная_структура, длина_клиентской_адресной_структуры)
```

По умолчанию вызов блокируется до соединения клиента с сервером. Если указана переменная *клиентская_адресная_структура*, после соединения клиента в эту структуру будут введены IP-адрес и порт клиента. В этом примере программы не проверяются IP-адрес и номер порта клиента, а просто два последних поля параметра заполняются нулями.

6. *child* = *fork* ( );
   ...
   close (*sockMain*);

В языке С команда *fork* создает новый дочерний процесс, который наследует все дескрипторы ввода/вывода родительской программы, а также имеет доступ к *sockMain* и *sockClient*. Операционная система отслеживает количество процессов, имеющих доступ к socket.

Соединение закрывается, когда последний обращающийся к socket процесс вызывает *close* ( ). Когда дочерний процесс закрывает *sockMain*, родительский процесс все еще имеет доступ к socket.

7. *close* (*sockClient*);

Этот вызов выполняется из родительской части программы. Когда родительский процесс закрывает *sockClient*, дочерний процесс все еще имеет доступ к socket.

8. *msgLength* = *recv* (*sockClient,buf, BUFLEN, 0*));
   ...
   close (*sockClient*);

   Вызов *recv* имеет форму:

```
длина_сообщения = recv(дескриптор_socket, буфер, длина_буфера, флаги)
```

По умолчанию вызов *recv* блокированный. Функции *fcntl* ( ) или *iocntl* ( ) позволяют изменить статус sockct на неблокированный режим.

После получения данных дочерним процессом и вывода сообщения на печать, доступ к *sockClient* закрывается. Это заставит соединение перейти в фазу закрытия.

21.7 Клиентская программа TCP

Клиент соединяется с сервером, посылает одно сообщение, и далее работа программы завершается (фрагменты программы рассматриваются в следующем разделе). Для запуска программы конечный пользователь должен ввести имя хоста сервера, номер порта и сообщение, которое будет послано на этот сервер. Например:

```
tcpclient plum.cs.yale.edu 1356 hello
```

```
/* tcpclient.c
 * Перед запуском клиента должен быть запущен сервер. Производится
 * поиск порта сервера. Для запуска клиента нужно ввести:
 * tcpclient имя_хоста порт сообщение
 */
#include <sys/types.h>
#include <sys/socket.h>
```

```
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <errno.h>

main(argc, argv)        /* Клиентская программа имеет входные аргументы. */
int argc;
char* argv [ ];
{
int sock;
struct sockaddr_in servAddr;
struct hostent *hp, *gethostbyname();

/* Аргументами будут 0:имя_программы, 1:имя_хоста, 2:порт, 3:сообщение
 */
if (argc < 4)
   {printf("ВВЕСТИ tcpclient имя_хоста порт сообщение\n");
    exit (1);
   }

/* 1. Создание TCB. */
if ( (sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
   {perror ("Не могу получить socket\n");
    exit (1);
   }

/* 2. Заполнить поля адреса и порта сервера в servAddr.
 *    Сначала заполнить нулями адресную структуру. Затем получить IP-адрес
 *    для данного имени хоста и ввести его в адресную структуру.
 *    Наконец ввести номер порта, взяв его из argv[2].
 */
bzero( (char *) &servAddr, sizeof (servAddr) );
servAddr.sin_family = AF_INET;
hp = gethostbyname (argv [1]);
bcopy(hp->h_addr, &servAddr.sin_addr, hp->h_length);
servAddr.sin_port = htons(atoi(argv[2]) );

/* 3. Соединиться с сервером. Вызывать bind не нужно.
 *    Система присвоит свободный порт во время выполнения соединения.
 */
if ( connect (sock, &servAddr, sizeof (servAddr) ) < 0)
   {perror("Клиент не может соединиться.\n");
    exit (1) ;
   }

/* 4. Клиент анонсирует свою готовность послать сообщение.
 *    Сообщение отправляется, и распечатывается последняя строка.
 */

printf ("CLIENT: Готов к пересылке\n");

if (send(sock, argv[3], strlen(argv[3]), 0) < 0)
   (perror("Проблемы с пересылкой.\n");
    exit (1);
   }
```

```
printf ("CLIENT: Пересылка завершена. Счастливо оставаться.\n");
close (sock);
exit (0);
}
```

21.7.1 Вызовы в клиентской программе TCP

1. *sock = socket (AF_INET, SOCK_STREAM, 0);*

Клиент создает блок управления пересылкой ("socket") так же, как это делал сервер.

2. Сервер должен инициализировать адресную структуру для использования в *bind.*

Эта структура содержит локальный IP-адрес и номер порта сервера. Клиент также инициализирует адресную структуру, хранящую те же сведения. Эта структура будет использоваться в вызове *connect* для указания точки назначения.

Вызов *bzero* ( ) помещает нули в *servAddr* — адресную структуру сервера. Еще раз мы трактуем семейство адресов как Интернет.

Затем нужно преобразовать введенное пользователем имя хоста в IP-адрес. Это делает функция *gethostbyname*, которая возвращает указатель на структуру *hostent*, содержащую имя сервера и IP-адрес.

Функция *bcopy* применяется для копирования IP-адреса (который находится в *hp->h_addr*) в *servAddr.*

Второй введенный конечным пользователем аргумент определял порт сервера. Он читался как текстовая строка ASCII, поэтому ее сначала нужно преобразовать в целое число через *atoi* ( ), а затем изменить порядок следования байт через *htons* ( ). Наконец номер порта копируется в адресную переменную из *servAddr.*

```
bzero( (char *) &servAddr, sizeof(servAddr) );
servAddr.sin_family = AF_INET;
hp = gethostbyname (argv [1]);
bcopy(hp->h_addr, &servAddr.sin_addr, hp->h_length);
servAddr.sin_port = htons(atoi(argv[2]) );
```

3. *connect (sock, &servAddr, sizeof (servAddr));*

Вызов *connect* имеет форму:

```
connect (дескриптор_socket, адресная_структура, длина_адресной_структуры)
```

Клиент откроет соединение с сервером, IP-адрес и порт которого хранятся в адресной структуре.

4. *send (sock, argv [3], strlen (argvs [3]), 0);*

Вызов *send* имеет форму:

```
возвращаемый_код = send(дескриптор_socket, буфер, длина_буфера, флаги)
```

Отметим, что введенный конечным пользователем третий аргумент (который появляется в программе как *argv [3]*) — это текст отправляемого сообщения. Обычно флаги используются для сообщения о срочных данных. В нашем случае параметры флагов установлены в 0.

5. *close (sock);*

Клиент выполняет *close* для закрытия соединения.

21.8 Более простой сервер

Многие серверы разрабатываются как в показанном выше примере. Однако можно использовать более упрощенную модель, когда сервер должен выполнять только простые запросы клиента (см. ниже).

Вместо создания дочернего процесса для каждого клиента сервер может непосредственно выполнять запрос, а затем закрывать соединение. Очередь сервера позволяет нескольким другим клиентам ожидать, пока он не будет готов обработать их запросы.

Ниже приведен листинг для более простого сервера. К этому серверу клиенты также могут обращаться через рассмотренную выше программу *tcpclient*.

```
/* tcpsimp.c
 * Для запуска программ ввести "tcpsimp"
 */

/* Сначала включить стандартные заголовочные файлы.
 */
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <netinet/in.h>
#include <netdb.h>
#include <errno.h>

main ( )
{
int sockMain, sockClient, length, child;
struct sockaddr_in servAddr;

/* 1. Создать главный socket.
 */
if ( (sockMain = socket (AF_INET, SOCK_STREAM, 0)) < 0)
   {perror("Сервер не может открыть главный socket.");
    exit (1);
   }

/* 2. Ввести информацию в структуру данных, используемую для
 *    хранения локального IP-адреса и порта. "sin" в именах
 *    переменных - это сокращение от "socket internet".
 */
bzero( (char *) &servAddr, sizeof (servAddr) );
servAddr.sin_family = AF_INET;
servAddr.sin_addr.s_addr = htonl(INADDR_ANY);
servAddr.sin_port = 0;

/* 3. Вызвать bind, которая запишет используемый номер порта
 *    в servAddr.
 */
if ( bind(sockMain, &servAddr, sizeof (servAddr)) )
   { perror("Вызов bind от сервера неудачен.");
     exit (1);
   }

/* 4. Чтобы узнать номер порта, следует использовать функцию
 *    getsockname() для копирования порта в servAddr.
 */
length = sizeof(servAddr);
if ( getsockname(sockMain, &servAddr, &length) )
   { perror("Вызов getsockname неудачен.");
     exit (1);
   }
```

```
printf ("SERVER: Номер порта %d\n", ntohs(servAddr.sin_port) );

/* 5. Установить очередь на пять клиентов.
 */
listen (sockMain, 5);

/* 6. Ожидать поступления клиентов. Вызов accept возвратит
 *    дескриптор нового socket, который следует использовать клиенту.
 */

for ( ; ; ) {
if ( (sockClient = accept(sockMain, 0, 0)) < 0)
   { perror("Неверный socket клиента.");
     exit (1);
   }

/* 7. Обслужить клиента и закрыть соединение с ним.
 */
   doTask(sockClient);
   close (sockClient);
   }
}

/* Читать один поступивший буфер, распечатать некоторую информацию
 * и завершить работу.
 */

#define BUFLEN 81
int doTask(sockClient)
int sockClient;
{
char buf[BUFLEN];
int msgLength;

/* 8. Опустошение буфера и вызов recv
 *    для получения сообщения от клиента.
 */
bzero (buf, BUFLEN);
if ( (msgLength = recv(sockClient,buf, 80, 0)) < 0)
   { perror("Неверное получение." );
     exit (1);
   }

printf ("SERVER: Socket для клиента %d\n", sockClient);
printf ("SERVER: Длина сообщения %d\n", msgLength);
printf ("SERVER: Сообщение: %s\n\n", buf);
}
```

21.9 Интерфейс программирования socket для UDP

Мы познакомились с наиболее общим интерфейсом программирования TCP. Теперь рассмотрим программирование сервера и клиента UDP. На рис. 21.3 показана схема диалога UDP между клиентом и сервером. Вызовы *socket* ( ) и *bind* ( ) быстро выполняются и немедленно возвращают ответ. Вызов *recvfrom* предполагает режим блокирования по умолчанию, который можно изменить на неблокированный (т.е. асинхронный) режим.

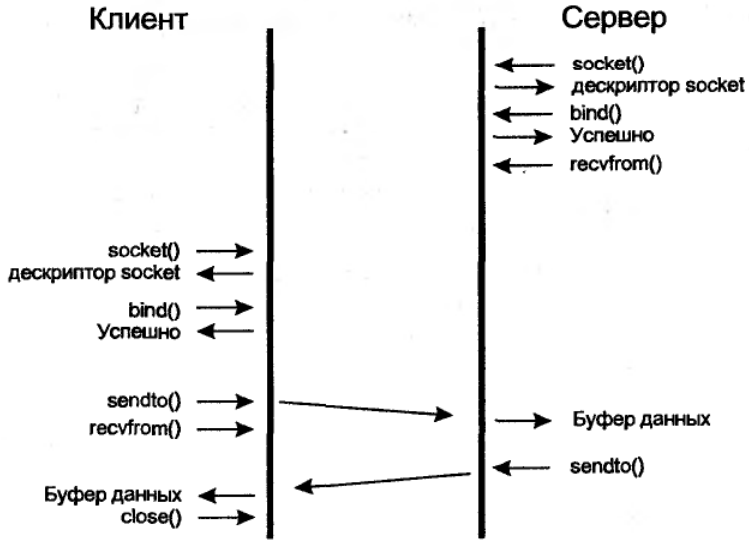

Рис. 21.3. Типичные программные вызовы в socket UDP

21.10 Программа сервера UDP

Показанная ниже программа создает socket для UDP, связывает вызов с портом, а затем получает и распечатывает сообщения, которые посылаются на этот порт:

```
/* udpserv.c
 * Для запуска программы ввести "udpserv".
 *
 * Сначала включить стандартные заголовочные файлы.
 */
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <netinet/in.h>
#include <netdb.h>
#include <errno.h>
#define BUFLEN 81

main()
{
int sockMain, addrLength, msgLength;
struct sockaddr_in servAddr, clientAddr;
char buf[BUFLEN];

/* 1. Создать socket для UDP.
 */
if ( (sockMain = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
   {perror("Сервер не может открыть socket для UDP.");
    exit (1);
   }
```

```
/* 2. Ввести информацию в структуру данных, используемую для хранения локальных
 *     IP-адресов и порта. Возложить на bind получение свободных портов.
 */
bzero( (char *) &servAddr, sizeof (servAddr) );
servAddr.sin_family = AF_INET;
servAddr.sin_addr.s_addr = htonl(INADDR_ANY);
servAddr.sin_port = 0;

/* 3. Вызвать bind, которая запишет номер используемого порта
 *     в TCB.
 */
if ( bind(sockMain, &servAddr, sizeof(servAddr)) )
   { perror("Вызов bind от сервера неудачен.");
     exit (1);
   }

/* 4. Извлекаем номер порта и используем функцию
 *     getsockname() для копирования порта в servAddr.
 */
addrLength = sizeof(servAddr);
if ( getsockname(sockMain, &servAddr, &addrLength) )
   { perror(Вызов getsockname неудачен.");
     exit (1);
   }

printf("SERVER: Номер порта is %d\n", ntohs(servAddr.sin_port));

/* 5. Бесконечный цикл ожидания сообщений от клиентов.
 */

for ( ; ; ) {
  addrLength = sizeof(clientAddr);
  bzero(buf, BUFLEN);
  if ( (msgLength = recvfrom(sockMain, buf, BUFLEN, 0,
                    &clientAddr, &addrLength)) < 0)
     { perror("Плохой socket клиента.");
       exit (1);
     }

/* 6. Распечатать клиентские IP-адрес и порт вместе с сообщением.
 */

  printf("SERVER: IP-адрес клиента: %s\n",
                     inet_ntoa( clientAddr.sin_addr) );

  printf("SERVER: Порт клиента: %d\n",
                     ntohs(clientAddr.sin_port) );
  printf("SERVER: Длина сообщения %d\n", msgLength);
  printf("SERVER: Сообщение: %s\n\n", buf);
  }
}
```

21.10.1 Вызовы в серверной программе UDP

1. *sockMain = socket (AF_INET, SOCK_DGRAM, 0);*

Семейство адресов — снова Интернет.

2. *bzero ((char *)&servAddr, sizeof (servAddr));*
servAddr.sin_family = AF_INET;
servAddr.sin_addr.s_addr = htonl (INADDR_ANY);
servAddr.sin_port = 0;

Вызовы инициализации адресной структуры сервера те же, что и в программе для TCP.

3. *bind (sockMain, &servAddr, sizeof (servAddr));*

Как и прежде, *bind* получает порт для сервера и записывает значения в TCB. Конечно, по сравнению с TCP, UDP содержит очень мало информации.

4. *getsockname (sockMain, &servAddr, &length);*

Использовать *getsockname*, чтобы извлечь присвоенный socket порт.

5. *msgLength = recvfrom (sockMain, buf, BUFLEN, 0, &clientAddr, &length);*

Вызов *recvfrom* имеет форму:

```
recvfrom(дескриптор_socket, входной_буфер, длина_буфера, флаги,
исходная_адресная_структура,
указатель_на_длину_исходной_адресной_структуры)
```

Флаги позволяют вызывающей стороне просмотреть сообщение без его фактического получения. После возвращения исходная адресная структура заполняется IP-адресом и номером порта клиента. Необходим указатель на длину исходного адреса, поскольку она может быть изменена при вставке в поле фактического адреса клиента.

6. *inet_ntoa (clientAddr.sin_addr);*

Этот вызов преобразует 32-разрядный адрес Интернета клиента в знакомую нам нотацию этого адреса с точками и десятичными значениями.

21.11 Клиентская программа UDP

Клиент соединяется с сервером, посылает одно сообщение и закрывает соединение. При запуске программы конечный пользователь должен ввести имя хоста, порт сервера и отправляемое на сервер сообщение. Например:

```
udpclient plum.cs.yale.edu 2315 "Это сообщение."
```

```
/* udpclient.c
 * Перед запуском клиента следует запустить сервер.
 * Далее нужно получить порт сервера.
 * Для запуска клиента ввести:
 * udpclient имя_хоста порт сообщение
 */
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <errno.h>
```

```
main(argc, argv)
int argc;
char *argv[];  /* Это вводимые пользователем аргументы. */
               /* argv[0] - имя программы. argv[1] указывает на имя хоста. */
               /* argv[2] ссылается на порт, */
               /* а argv [3] ссылается на текстовое сообщение. */

{
int sock;
struct sockaddr_in, servAddr, clientAddr;
struct hostent *hp, *gethostbyname();

/* Должно быть четыре аргумента. */
if (argc < 4)
   {printf ("ВВЕСТИ udpclient имя_хоста порт сообщение\n");
    exit (1);
   }

/* 1. Создать socket для UDP. */
if ( (sock = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
   {perror("Не получен socket\n">;
    exit (1);
   }

/* 2. Занести адрес и порт сервера в servAddr.
 *    Сначала заполнить адресную структуру нулями.
 *    Использовать функцию gethostbyname для получения имени хоста
 *    и его IP-адреса. Затем скопировать IP-адрес
 *    в servAddr функцией bcopy.
 *    Наконец занести номер порта из argv[2].
 */
bzero( (char *) &servAddr, sizeof(servAddr) );
servAddr.sin_family = AF_INET;
hp = gethostbyname (argv[1]);
bcopy(hp->h_addr, &servAddr.sin_addr, hp->h_length);
servAddr.sin_port = htons(atoi(argv[2]) );

/* 3. Вызвать bind для получения порта UDP. Система
 *    назначает свободный порт.
 */
bzero( (char *) &clientAddr, sizeof(clientAddr) );
clientAddr.sin_family = AF_INET;
clientAddr.sin_addr.s_addr = htonl(INADDR_ANY);
clientAddr.sin_port = 0;
if ( bind(sock, &clientAddr, sizeof(clientAddr)) < 0)
    {perror("Клиент не получил порт.\n");
     exit (1);
    }
```

```
/* 4. Клиент анонсирует свою готовность к приему сообщений.
 *    Он посылает сообщение и распечатывает последнюю строку.
 */
printf ("CLIENT: Готов к пересылке\n">;
if (sendto(sock, argv[3], strlen(argv[3]), 0, &servAddr,
 sizeof(servAddr)) < 0)
      {perror "Проблема с sendto.\n");
       exit (1);
      }
printf ("CLIENT: Пересылка закончена. Счастливо.\n");
/* Закрытие socket */
close (sock);
}
```

21.11.1 Запросы в клиентской программе UDP

1. *sock = socket (AF_INET, SOCK_DGRAM, 0);*

UDP клиента создает socket для UDP.

2. *bzero ((char *) &servAddr, sizeof (servAddr));*
   servAddr.sin_family =AF_INET;
   hp = gethostbyname (argv [1]);
   bcopy (hp->h_addr, &servAddr.sin_addr, hp->h_length);
   servAddr.sin_port =htons (atoi (argv [2]));

Структура *servAddr* заполнена введенными конечным пользователем значениями, как это делалось и в клиенте для TCP.

3. *bind (sock, &clientAddr, sizeof (clientAddr);*

Клиент вызывает *bind* для получения порта.

4. *sendto (sock, argv [3], strlen (argv [3]), 0, &servAddr, sizeof (servAddr));*

Вызов *sendto* имеет форму:

```
sendto(дескриптор_socket, буфер, длина_буфера, флаги,
       адресная_структура_назначения,
       длина адресной_структуры_назначения)
```

Этот запрос содержит всю информацию о точке назначения, необходимую для отправки датаграммы протокола UDP.

21.12 Дополнительная литература

Любое техническое руководство по программированию в Unix содержит описания программных вызовов socket. В книге Ричарда Стивенса (Richard Stevens) *Unix Network Programming* детально обсуждается программирование socket. Руководства программиста TCP/IP для других операционных систем описывают вызовы socket и часто содержат примеры типичных программ. Следует ознакомиться с подобным руководством, поскольку между операционными системами могут существовать различия.

Глава
22

IP версии 6

22.1 Введение

За относительно короткий период времени персональные компьютеры стали подключаться к локальным сетям, которые объединялись в региональные сети. Многие из этих систем связаны со всем миром. Результатом стало распространение Интернета среди миллионов пользователей.

Структура исходной схемы адресации IP не вполне подходит для такого окружения. Пространство номеров ограничено, и, в отличие от структуры телефонных номеров с иерархической системой кодов стран и областей, здесь нумерация не является иерархической. Присваивание организациям блоков адресов не слишком эффективно, поскольку большая часть адресного пространства не используется.

Пространство номеров быстро истощилось. Кроме того, поскольку номера присваивались не иерархически, быстро разрослись таблицы маршрутизации.

Однако расширение Интернета нельзя остановить. Количество персональных компьютеров и их подключений к глобальной сети постоянно растет. Появляются новые потребности:

- Работа в сети нового поколения мобильных компьютеров, заменяющих деловые бумаги и выполняющих роль цифрового персонального секретаря.
- Требования для аудио и видео в реальном времени, приведшие существующую технологию к своему естественному пределу.

К Интернету обратился серьезный деловой мир, которому требуется обеспечение реальной безопасности сетевых инфраструктур.

Существует и проблема с сетевым управлением. Многие организации используют магистральные соединения по IP для объединения своих сетей и пересылке трафика средствами протокола IP. На сегодняшний день эти задачи не решены полностью и нет эффективных механизмов для управления нагрузкой.

Разработка IP версии 6 (IPv6, называемый еще *IP следующего поколения*) проводилась для решения проблем адресации, маршрутизации, производительности, безопасности и нагрузки в Интернете. В этой главе рассматриваются наиболее важные возможности IPv6. При реализации следует учитывать требования самых последних RFC.

22.2 Обзор IPv6

Протокол IPv6 имеет следующие характеристики:

- Введен 128-разрядный адрес (16 октетов), который иерархически структурирован для упрощения делегирования прав выделения адресов и маршрутизации.
- Упрощен главный заголовок IP, но определены многие необязательные *заголовки расширения*, что позволяет при необходимости добавлять новые сетевые возможности.
- Поддерживаются аутентификация, целостность данных и конфиденциальность на уровне IP.
- Введены *потоки*, поддерживающие многие новые типы пересылки запросов, например видео в реальном времени.
- Упрощена инкапсуляция других протоколов, и предложен механизм для управления нагрузкой при пересылке данных от других протоколов.

- Реализован новый метод автоматической самоконфигурации адресов и проверки уникальности IP-адресов.
- Улучшены методы исследования маршрутизаторов, определения неисправных путей и недостижимых соседей по связи.

На момент написания книги многие детали IPv6 находились еще в стадии разработки, однако основные архитектурные элементы уже подготовлены и рассматриваются в этой главе. Уже стали стандартами IPv6, ICMPv6, расширение DNS и архитектура адресации IPv6.

22.3 Терминология

Версия 6 вносит некоторые изменения в терминологию версии 4 и вводит новые термины:

- *Пакетом* (packet) называется заголовок IPv6 плюс полезные данные
- *Узел* (node) — любая система, реализующая IPv6
- *Маршрутизатор* (router) — узел, пересылающий не адресованные ему пакеты IPv6
- *Связь* (link) — носитель, по которому взаимодействуют узлы на уровне связи данных
- *Соседи* (neighbor) — узлы, подключенные к одной связи

Термином "*пакет*" наиболее злоупотребляют в сетевом мире. Пакетами называются любые элементы данных протокола (PDU) от уровня связи данных до уровня приложений.

Почему авторы версии 6 перешли от термина "*датаграмма*" к "*пакету*"? Одно из новшеств в IPv6 — это возможность переноса трафика для многих других протоколов. Следовательно, полезные данные не обязательно будут PDU из набора протоколов TCP/IP. Когда пересылается родной PDU из IP, можно пользоваться термином "*датаграмма*".

В этой главе рассматриваются текущие документы IPv6 и используется термин "*пакет*".

22.4 Адреса IPv6

Адреса IPv6 имеют длину 16 октетов (128 бит). Для записи адресов используется компактная (хотя и уродливая) нотация. Адреса представлены как 8 шестнадцатеричных чисел, разделенных двоеточиями. Каждое шестнадцатеричное число представляет 16 бит. Например:

```
41BC:0:0:0:5:DDE1:8006:2334
```

Ведущие нули в шестнадцатеричных полях могут быть опущены (например, 0 вместо 0000 и 5 вместо 0005). Формат может быть еще более сжат при замене последовательности смежных нулевых полей на "::". Например:

```
41BC::5:DDE1:8006:2334
```

Отсутствуют три позиции, так как "::" заменяет последовательность ":0:0:0:".

Адреса версии 4 протокола IP часто вкладываются в последние четыре октета адреса версии 6. Они могут быть записаны с использованием смешанного формата адреса (сочетающего как нотацию с точками, так и нотацию с двоеточиями), например:

```
0:0:0:0:0:FFFF:128.1.35.201
```

22.4.1 Выделение адресов

128-разрядное пространство адреса обеспечивает место для множества различных типов адресов, включая:

- Иерархические глобальные одноадресные рассылки на основе адресов провайдеров
- Иерархические глобальные одноадресные рассылки по географическому признаку
- Личные адреса сайтов для использования только в пределах организации
- Локальные и глобальные многоадресные рассылки

Версия 6 не использует широковещательные рассылки, но для функций управления (например, разрешения адресов или загрузки) использует многоадресные рассылки. Это связано с тем, что сообщения широковещательных рассылок прерывают работу всех устройств связи, хотя в большинстве случаев

они предназначаются лишь для небольшого количества устройств. Кроме того, ограничение управляющих сообщений только многоадресными рассылками предотвращает взаимовлияние устройств версии 6 и версии 4, совместно использующих одну и ту же связь.

22.4.2 Общие принципы выделения адресов

Работу по делегированию прав присвоения блоков адресного пространства IPv6 региональным организациям регистрации ведет *Internet Assigned Numbers Authority* (IANA). Региональные организации регистрации могут передавать блоки адресов в меньшие географические области, национальные организации или провайдерам.

В таблице 22.1 показана общая схема распределения адресного пространства:

- Большой блок используется для адресации провайдеров.
- Имеются блоки, выделенные автономным локальным сетям или отдельным сайтам, которые не связаны с Интернетом, что позволяет им самостоятельно присваивать адреса.
- Специальные блоки предоставлены для адресов IPX и точек доступа к сетевым службам модели OSI (OSI Network Service Access Point — NSAP).
- Большой блок зарезервирован для адресации по географическому принципу.

В настоящее время почти 3/4 адресного пространства не предназначено для конкретного использования.

Таблица 22.1

Выделение адресного пространства IPv6

Выделено	Префикс (двоичный)	Доля адресного пространства
Зарезервировано	0000 0000	1/256
Не присвоено	0000 0001	1/256
Зарезервировано для NSAP	0000 001	1/128
Зарезервировано для IPX	0000 010	1/128
Не присвоено	0000 011	1/128
Не присвоено	0000 1	1/32
Не присвоено	0001	1/16
Не присвоено	001	1/8
Одноадресные рассылки среди провайдеров	**010**	**1/8**
Не присвоено	011	1/8
Зарезервировано для одноадресных рассылок по географическому принципу	100	1/8
Не присвоено	101	1/8
Не присвоено	110	1/8
Не присвоено	1110	1/16
Не присвоено	1111 0	1/32
Не присвоено	1111 10	1/64
Не присвоено	1111 110	1/128
Не присвоено	1111 1110 0	1/512
Адреса для локальных связей	1111 1110 10	1/1024
Адреса для локальных сайтов	1111 1110 11	1/1024
Многоадресные рассылки	1111 1111	1/256

22.4.3 Префикс формата адреса

Первые несколько бит адреса называются *префиксом формата* (format prefix) и идентифицируют тип адреса. Например, префикс 010 определяет IP-адреса для одноадресных рассылок между провайдерами. Формат остальной части адреса зависит от префикса формата.

22.4.4 Адресация провайдеров

В настоящее время для адресов провайдеров предложена простая иерархическая структура:

3 бита	n бит	m бит	o бит	125-n-m-o бит
010	Идентификатор регистратора	Идентификатор провайдера	Идентификатор подписчика	Идентификатор интра-подписчика

Маршрутизация провайдера проста. Достаточно сравнить первую часть адреса со строками в таблице маршрутизации. Далее провайдер может маршрутизировать данные к своим подписчикам, сравнивая большой фрагмент адреса со строкой своей таблицы.

При такой схеме адреса организация подписчика будет иметь достаточное адресное пространство, чтобы построить удобную внутреннюю иерархию. Организация может структурировать адресное пространство на подсети и хосты (как это делается и сейчас) или прибавить один или несколько дополнительных уровней иерархии. Например, иерархия организации может состоять из областей, подсетей и хостов.

В адресах версии 6 не запрещаются поля со всеми единицами или нулями.

22.4.5 Адреса для независимых сайтов

В настоящее время не связанная с Интернетом локальная сеть в версии 4 использует специальный блок адресов, например 10.0.0.0 или 172.16.0.0, который был зарезервирован для этой цели. Но, если организация впоследствии должна соединиться с внешним миром, потребуется вручную переконфигурировать сеть.

Версия 6 предоставляет более удобные способы переназначения адреса (см. ниже).

22.4.6 Адреса локальных связей

Связь — это вариант коммуникации, например Ethernet (в версии 6 для него определен новый код типа X'86-DD), Token-Ring, FDDI, сети Frame Relay, ATM или линии "точка-точка". Легко автоматизировать адресацию изолированной связи, которая не соединяется с маршрутизатором. Адреса *локальных связей* (Link-Local) имеют формат:

1111111010 (10 бит)	00...00	Уникальный адрес технологии связи

Для локальной сети адрес имеет вид:

1111111010	00...00	MAC-адрес локальной сети

Адреса локальных связей весьма полезны во время инициализации.

22.4.7 Адреса локальных сайтов

Если сайт имеет маршрутизаторы, но не связан с провайдером, можно автоматически генерировать внутренние адреса в виде:

1111111011 (10 бит)	00...00	Идентификатор подсети	Уникальный адрес технологии связи (например, MAC-адрес локальной сети)

Для такой связи префикс предоставляют маршрутизаторы (включая идентификатор подсети).

От этого формата очень легко перейти к соединению с провайдером. Маршрутизатор просто конфигурируется с новым префиксом, который содержит идентификаторы регистратора адреса, провайдера и подписчика вместе с номером подсети. Маршрутизатор предоставляет новый префикс, а хосты начинают им пользоваться. Назначенная в сайте часть адреса не меняется.

22.4.8 Формат многоадресной рассылки

Многоадресные рассылки в версии 6 имеют более четкое и гибкое определение, чем в версии 4. Введено множество типов таких рассылок. Они немного различаются в зависимости от своих свойств: постоянный адрес (permanent), кратковременный (transient), локальный (local) или глобальный (global). Многоадресные рассылки имеют следующий формат:

8 бит	4	4	112 бит
11111111	000T	Вложенность	Идентификатор группы

T = 0 для общеизвестного постоянного адреса многоадресной рассылки.

T = 1 для кратковременного адреса многоадресной рассылки.

Коды вложенности указывают, находится ли область действия в пределах того же узла, локальной связи, локального сайта, данной организации или это глобальная структура. Область действия внутри узла охватывает и случай, когда клиент посылает сообщение многоадресной рассылки на сервер, находящийся в том же самом хосте. Определены следующие коды вложенности:

0 *зарезервировано*

1 локальный узел

2 локальная связь

3 *не присвоено*

4 *не присвоено*

5 локальный сайт

6 *не присвоено*

7 *не присвоено*

8 внутри организации

9 *не присвоено*

A *не присвоено*

B *не присвоено*

C *не присвоено*

D *не присвоено*

E глобально

F *зарезервировано*

22.4.9 Нескольколадресные рассылки

Предложен новый (и экспериментальный) вид адресации — *нескольколадресные рассылки* (anycast). Адрес в таких рассылках соответствует нескольким одноадресным рассылкам, присвоенным в нескольких сетевых интерфейсах. Первоначально нескольколадресные рассылки могут быть назначены только маршрутизаторам. Нескольколадресная рассылка может выделять:

- Все маршрутизаторы, принадлежащие провайдеру
- Все маршрутизаторы на границе данной автономной системы
- Все маршрутизаторы отдельной локальной сети

Адреса таких рассылок могут быть включены в маршрут от источника, что будет означать: "Использовать ближайший маршрутизатор, который доступен по данной нескольколадресной рассылке". Например, если адреса такой рассылки идентифицируют маршрутизаторы провайдера, можно указать: "Добраться к этому провайдеру по самому короткому пути".

Интерфейс маршрутизатора, который был назначен в нескольколадресной рассылке, также имеет собственный реальный адрес.

22.5 Специальные адреса

Существует несколько форматов специальных адресов IPv6.

22.5.1 Неспецифицированные адреса

Адреса со всеми нулями

0:0:0:0:0:0:0:0

означают "неспецифицированные адреса" (unspecified address). Они иногда используются как адрес источника во время инициализации, когда система еще не знает собственного адреса.

22.5.2 Кольцевые адреса версии 6

Кольцевые адреса (loopback) в версии 6 определены как:

0:0:0:0:0:0:0:1

22.5.3 Адреса версии 4

В смешанном окружении адресов версий 4 и 6 IP-адреса систем версии 4, которые *не поддерживают* версию 6, отображаются в адреса версии 6 следующим образом:

0:0:0:0:0:FFFF:a.b.c.d

где a.b.c.d — исходный IP-адрес.

22.5.4 Взаимодействие адресов версии 6 с сетями версии 4

Еще один специальный формат используется узлами версии 6, которые связываются друг с другом через промежуточные сети версии 4 (это называется туннелями IPv4). Как показано на рис. 22.1, интерфейсам на границах должны быть присвоены адреса версии 4. Они будут преобразованы в специальный формат *совместимости IPv4* — IPv6:

0:0:0:0:0:0:a.b.c.d

Таким образом, эти адреса легко отображаются между их представлениями в версиях 4 и 6.

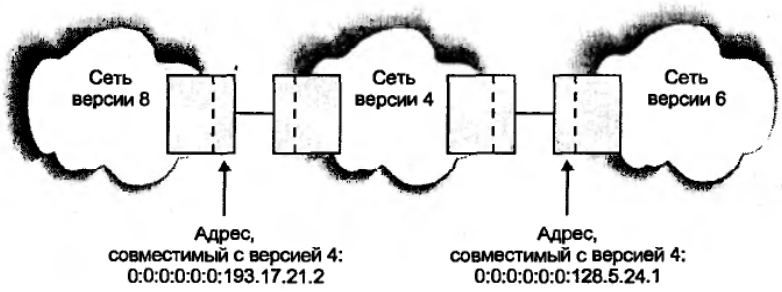

Рис. 22.1. Совместимость адресов IPv4 и IPv6

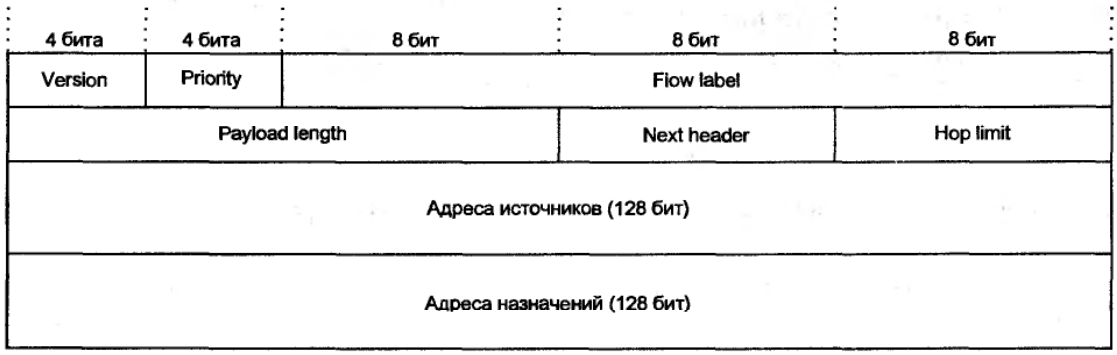

Рис. 22.2. Формат заголовка IPv6

22.6 Формат заголовка IPv6

Основной заголовок очень прост (см. рис. 22.2) и имеет немного полей:

Version	Версия. Равна 6 для IP следующего поколения.
Priority	Приоритет. Дифференцирует конкретное взаимодействие из общего трафика или определяет последовательность отбрасывания во время перегрузки.
Payload length	Длина полезной нагрузки. 16 бит. Если длина меньше или равна 64 Кбит, это поле сообщает о длине части пакета, следующей за начальным заголовком IPv6. Если длина больше 64 Кбит, длина полезной нагрузки указывается как нулевая, а фактическая длина будет сообщена *Jumbo Payload* (гигантская полезная нагрузка) в следующем далее заголовке.
Hop limit	Предел для попаданий. Уменьшается на 1 в каждом маршрутизаторе. Пакет будет отброшен, когда значение достигнет нуля.
Next header	Следующий заголовок. Идентифицирует тип следующего далее заголовка протокола (например, 6 для заголовка TCP).
Flow label	Метка потока. Указывает на трафик со специальными свойствами (например, видео в реальном времени).

22.6.1 Приоритет

Поле Priority выполняет две функции. При управлении нагрузкой для трафика TCP большим номерам соответствуют управляющие пакеты и интерактивный трафик, а меньшим номерам — обычный трафик. Определены следующие значения:

0 Трафик не специфицирован

1 Заполняющий трафик (например, сетевые новости)

2 Неважная пересылка данных (например, электронная почта)

3 Зарезервировано

4 Важный мощный трафик (например, пересылка файлов)

5 Зарезервировано

6 Интерактивный трафик (например, *telnet*)

7 Управляющий трафик Интернета (например, протоколы маршрутизации)

IPv6 может переносить трафик ISO, DECnet и т.д. Приоритеты от 0 до 7 могут использоваться для любого протокола, который предполагает собственное управление потоком.

Приоритеты от 8 до 15 используются как средство для управления перегрузками, когда протокол (например, UDP или IPX) не имеет собственных возможностей для этого. Когда сеть перегружена, трафик отбрасывается, что может оказать вредное влияние на некоторые типы прикладных данных. Малые значения (8 или 9) подразумевают большую вероятность того, что пакет будет отброшен.

22.6.2 Использование меток потока

Поток — это последовательность пакетов от источника до точки назначения, требующая специального обслуживания. Например, обработка аудио или видео в реальном масштабе времени отличается от обработки обычных данных.

Метка потока идентифицирует трафик со специальным механизмом обработки (например, резервирования определенной полосы пропускания).

Принадлежность пакета потоку обозначается ненулевой меткой потока. Пакеты одного потока имеют одинаковые адреса источника и назначения, приоритеты и метки потока.

22.7 Дополнительные заголовки IPv6

Использование дополнительных заголовков (extension header) — это прогрессивная идея, позволяющая последовательно добавлять в IP версии 6 новые функциональные возможности.

Напомним, что в заголовке IP версии 4 поле *протокола* служит для идентификации типа заголовка (например, TCP или UDP), следующего за заголовком IP. Версия 6 использует более общее поле *Next Header*. Если следующий заголовок определен как TCP или UDP, значением поля будет 6 или 17.

Между заголовком IPv6 и заголовком верхнего уровня можно вставить несколько дополнительных заголовков для необязательных вариантов, подобных маршрутизации от источника или поддержке безопасности. Фрагментация также может быть перенесена в дополнительные заголовки.

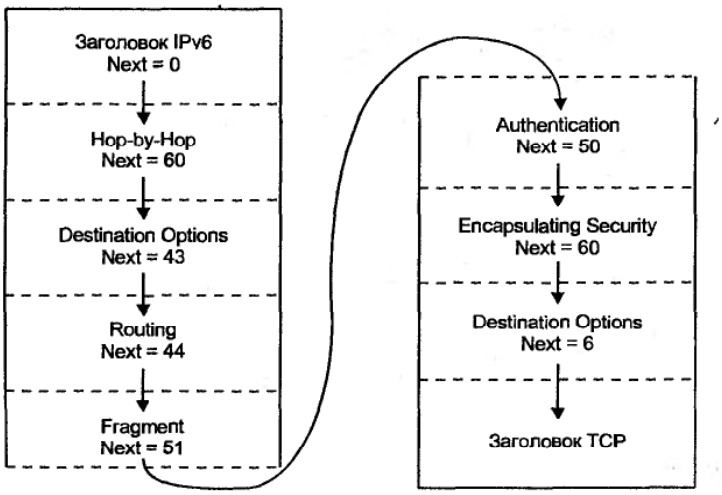

Рис. 22.3. Дополнительные заголовки

Как показано на рис. 22.3, каждый дополнительный заголовок имеет поле Next Header, что позволяет связать все эти заголовки в цепочку. Протокол следующего уровня идентифицируется включенным в общую последовательность дополнительным заголовком.

Такая схема обеспечивает большую гибкость. По мере необходимости могут определяться новые возможности, поскольку нет ограничений на общую длину. Отметим, что заключительный дополнительный заголовок может ссылаться на заголовок полностью независимого протокола, например ISO или DECnet.

Определенные на настоящий момент заголовки представлены в таблице 22.2. Некоторые из них содержат информацию, которую следует обрабатывать на каждом узле по пути следования пакета, в то время как другие заголовки обрабатываются только в точке назначения.

Показанный на рис. 22.3 порядок отражает рекомендации по размещению заголовков. Отметим, что могут присутствовать два заголовка *Destination Options*. Первый из них стоит перед *Routing* и применяется к каждому из попаданий, указанных в заголовке *Routing*. Второй присутствует как последний заголовок и применяется только в точке назначения.

Таблица 22.2

Заголовки IPv6

Заголовок	Номер из предыдущего поля Next Header
Hop-by-Hop Options (варианты "от попадания к попаданию")	0
Destination Options (варианты для точки назначения)	60
Routing (маршрутизация)	43
Fragment (фрагмент)	44
Authentication (аутентификация)	51
Encapsulating Security Payload (инкапсуляция с обеспечением безопасности полезной нагрузки)	50
No Next Header (следующего заголовка не существует)	59

Возможно, потребуется отправить пакет, состоящий из одних заголовков и не несущий полезной информации. В этом случае заключительное поле Next Header будет равно 59, что означает "дальше данных нет".

22.7.1 Использование заголовка Routing

Заголовок *Routing* выполняет очень важную функцию в версии 6. В комбинации с несколькоадресной рассылкой его можно применять для управления путем следования пакета на основе предварительных предположений или для указания специализированных провайдеров (например, обращения к мобильному пользователю). Несколькоадресные рассылки позволяют указать вариант доставки "Перейти к ближайшему маршрутизатору провайдера X".

Когда применяется заголовок Routing, точка назначения может вернуть полученный пакет назад к источнику по тому же самому пути.

22.7.2 Операции с заголовком Routing

В заголовке Routing содержится поле типа, обеспечивающее добавление в будущем различных типов для данного заголовка. В настоящий момент определен только тип 0, аналогичный маршрутизации от источника в IPv4.

Формат заголовка Routing типа 0 представлен на рис. 22.4. В таком заголовке содержится список узлов, которые нужно пройти по пути в точку назначения.

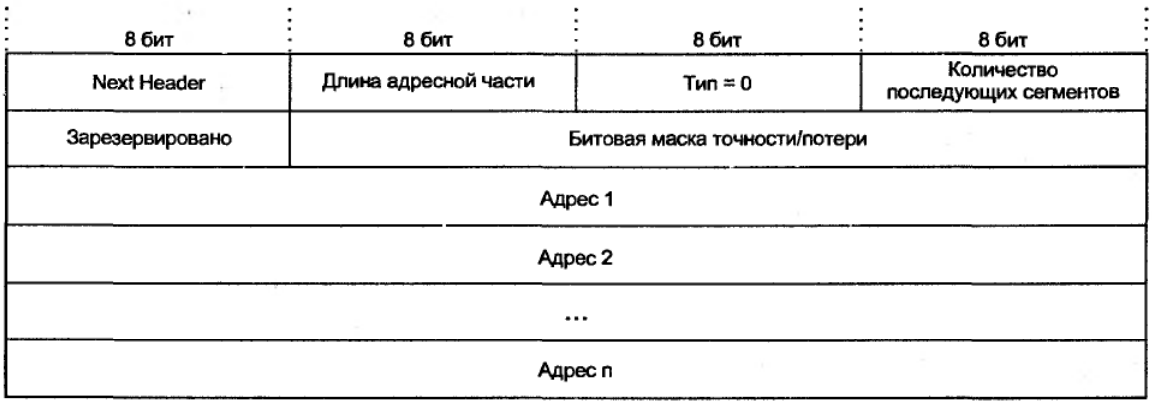

Рис. 22.4. Заголовок Routing типа 0

Как и в IP версии 4, конечной точкой назначения является "*Адрес n*". Сначала пакет направляется на адрес из главного заголовка IPv6. Затем обращаются к заголовку *Routing* и меняют местами "Адрес 1" с адресом назначения из заголовка IPv6. Счетчик количества *последующих сегментов* уменьшается на 1, а пакет направляется дальше. Последний адрес из заголовка *Routing* определяет реальную точку назначения, по прибытии в которую адресный список будет содержать адреса всех узлов, через которые прошел пакет.

Битовая маска точности/потери указывает, было ли соответствующее попадание соседним (strict) или нет (loose).

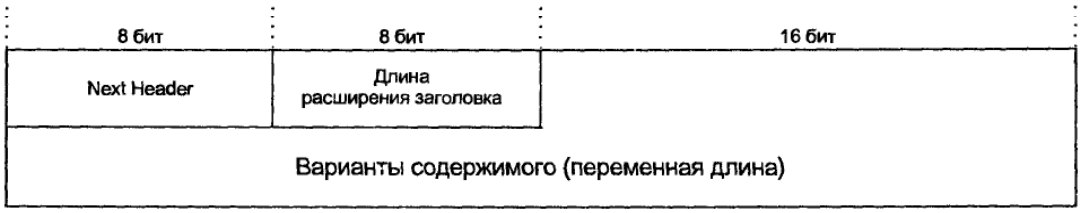

Рис. 22.5. Заголовок Hop-by-Hop

22.7.3 Дополнительный заголовок Hop-by-Hop

Заголовок Hop-by-Hop переносит информацию, которая должна проверяться на каждом участке попадания по пути следования пакета. Формат этого заголовка показан на рис. 22.5.

Заголовок Hop-by-Hop может выполнять различные функции. Каждая из них самоидентифицируется и кодируется тремя полями:

Тип варианта 8 бит	Длина варианта 8 бит	Содержимое n бит

Jumbo Payload (гигантская нагрузка) является одним из вариантов Hop-by-Hop. Он используется для декларирования полезной нагрузки, превышающей 64 Кбит. Длина содержимого (в октетах) задается 4-байтным значением. Указанная длина полезной нагрузки учитывает весь пакет, за исключением заголовка IPv6.

22.7.4 Фрагментация

В отличие от версии 4 *фрагментация никогда не выполняется маршрутизаторами*, а только в исходном узле. Ее следует по возможности избегать, хотя и допустимо пользоваться этим способом. Фрагментировать пакет должен узел-источник, а сборка пакета будет выполнена в узле назначения.

Если маршрутизатор получает слишком большой для пересылки пакет, он отбрасывает его и отсылает назад сообщение ICMP, анонсирующее максимальный пересылаемый элемент (MTU) для участка следующего попадания.

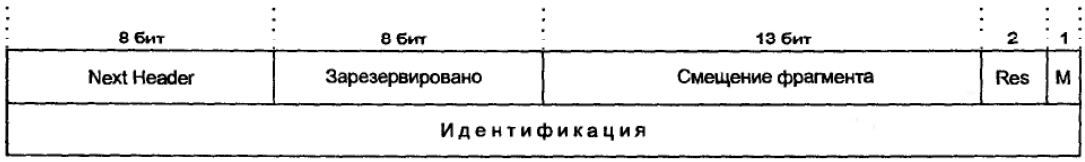

Рис. 22.6. Формат заголовка Fragmentation

Когда в узле источника создается фрагмент, в пакет включается заголовок *Fragmentation*, формат которого показан на рис. 22.6.

Как и в версии 4, поле смещения имеет длину 13 бит и измеряет *смещение фрагмента* в 8-октетных блоках. Бит M (more — *больше*) указывает, является ли этот фрагмент последним. Поле *идентификации* расширено до 32 бит.

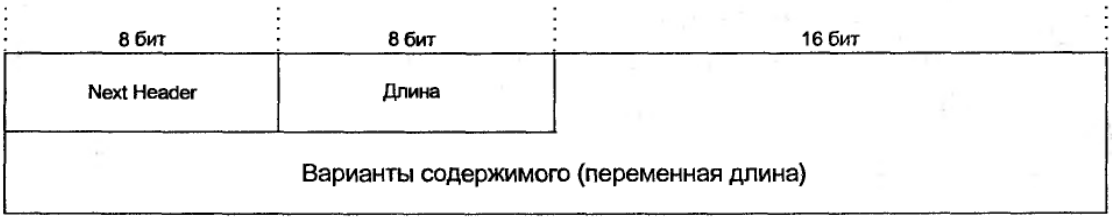

Рис. 22.7. Формат заголовка Destination Option

22.7.5 Варианты Destination

Заголовок *Destination Option* (варианты точки назначения) обеспечивает сведения о точке (точках) назначения для многоадресной рассылки. В настоящее время для этого заголовка не специфицировано никаких вариантов, кроме полей заполнения. Формат заголовка показан на рис. 22.7.

Если в пакете есть заголовок *Routing*, то могут присутствовать два заголовка *Destination Option*. Первый из них (расположен до *Routing*) содержит варианты, рассматриваемые каждым узлом, перечисленным в заголовке. Второй заголовок стоит после всех остальных и применяется только в точке назначения.

22.8 Автоконфигурация в версии 6

Раньше сети IP требовали большой работы по конфигурации и обслуживанию. Одним из преимуществ версии 6 является обеспечение эффективной автоматической инициализации. Очень важно помочь в переводе сайтов на новый формат адресов. Кроме того, не менее важно автоматизировать изменение адресов, связанное с изменениями у провайдера.

В независимых локальных сетях хосты IPv6 могут автоматически строить адреса IP на основе адреса адаптера сетевого интерфейса или иных уникальных идентификаторов на уровне связи данных.

Когда организации потребуется маршрутизировать данные или подключиться к провайдеру, маршрутизаторы обеспечат хосты всей необходимой информацией для автоконфигурации своих адресов и начала работы в сети.

22.8.1 Назначение маршрутизаторов

Каждый маршрутизатор предоставляет хостам следующие сведения:

- Свой адрес
- Список всех адресных префиксов, используемых связью
- Конкретный префикс, который должен использоваться хостом для создания своего адреса
- Предполагаемое максимальное значение предела счетчика попаданий
- Будет ли хост извлекать конфигурационную информацию из сервера DHCP
- MTU для связи, где возможны различные значения MTU
- Значения для различных таймеров

22.8.2 Список адресных префиксов

Многие сетевые администраторы с удовлетворением услышат о кончине ненавистных масок подсети. В версии 6 выбор маршрута выполняется на основе сравнения адресных префиксов.

Маршрутизатор объявляет список адресных префиксов локальной связи. Префикс — это весь адрес IPv6 или его часть вместе с номером, указывающим на реальное количество бит в этом префиксе. Хосты хранят списки префиксов.

Когда хосту нужно выяснить, находится ли точка назначения на данной связи, он просматривает свой список префиксов этой связи и сравнивает необходимое число бит с соответствующими битами адреса назначения.

22.8.3 Адреса интерфейсов IPv6

Каждый интерфейс версии 6 имеет *список* соответствующих ему адресов. Как минимум, список содержит уникальный *адрес локальной связи* (link local address), имеющий формат:

1111111010 (10 бит)	00...00	Уникальный адрес технологии связи

Каждому узлу необходим способ генерации собственных уникальных адресов интерфейса связи. Например, интерфейс локальной сети может иметь уникальной частью адреса собственный MAC-адрес, размещенный в крайних правых 48 битах. Системы могут взаимодействовать по связи через адреса локальной связи.

Как хост автоматически генерирует глобальные адреса и адреса локального сайта? Маршрутизатор объявляет список префиксов. Некоторые префиксы предназначены для конструирования адресов хостов. Новые адреса локальных сайтов и глобальные адреса создаются из предложенного маршрутизатором префикса и следующего далее уникального адреса связи. Полученный адрес добавляется в список хоста.

Предложенные маршрутизатором сведения также указывают хостам место извлечения дополнительной адресной информации от сервера DHCP (который может присваивать адреса, конфигурируемые администратором сети). Там же указывается и способ извлечения этих сведений.

Кроме того, в версии 6 остается ручное конфигурирование адресов (если оно кому-нибудь понадобится).

22.8.4 Изменение адресов

Возможность применения более одного глобального префикса упрощает переход от одного провайдера к другому.

От маршрутизатора поступают значения для установки индивидуальных таймеров на каждый префикс провайдера. При переключении с одного провайдера на другой старый префикс просто устраняется по истечении срока действия. Значения тайм-аута для нового активного префикса периодически обновляются, чтобы исключить их удаление при переполнении таймера.

Тайм-ауты дают возможность выбрать хост и подключить его к другой связи данного сайта. Префикс содержит идентификатор подсети и сведения о провайдере и регионе, поэтому при устранении старого префикса по тайм-ауту сразу становится доступен новый префикс.

22.8.5 Тестирование уникальности адреса

Перед использованием адреса локальной связи хост должен проверить его уникальность с помощью многоадресного запроса. Это позволит обеспечить уникальность IP-адреса, а также всех адресов, созданных из этого адреса с помощью различных префиксов. Адреса, конфигурируемые вручную или получаемые от сервера DHCP, также проверяются на уникальность перед началом использования.

22.9 Конфигурирование через DHCPv6

Система может получить полный набор конфигурационных параметров от сервера DHCP. Для перехода на DHCP версии 6 нужны некоторые изменения.

Новый протокол DHCP должен поддерживать адреса версии 6. Кроме того, старый тайм-аут выделения адреса нужно заменить как утративший смысл и назначение.

Интересно, что DHCPv6 позволяет не только автоконфигурировать хосты, но и проводить автоматическую регистрацию имен хостов и их адресов в DNS. Инициализируемый хост может запросить применение определенного имени или разрешить присвоить себе имя сервера DHCPv6.

Если у клиента завершается время использования адреса, сервер DHCPv6 удалит запись об этом клиенте в DNS.

22.10 Переход на IPv6

IP широко распространен во всем мире. Однако нельзя требовать, что бы все одновременно перешли на версию 6. Этот переход должен быть постепенным:

- Узлы версии 6 должны взаимодействовать с узлами версии 4.
- От организаций нельзя требовать отказа от их текущих адресов.

- Организации должны иметь возможность модернизировать отдельные узлы, оставляя другие без изменения.
- Переход должен быть прост и понятен.

22.10.1 Необходимость изменений

Провайдерам IPv6 необходим для более эффективной магистральной маршрутизации и увеличения количества своих подписчиков. Однако зачем переходить на версию 6 независимым организациям, у которых прекрасно работают сети на старой системе адресации? Если нет проблем с обслуживанием IP-адресов или введением новых служб (например, потоков), то переходить на новую версию необязательно.

Серверы Интернета могут одновременно работать с двумя стеками и двумя системами адресации еще очень долгое время. Однако в некоторый момент станет удобнее пользоваться версией 6, чем игнорировать ее.

22.10.2 Путь перехода на новую версию

Первым шагом на пути к версии 6 будет модернизация программного обеспечения сервера имен доменов сайта, чтобы сервер DNS смог отвечать на запросы, используя новый формат адресов.

Вероятно, первыми модернизируемыми системами станут маршрутизаторы интерфейсов с внешними сетями. Эти маршрутизаторы будут преобразованы для совместной работы как с версией 4, так и с версией 6. Постепенно на наиболее важных маршрутизаторах появится стек протоколов версии 6. В смешанном окружении трафик версии 6 будет пересылаться по тоннелям в сетях версии 4.

В переходный период будут применяться адреса локальных сайтов IPv6. При подключении сайтов к провайдеру на них появятся сведения о префиксах региона, провайдера и подписчиков.

22.10.3 Изменения в DNS

Новый тип записи о ресурсе, *AAAA*, отображает имена доменов в адреса IP версии 6. Пример такой записи:

```
MICKEY IN AAAA 4321:0:1:2:3:4:567:89AB
```

Должен быть обеспечен и обратный просмотр. Для преобразования *адресов в имена* для IPv6 потребуется добавить новые домены. Обратный поиск доменов включается в дерево доменов от узла *IP6.INT*.

Адреса IP версии 4 рассматриваются в обратном порядке, чтобы получить свои метки в домене *in-addr.arpa*. Адреса версии 6 также просматриваются наоборот и переписываются как ряд шестнадцатеричных цифр, разделенных точками. Например, обратная запись элемента:

```
4321:0:1:2:3:4:567:89AB
```

появится в дереве домена как:

```
B.A.9.8.7.6.5.0.4.0.0.0.3.0.0.0.2.0.0.0.1.0.0.0.0.0.0.0.1.2.3.4.IP6.INT
```

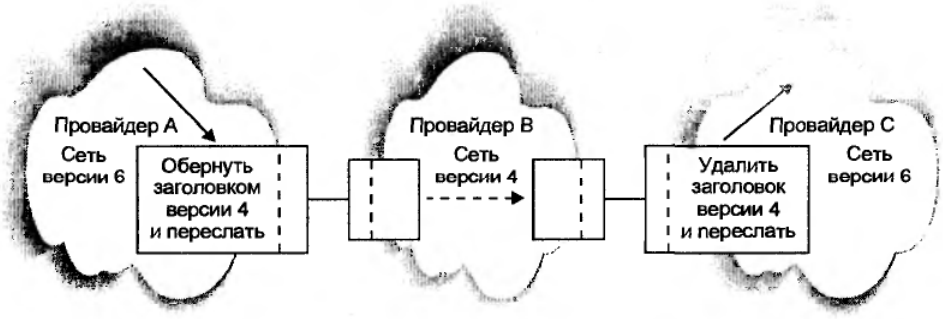

Рис. 22.8. Трафик в туннеле сети версий 4

22.10.4 Туннели через сети версии 4

В течение переходного периода датаграммы иногда будут пересекать на своем пути сети версии 4. На рис. 22.8. провайдеры A и C поддерживают версию 6, а провайдер B — нет. Граничные маршрутизаторы интерфейсов имеют адреса совместимости IPv4 с IPv6, которые легко преобразовать в адреса версии 4, удаляя нулевые префиксы. Пакеты версии 6 "обернуты" заголовком версии 4 и пересекают промежуточную сеть по туннелю.

Формирование туннеля может происходить и в пределах сайта, который преобразовал некоторые из своих сетей в версию 6. Оно может использоваться в любом удобном для этого месте: между маршрутизаторами, между хостами или на пути от хостов к маршрутизаторам.

22.11 Резюме

Рабочие группы разработки IP следующего поколения заложили основы новой версии, которая разрешает проблему истощения пространства адресов Интернета и предлагает более эффективную маршрутизацию. Новый протокол предоставляет возможности автоматической конфигурации и сосуществования со старой версией, а также позволяет осуществлять постепенный переход на новую версию. Цепочечные заголовки обеспечивают безболезненную будущую модернизацию и удобный путь перемещения в сетях IP данных других протоколов.

22.12 Дополнительная литература

RFC 1884 описывает адреса IPv6, а RFC 1883 — основы протокола версии 6. RFC 1885 посвящен ICMPv6, а RFC 1886 имеет дело с расширениями DNS. В RFC 1887 обсуждается архитектура выделения адресов. После выхода данной книги должны появиться и другие RFC.

Глава
23

ICMPv6 и исследование соседей

23.1 Введение

Версия 6 протокола Internet Control Message Protocol (ICMPv6) сохраняет многие функции версии 4, но вводит и несколько важных изменений:

- Сообщения ICMPv6 помогают в автоматической конфигурации адресов.
- Новые сообщения и процедуры ICMPv6 заменяют протокол ARP.
- Автоматизируется исследование максимального элемента пересылки (MTU) по пути. Поскольку маршрутизаторы более не фрагментируют пакеты, то в случае слишком большого размера пакетов источнику отправляется сообщение *Packet Too Big* (пакет слишком велик).
- ICMPv6 не посылает сообщений *Source Quench*.
- ICMPv6 принимает на себя функции отчета о членстве в многоадресной группе протокола Internet Group Management Protocol.
- ICMPv6 помогает определить выключение маршрутизатора или партнера по коммуникации.

ICMPv6 настолько отличается от старой версии, что ему присвоен новый номер 58 в заголовке Next Header.

23.2 Базовые сообщения ICMPv6

В таблице 23.1 перечислены основные типы сообщений ICMPv6. Отметим, что сообщениям об ошибке присвоены номера от 0 до 127, а информационным сообщениям — от 128 до 255. Общий формат сообщения ICMP показан на рис. 23.1. Сначала рассмотрим сообщения ICMP, сходные с сообщениями версии 4.

Таблица 23.1

Типы сообщений ICMP

Сообщения об ошибках	Тип
Destination Unreachable (точка назначения недоступна)	1
Packet Too Big (пакет слишком велик)	2
Time Exceeded (истекло время)	3
Parameter Problem (проблема с параметрами)	4
Информационные сообщения	Тип
Echo Request (эхо-запрос)	128
Echo Reply (эхо-ответ)	129
Group Membership Query (запрос о членстве в группе)	130
Group Membership Report (отчет о членстве в группе)	131
Group Membership Reduction (исключение из членов группы)	132



Рис. 23.1. Формат сообщения ICMPv6

23.2.1 Destination Unreachable

Причина отправки сообщения *Destination Unreachable* (точка назначения недоступна) определяется кодами:

0 Нет маршрута к точке назначения

1 Административно запрещено взаимодействие с точкой назначения

2 Следующее назначение в заголовке Routing не является соседом, но установлен бит strict.

3 Адрес недоступен

4 Порт недоступен

Формат сообщения *Destination Unreachable* показан на рис. 23.2.

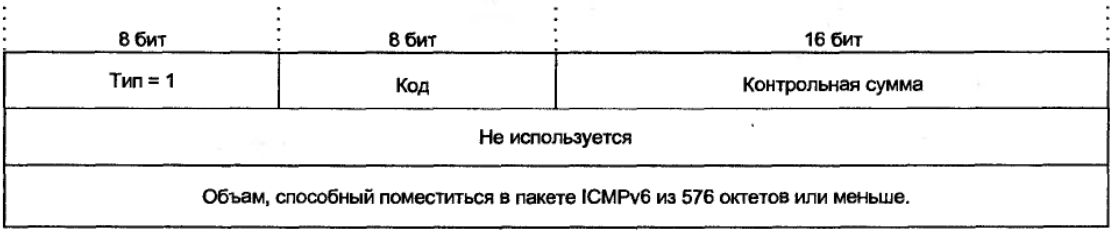

Рис. 23.2. Формат сообщения Destination Unreachable

23.2.2 Packet Too Big

Маршрутизатор посылает сообщение *Packet Too Big* (пакет слишком велик), когда пакет больше MTU связи следующего попадания. Это значение будет включено в отправляемое сообщение. В версии 4 этот же смысл имеет сообщение *Destination Unreachable*. Формат сообщения *Packet Too Big* показан на рис. 23.3.

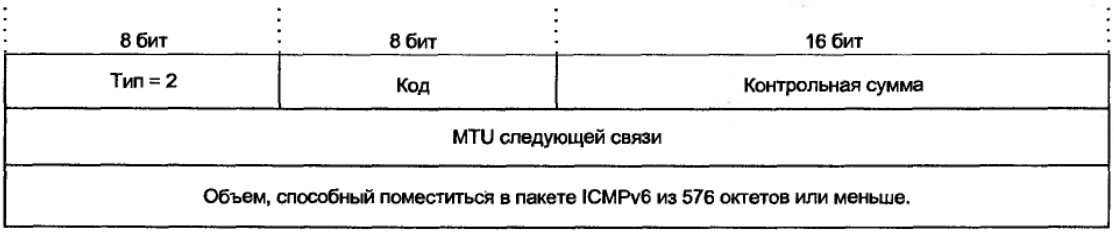

Рис. 23.3. Формат сообщения Packet Too Big

23.2.3 Time Exceeded

Сообщение *Time Exceeded* (истекло время) отправляется маршрутизатором, который уменьшил счетчик попаданий до нуля (код = 0), или системой, у которой закончилось время на сборку пакета (код = 1). Формат сообщения идентичен *Destination Unreachable*, но поле типа равно 3.

23.2.4 Parameter Problem

Сообщение *Parameter Problem* (проблема с параметрами) отправляет система, которая не может обработать пакет из-за одного из полей заголовка. Коды сообщения:

0 Неправильное количество полей заголовка

1 Нераспознанный тип в поле Next Header

2 Нераспознанный вариант IPv6

Формат сообщения идентичен *Destination Unreachable*, но неиспользованное поле занято указателем, описывающим смещение октета с ошибкой, а тип равен 4.

23.2.5 Echo Request и Echo Reply

Сообщения *Echo Request* (эхо-запрос) и *Echo Reply* (эхо-ответ) имеют формат, как и в версии 4, но для запроса используется тип = 128, а для ответа тип = 129.

23.2.6 Group Membership

Формат сообщений многоадресных рассылок Group Membership (членство в группе) показан на рис. 23.4. Он был изменен относительно версии 4 для согласования с форматом ICMPv6. Поле *Maximum Response Delay* (максимальная задержка ответа) имеет ненулевое значение только в сообщениях запросов. Оно указывает максимальное время, на которое может быть задержан ответ. В сообщении могут быть перечислены следующие типы:

130 Group Membership Query

131 Group Membership Report

132 Group Membership Reduction

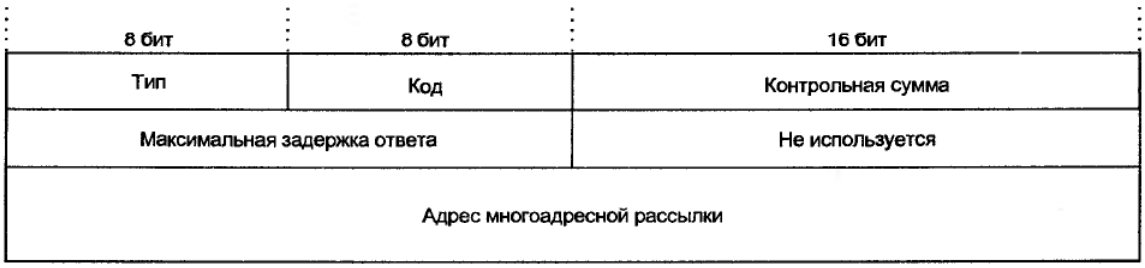

Рис. 23.4. Формат сообщений Group Membership

23.3 Исследование соседей

На момент выхода книги еще продолжалась работа над очень важным набором спецификаций для автоматизации функций связи. К ним можно отнести:

Router Discovery	Исследование маршрутизаторов. Поиск маршрутизаторов в локальной связи.
Prefix Discovery	Исследование префикса. Исследование и использование префикса точки назначения (на связи или удаленной). Используется вместо маски подсети.
Parameter Discovery	Исследование параметров. Выяснение значений параметров (например, MTU или предела по умолчанию для счетчика попаданий).
Address Autoconfiguration	Автоконфигурация адресов. Самоконфигурация адресов интерфейса связи.
Address Resolution	Разрешение адресов. Отображение IP-адреса соседа по связи на адрес уровня связи данных.

Next-hop Determination	Определение следующего попадания. Отображение IP-адреса на адрес участка следующего попадания.
Neighbor Unreachability Detection	Определение недостижимости соседа. Определение отказавшего соседнего хоста или маршрутизатора.
Duplicate Address Detection	Определение дублирования адресов. Проверка, не используется ли присваиваемый IP-адрес другой системой.
Redirect	Перенаправление. Получение уведомления, что существует лучший маршрутизатор для данной точки назначения или что точка назначения находится в локальной связи.

В таблице 23.2 перечислены сообщения ICMPv6, предлагаемые для реализации функций Neighbor Discovery.

Таблица 23.2

Сообщения ICMP для исследования соседей

Информационное сообщение	Тип
Router Solicitation Message (сообщение-ходатайство маршрутизатора)	133
Router Advertisement Message (сообщение-объявление маршрутизатора)	134
Neighbor Solicitation Message (сообщение-ходатайство соседа)	135
Neighbor Advertisement Message (сообщение-объявление соседа)	136
Redireckt (перенаправить)	137

23.3.1 Автоконфигурация через маршрутизаторы

Маршрутизаторы предоставляют хостам:

- Адресную информацию маршрутизатора
- Список всех префиксов, используемых связью
- Префиксы, которые должны применять хосты для создания собственных адресов
- Максимальный предел попаданий, который должен использоваться для путей, проходяших через этот маршрутизатор
- Указание, должны ли хосты использовать сервер загрузки для получения дополнительных конфигурационных данных
- MTU для связи, имеющей различные MTU
- Значения для различных таймеров

Все эти операции выполняются через сообщения ICMPv6 *Router Advertisement* (объявления маршрутизатора), имеющие тип 134. Хосты слушают сообщения *Router Advertisement* по всем адресам многоадресных рассылок на всех узлах локальной связи.

Когда хост загружается, он может не дождаться *Router Advertisement* и отправить сообщение *Router Solicitation* (ходатайство маршрутизатору) с типом 133, чтобы вызвать объявление от маршрутизатора. Маршрутизатор отвечает сообщением *Advertisement* по адресу локальной связи хоста.

23.3.2 Сообщения Neighbor Solicitation и Advertisement

В настоящее время предлагается заменить запросы старого протокола Address Resolution Protocol (ARP) новыми многоадресными сообщениями протокола ICMP *Neighbor Solicitation* и *Advertisement* (ходатайство и объявление соседа). Сообщение *Neighbor Advertisement* является ответом на *Neighbor Solicitation*. Кроме исследования соседей по адресам уровня связи данных, сообщение *Neighbor Solicitation* применяется для:

- Обнаружения дублированных IP-адресов
- Тестирования, является ли маршрутизатор отключенным
- Тестирования, является ли отключенным сосед, которому посылались пакеты

23.3.3 Address Resolution

Для исследования адреса соседа на уровне связи данных сообщение *Neighbor Solicitation* отправляется на специальный адрес, называемый *целевым адресом ходатайствующего узла для многоадресной рассылки* (solicited-node multicast address). Этот адрес сформирован из младших 32 бит целевого IP-адреса и предопределенного 96-разрядного префикса FF02:0:0:0:0:1. Таким способом создается адрес многоадресной рассылки, вложенный в локальную связь. Отправитель включает в сообщение собственный адрес уровня связи данных.

Использование этой специализированной многоадресной рассылки существенно сокращает количество систем, слушающих запрос. Вполне вероятно, что только целевая система будет реагировать на такой запрос.

23.3.4 Обнаружение дублирования IP-адресов

Перед использованием IP-адреса локальной связи или любого другого адреса, который *не был построен* путем добавления префикса к адресу локальной связи, узел должен послать сообщение *Neighbor Solicitation* с целью узнать, имеет ли кто-то из соседей выбранный IP-адрес. В качестве исходного адреса для сообщения узел применяет неспецифицированный адрес. Если адрес IP уже используется, его владелец пошлет ответ в многоадресной рассылке.

23.3.5 Обнаружение недостижимости соседа

Обнаружение неисправного маршрутизатора было рискованным делом в IPv4. В версии 6, когда тайм-аут указывает на бездействующий маршрутизатор, система проверяет такой маршрутизатор одноадресным сообщением *Neighbor Solicitation*.

Такая же процедура выполняется, чтобы проверить недостижимость соседнего хоста.

23.3.6 Сообщение Redirect

Как и в версии 4, когда хост пересылает датаграмму на неправильный локальный маршрутизатор, он получает обратно сообщение *Redirect* (перенаправление), указывающее правильный узел для первого попадания. Сообщение *Redirect* может использоваться для уведомления отправителя о размещении точки назначения в локальной связи. Возможно, именно поэтому сообщение *Redirect* определено в спецификации Neighbor Discovery.

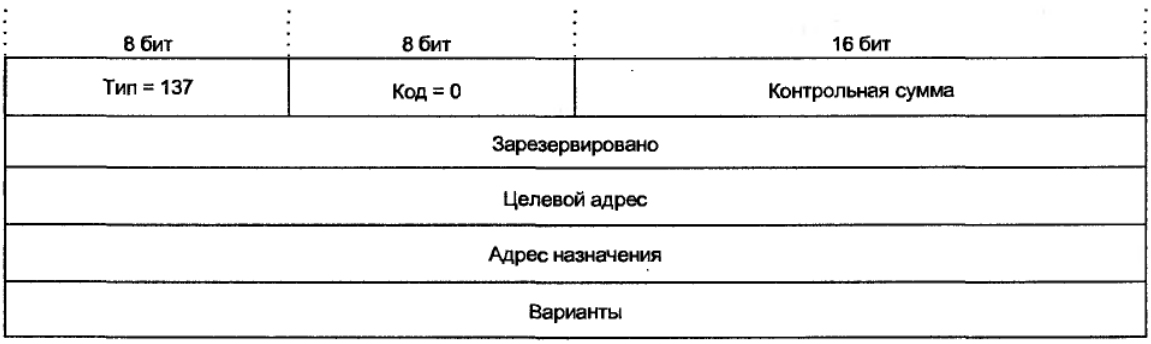

Рис. 23.5. Формат сообщения Redirect

На рис. 23.5 показан предложенный формат для сообщения *Redirect* в ICMPv6. Целевой адрес — это адрес IP следующего попадания, который должен использоваться при пересылке пакета. Адрес назначения — это выбранная точка назначения. Поле выбора содержит адрес уровня связи данных целевой системы и может также включать сведения для перенаправления датаграммы.

23.4 Дополнительная литература

ICMPv6 описан в RFC 1885. На момент выхода книги протоколы Neighbor Discovery были еще на стадии обсуждения.

Глава

24

Безопасность в IP

24.1 Введение

Необходимость разработки новой версии IP стала дополнительным стимулом для решения проблем безопасности TCP/IP. Предлагаемый механизм обеспечивает безопасность на уровне IP. Он разработан для совместимости как с версией 4, так и с версией 6. Для упрощения все сценарии этой главы предполагают использование версии 4.

Все признают необходимость средств защиты, но как обеспечить их на уровне IP? Почему не подходит уровень приложений? На практике множество приложений *реализует* собственные методы обеспечения безопасности. Однако в окружении, где очень легко "подглядеть" за проходящим трафиком и захватить его для дальнейшего анализа, или где есть возможность фальсификации IP-адресов, трудно обеспечить достоверность каждой датаграммы.

Почему не подходит физический уровень? Весь трафик связи должен шифроваться. Это позволит решить проблему с "подсматриванием", однако приведет к необходимости автоматического дешифрирования в каждом маршрутизаторе. Сегодня мы еще не можем доверять каждому маршрутизатору.

Кроме того, в этом случае не решается проблема с аутентификацией, равно как и с перегрузкой высокоскоростного трафика, когда шифрование/дешифрирование реализуется на аппаратном уровне. Более того, каждый интерфейс локальной сети должен быть способен шифровать и дешифрировать данные, а это серьезно увеличит стоимость оборудования.

24.2 Безопасность

В главе 3 мы уже отмечали три атрибута безопасности:

Аутентификация	Проверка подлинности пользователя, клиентского процесса или серверного приложения
Целостность	Проверка отсутствия изменений в данных
Конфиденциальность	Предотвращение несанкционированного доступа к информации

В той же главе представлены несколько механизмов реализации указанных атрибутов. В следующем разделе мы рассмотрим адаптацию этих механизмов для обеспечения безопасности на уровне IP.

24.3 Стратегия безопасности

Интеграция безопасности в IP стала одной из наиболее сложных работ, выполненных IETF. Аутентификация, целостность данных и конфиденциальность стали насущными и необходимыми. Стратегия безопасности предполагает:

- Содействие совместной работе, начинающейся с уже известных и реализованных механизмов для аутентификации, целостности данных и конфиденциальности.
- Разработку основ безопасности, позволяющих перейти на новые механизмы безопасности.

В качестве исходных были выбраны следующие механизмы:

- MD5 для аутентификации и целостности данных (в настоящее время проявились проблемы с MD5 при реализации высокоскоростных коммуникаций, поскольку требуется большой объем вычислений).
- Симметричное шифрование в режиме Cipher Block Chaining американского стандарта Data Encryption Standard (CBC-DES) для обеспечения конфиденциальности.

Для распространения информации используется шифрование общедоступными ключами.

24.4 Сценарии обеспечения безопасности

Существует множество способов использования различных вариантов безопасности (они описаны ниже), но сначала мы познакомимся с несколькими сценариями для разъяснения причин выбора некоторых вариантов.

Сценарий 1. Компания XYZ хочет обезопасить свои внешние коммуникации клиент/сервер. Ей нужно устранить возможность фальсификации своих данных при подделке исходного IP-адреса или изменения данных при пересылке.

Сценарий 2. Администратор компании XYZ копирует очень важные файлы между хостами. Эту операцию должен выполнять только этот администратор и никто другой. Кроме того, важно предотвратить "подглядывание", т.е. захват и несанкционированное использование данных из файлов.

Сценарий 3. Компания XYZ соединила по Интернету свои производственные подразделения с удаленным главным офисом. Она хочет скрыть все свои коммуникации от остального мира.

Для простоты можно считать, что каждый клиентский или серверный хост имеет единственный IP-адрес и один интерфейс. Однако механизмы безопасности должны работать и для систем с несколькими интерфейсами и несколькими IP-адресами.

24.4.1 Сценарий 1

Технология Message Digest (резюме сообщения) подойдет для сценария 1 — аутентифицировать отправителя и определить изменения в данных. Рассмотрим, как работает этот механизм (см. рис. 24.1):

- Источник и назначение знают секретный ключ.
- Источник выполняет вычисление, используя данные и секретный ключ.
- Источник отправляет в сообщении результат вместе с данными.
- В точке назначения выполняются те же самые вычисления и сравниваются результаты.

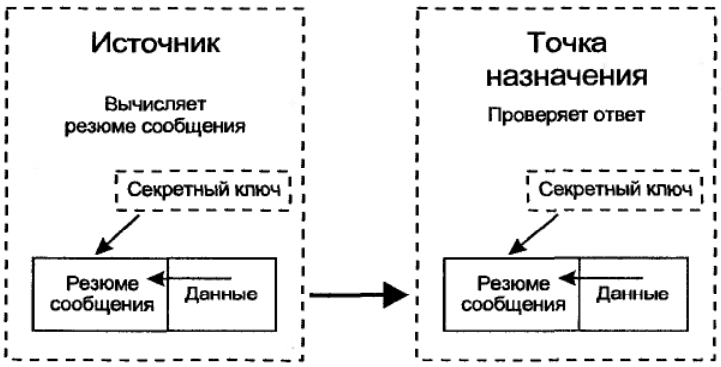

Рис. 24.1. Использование Message Digest

24.4.2 Конфигурирование аутентификационной информации для сценария 1

Предположим, что компания XYZ имеет важный сервер с IP-адресом 130.15.20.2. В рамках работ по безопасности администратор сервера нумерует хосты клиентов и присваивает секретные ключи аутентификации каждому клиентскому IP-адресу.

Серверу нужно хранить информацию о безопасности. Для этого можно воспользоваться таблицей (например, 24.1). В ней индексируются все присвоенные клиентским хостам номера, называемые *индексами параметров безопасности* (Security Parameters Index — SPI). Если сервер имеет несколько IP-адресов, таблица индексируется и по адресам точек назначения.

Таблица 24.1

Информация безопасности в точке назначения 130.15.20.2

SPI (для хоста клиента)	IP-адрес источника	Ключ аутентификации клиента	Метод аутентификации клиента
301	130.15.24.4	X'2E-41-43-11-5A-5A-74-53-E3-01-88-55-10-15-CD-23	MD5
302	**130.15.60.10**	**X'35-14-4F-21-2B-2C-12-34-82-22-98-44-C0-1C-33-56**	**MD5**
...	...	...	...

Конечно, каждый клиент должен быть конфигурирован с SPI и секретным ключом, используемым при доступе к серверу. Таблица 24.2 показывает конфигурационные данные второго клиента. Клиент нуждается в отдельных вхождениях для каждой точки назначения, к которой будет обращаться.

Таблица 24.2

Информация безопасности в источнике 130.15.60.10

IP-адрес назначения	SPI	IP-адрес источника	Ключ аутентификации клиента	Метод аутентификации клиента
130.15.20.2	**302**	**130.15.60.10**	**X'35-14-4F-21-2B-2C-12-34-82-22-98-44-C0-1C-33-56**	**MD5**
130.15.65.4	...	...	...	...

Что происходит, когда клиентский хост хочет послать аутентифицированную датаграмму серверу?

- Клиент выбирает IP-адрес точки назначения из своей таблицы.
- Для вычисления резюме датаграммы используется ключ аутентификации.
- Номер SPI и резюме сообщения помещаются в заголовок Authentication
- Датаграмма отсылается.

При получении сервером датаграммы выполняются следующие действия:

- Сервер использует SPI из заголовка Authentication, чтобы найти в таблице запись для клиента.
- IP-адрес источника сообщения сравнивается с адресом источника в таблице.
- По ключу аутентификации из таблицы вычисляется резюме сообщения.
- Результат сравнивается со значением из заголовка Authentication.

24.4.3 Односторонняя безопасность

На данный момент выполнена только половина работы. Мы установили аутентификацию *только для одного направления* обмена данными. Аутентифицируются только датаграммы, отправляемые от клиента на сервер.

Такой метод называется *односторонней ассоциацией безопасности* (Security Association). Для идентификации используемого элемента таблицы важно обеспечить *ассоциацию* комбинации IP-адреса *назначения* и SPI как в источнике, так и в точке назначения, т.е. ассоциация безопасности связана с точкой назначения и с SPI.

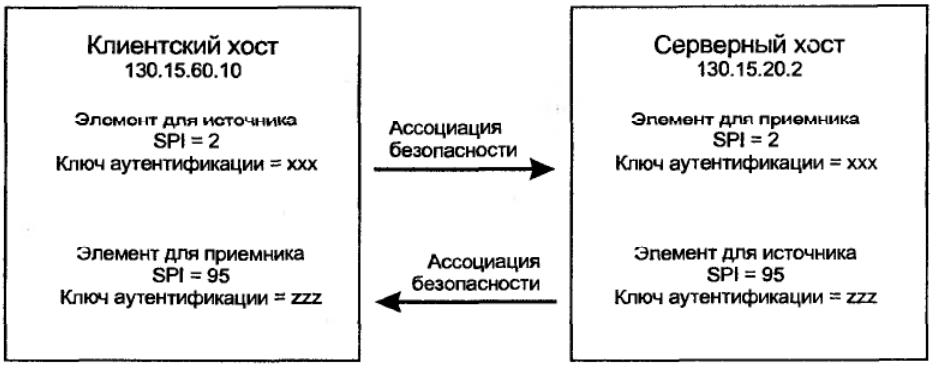

Рис. 24.2. Пары ассоциаций безопасности

Для аутентификации потока данных от сервера к клиенту *необходим отдельный набор элементов таблицы, определяющих ключи аутентификации для ассоциации безопасности и для обратного направления обмена.* Поэтому каждый хост должен:

- Иметь таблицу безопасности, когда он является источником датаграммы
- Иметь таблицу безопасности, когда он является получателем датаграммы

На рис. 24.2 показаны пары ассоциаций безопасности.

24.4.4 Количество ключей аутентификации

Сколько ключей аутентификации нужно для работы сервера с клиентами? Может показаться, что серверу достаточно иметь один ключ MD5, с помощью которого он может сказать: "Я тот самый сервер".

Однако этот ключ будут знать все клиенты. Один из них сможет фальсифицировать IP-адрес и замаскироваться под сервер. Для предотвращения такого варианта каждому клиентскому хосту следует присвоить отдельные ключи аутентификации. Общее количество ключей можно сократить, если использовать одинаковые ключи аутентификации для взаимодействий сервер-клиент и клиент-сервер.



Рис. 24.3. Несколько ассоциаций безопасности для клиента и сервера

24.4.5 Сценарий 2

В сценарии 1 безопасность реализована на уровне хостов. Но предположим, что имеется пользователь или роль, требующие другого уровня безопасности. Основы безопасности должны обеспечиваться на уровнях пользователя, роли и важной информации.

Допустим, что хост клиента из сценария 1 является многопользовательской системой. В сценарии 2 для обычных пользователей клиентского хоста 130.15.60.10 важны ключи аутентификации, совместно используемые на одном хосте. Администратору системы, выполняющему пересылку файлов на сервер, потребуются специальная аутентификация и шифрование. Создаваемые для этого ассоциации безопасности показаны на рис. 24.3.

Таблица 24.3

Информация безопасности об источнике в 130.15.20.2

SPI	IP-адрес источника	Клиентский ключ аутентификации	Клиентский метод аутентификации	Клиентский ключ шифрования	Клиентский метод шифрования
301	130.15.24.4	...	MD5	Нет	Нет
2	**130.15.60.10**	**..xxx..**	**MD5**	**Нет**	**Нет**
72	**130.15.60.10**	**..JJJ..**	**MD5**	**#$BB7&%**	**CBC-DES**
...	...	...	...	...	...

Рассмотрим таблицы ассоциаций безопасности с добавленными в них дополнительными элементами для администратора и ключами шифрования. В таблице 24.3 приведена информация о приемнике на сервере, а в таблице 24.4 — об источнике у клиента. Появились отдельные новые SPI для исходных пользователей 130.15.60.10 и для администратора, находящегося по тому же адресу.

Таблица 24.4

Информация безопасности об источнике в 130.15.60.10

IP-адрес назначения	Роль или идентификатор пользователя	SPI	IP-адрес источника	Клиентский ключ аутентификации	Клиентский метод аутентификации	Клиентский ключ шифрования	Клиентский метод шифрования
130.15.20.2	**Хост**	**2**	**130.15.60.10**	**..xxx..**	**MD5**	**Нет**	**Нет**
130.15.20.2	**Администратор**	**72**	**130.15.60.10**	**..JJJ..**	**MD5**	**#$BB7&%**	**CBC-DES**
130.15.65.4	Хост	...	...	...	MD5		...
...	...	...	...	...	...	...	

Таблицы 24.3 и 24.4 включают параметры безопасности для однонаправленных ассоциаций безопасности с источником, находящимся в клиенте 130.15.60.10, и точкой назначения на сервере 130.15.20.2. Отдельный набор параметров должен быть определен для обратного направления — от сервера-источника к клиенту-приемнику. Здесь снова разработчики конкретной системы должны решить, использовать ли те же самые ключи в обоих направлениях или присвоить различные ключи для трафиков клиент-сервер и сервер-клиент.

24.4.6 Сценарий 3

Сценарий 3 показан на рис. 24.4. Цель состоит в том, чтобы сделать невидимым для внешнего мира весь трафик, который компания XYZ посылает через недоверенную сеть. Для этого используется инкапсуляция в *режиме туннеля*, т.е. датаграммы шифруются и инкапсулируются внутри других датаграмм.

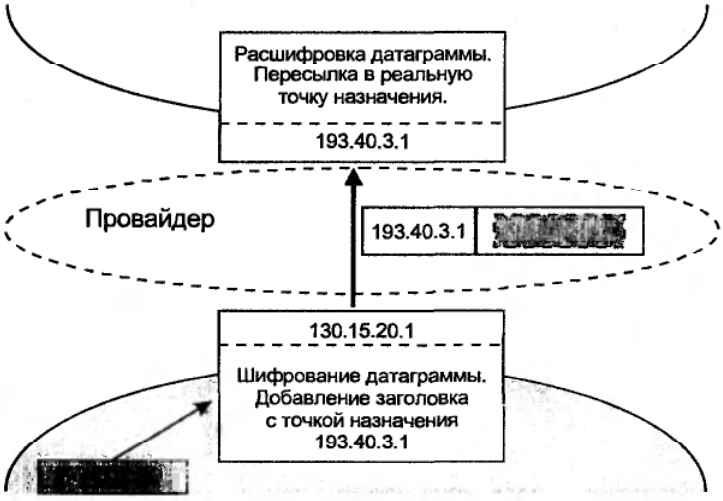

Рис. 24.4. Трафик в туннеле между двумя сетями

Когда датаграмма с точкой назначения в сети 193.40.3 достигает граничного маршрутизатора для сети 130.15, он зашифровывает всю эту датаграмму, включая заголовки. Он подставляет временный (открытым текстом) IP-заголовок и пересылает датаграмму через сеть провайдера на граничный маршрутизатор сети 193.40.3. Можно подставить и другие заголовки (например, отдельный заголовок аутентификации для взаимодействия маршрутизатор-маршрутизатор). В маршрутизаторе-приемнике временный заголовок удаляется, датаграмма дешифрируется и отправляется в истинную точку назначения. В данном случае ассоциации безопасности устанавливаются между двумя граничными маршрутизаторами.

24.4.7 Обобщение

Мы рассмотрели некоторые конкретные примеры, чтобы познакомится с основной структурой безопасности. В целом можно использовать обычный набор механизмов для защиты пересылаемого трафика:

- Между хостами
- Между маршрутизаторами
- Между хостом и маршрутизатором
- Между маршрутизатором и хостом

Если хост назначения имеет более одного IP-адреса, следует установить отдельные параметры ассоциации безопасности для каждого адреса. Не существует никаких ограничений на реализацию аутентификации, целостности данных и конфиденциальности.

В сценарии 2 безопасность определена для уровней пользователя и роли. При необходимости можно еще глубже структурировать безопасность. Более того, параметры безопасности могут конфигурироваться на основе важности информации (например, "не секретно" или "совершенно секретно"). Обслуживание множества различных параметров следует переложить на хорошее приложение.

24.5 Элементы протокола безопасности

Рассмотрим реализацию безопасности более детально.

24.5.1 Ассоциации безопасности

Безопасность одновременно управляет только одним направлением обмена. Для обеспечения безопасности коммуникации источника и получателя каждый из них должен хранить набор параметров, например:

- Адрес источника
- Используемый алгоритм аутентификации и целостности данных
- Используемый алгоритм конфиденциальности
- Секретные ключи и другую необходимую для алгоритмов информацию
- Ограничения по времени на ключи
- Ограничение по времени на ассоциации безопасности
- Гриф секретности (например, "не секретно" или "совершенно секретно")

Ассоциация безопасности формально определяется как набор защищенных параметров, которые поддерживают безопасность однонаправленного взаимодействия между источником и приемником. Из приведенных выше сценариев видно, что:

- Хост источника может применять один набор параметров для пересылки данных в точки назначения.
- Хост может использовать несколько ассоциаций безопасности для различных хостов точек назначения. Ассоциации выбираются на основе идентификатора пользователя, роли или важности информации.

Каждому набору параметров безопасности в каждой из точек назначения присваивается цифровой идентификатор, называемый индексом параметров безопасности (Securuty Parameter Index — SPI). Некоторым наборам стандартных параметров значение SPI присваивает IANA.

Одинаковые SPI могут использоваться для различных точек назначения. Наборы параметров (Назначение = A, SPI = 300) и (Назначение = B, SPI = 300), скорее всего, будут различными. Другими словами, набор параметров идентифицируется как SPI, так и точкой назначения.

Для реализации безопасности в IP версий 4 и 6 применяются заголовки *Authentication Header* и *Encapsulating Security Payload Header*.

24.5.2 Authentication Header

Если для аутентификации используется резюме сообщения, заголовок *Authentication Header* (заголовок аутентификации) выполняет две задачи:

- Проверяет отправителя, поскольку настоящий отправитель должен знать секретный ключ для вычисления резюме сообщения.
- Проверяет, не были ли данные изменены при пересылке.

Формат Authentication Header показан на рис. 24.5. Получатель использует SPI для выбора требуемого протокола и ключа аутентификации. Ключ служит для вычисления приемником резюме по алгоритму MD5.

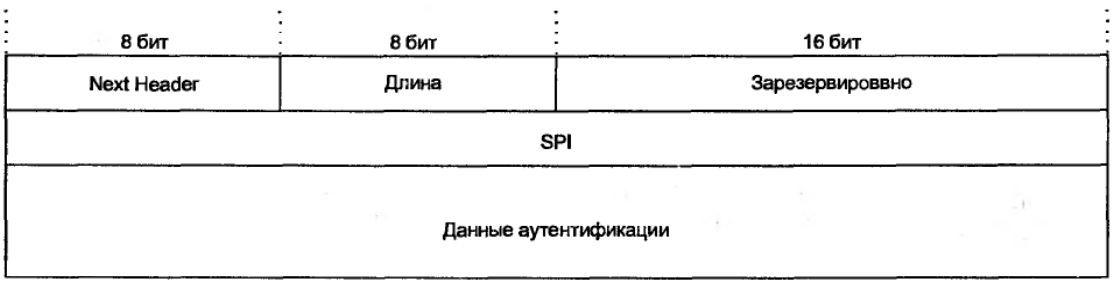

Рис. 24.5. Формат заголовка Authentication Header

Вычисление аутентификации по MD5 выполняется над всеми полями датаграммы IP, которые не изменяются при пересылке (изменяемые поля, например счетчик попаданий или указатель пути в версии 6, при вычислении трактуются как нулевые). Результат у получателя сравнивается со значением из поля *данных аутентификации*. При расхождении датаграмма отбрасывается.

24.5.3 Режимы транспорта и туннеля

Рассмотрим способы реализации конфиденциальности. Формат датаграммы IP версии 6 с шифрованной полезной нагрузкой более высокого уровня показан на рис. 24.6. Такой формат определяет *режим транспорта* (Transport-mode).

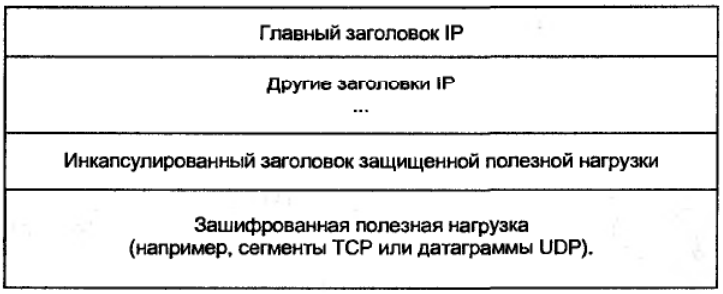

Рис. 24.6. Шифрование для режима транспорта

На рис. 24.7 показан формат для *режима туннеля* (Tunnel-mode). Шифруется вся датаграмма, включая все ее заголовки. Для пересылки подставляется новый заголовок. Режим туннеля между хостами может не сработать, если по пути следования попадутся фильтрующие маршрутизаторы со средствами защиты. Такие системы будут проверять информацию, подобную IP-адресам источника и назначения либо порты, а эти сведения будут скрыты внутри зашифрованного сообщения.

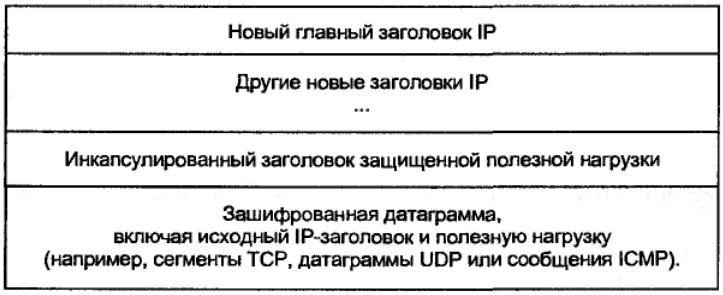

Рис. 24.7. Шифрование для режима туннеля

24.5.4 Инкапсуляция защищенной полезной нагрузки

Заголовок инкапсуляции защищенной полезной нагрузки протокола IP (IP Encapsulating Security Payload) применяется как для режима транспорта, так и для режима туннеля.

Формат этого заголовка показан на рис. 24.8. Получатель использует индекс SPI для выбора алгоритма и ключа (ключей). Оставшиеся данные зависят от выбранного алгоритма.

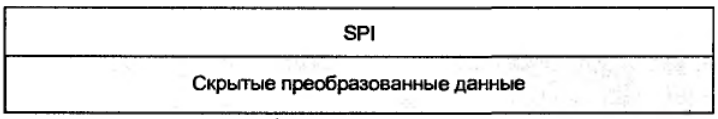

Рис. 24.8. Заголовок Encapsulating Security Payload

При использовании CBC-DES формат заголовка Encapsulating Security Payload и оставшаяся часть сообщения будут выглядеть как на рис. 24.9.

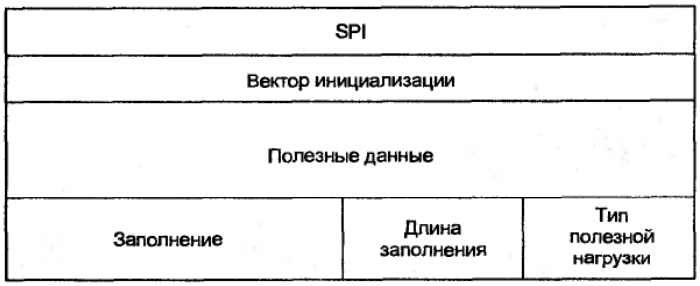

Рис. 24.9. Заголовок и полезная нагрузка при использовании алгоритма CBC-DES.

Вектор инициализации (Initialization Vector) — это блок данных, необходимых для начала работы алгоритма CBC-DES. Затененная область на рисунке представляет зашифрованные данные. Тип 4 означает инкапсулирование в полезной нагрузке всей датаграммы (режим туннеля).

Хотя первоначально планируется использование CBC-DES, в будущем могут появиться другие протоколы для инкапсуляции полезной нагрузки, комбинирующие аутентификацию и целостность данных с шифрованием.

24.5.5 Аутентификация в режиме туннеля

При обмене между граничными маршрутизаторами в режиме туннеля в сообщение могут включаться два независимых заголовка аутентификации. Один будет размещен внутри *исходного* заголовка датаграммы и будет зашифрован и скрыт от остального мира. Этот заголовок обеспечит аутентификацию между конечными точками. Другой заголовок станет частью нешифрованного заголовка IP, используемого для обмена между граничными маршрутизаторами. Он обеспечит аутентификацию между границами сетей.

24.5.6 Обслуживание ключей

Широкое использование безопасности в IP требует распространения множества секретных ключей среди большого количества сетевых узлов. Ключи должны периодически обновляться, и их нужно синхронизовать между собой.

Существует много литературы по управлению ключами. Но ни один из методов управления ключами не специфицирован, поэтому имеются возможности для экспериментирования.

Использование асимметричных общедоступных/личных пар ключей вместо симметричного метода CBC-DES может значительно уменьшить количество администрируемых ключей.

24.6 Дополнительная литература

Следующий список RFC являлся актуальным на момент выхода книги. Последние изменения можно найти в индексе RFC.

RFC 1825 *Security Architecture for the Internet Protocol* (Архитектура безопасности для протокола Интернета). Справочный раздел этого документа содержит список множества других публикаций по теме безопасности.

RFC 1826 *IP Authentication Header* (Заголовок аутентификации протокола IP)

RFC 1828 *IP Authentication using Keyed MD5* (Аутентификация в IP по ключам MD5)

RFC 1321 *The MD5 Message-Digest Algorithm* (Алгоритм вычисления резюме сообщения MD5)

RFC 1827 *IP Encapsulating Security Payload — ESP* (Инкапсуляция защищенной полезной нагрузки в протоколе IP)

RFC 1829 *The ESP DES-CBC Transform* (Преобразование ESP по алгоритму DES-CBC)

Приложение

A

Сокращения и аббревиатуры

AAL	ATM Adaptation Layer (Уровень адаптации в ATM)
ACK	An Acknowledgment (Подтверждение)
AF	Address Family (Семейство адресов)
AH	Authentication Header (Заголовок аутентификации)
ANSI	American National Standards Institute (Американский национальный институт стандартов)
API	Application Programming Interface (Интерфейс программирования приложений)
ARP	Address Resolution Protocol (Протокол разрешения адресов)
ARPA	Advanced Research Projects Agency (Агентство перспективных исследовательских проектов)
ARPANET	Advanced Research Projects Agency Network (Сети ARPA)
AS	Autonomous System (Автономная система)
ASA	American Standards Association (Американская ассоциация по стандартизации)
ASCII	American National Standard Code for Information Interchange (Американский национальный стандартный код для обмена информацией)
ASN.1	Abstract Syntax Notation 1 (Первая абстрактная синтаксическая нотация)
ATM	Asynchronous Transfer Mode (Режим асинхронной пересылки)
BBN	Bolt, Beranek, and Newman, Incorporated (название компании)
BCP	Best Current Practices (лучший текущий способ применения)
BECN	Backward Explicit Congestion Notification (Frame Relay) (Нотификация о явной обратной перегрузке в сетях Frame Relay)
BER	Basic Encoding Rules (Базовые правила декодирования)
BGP	Border Gateway Protocol (Протокол граничного шлюза)
BIND	Berkeley Internet Name Domain (Имена доменов Интернета системы Berkeley)
BOOTP	Bootstrap Protocol (Протокол загрузки)
BPDU	Bridge Protocol Data Unit (Элемент данных протокола для моста)
BRI	Basic Rate Interface (Интерфейс базового уровня передачи)
BSD	Berkeley Software Distribution (Дистрибутив программного обеспечения Беркли — название операционной системы)
CBC	Cipher-Block Chaining (Связывание нулевыми блоками)
CCITT	International Telegraph and Telephone Consultative Committee (Международный консультативный комитет по телеграфии и телефонии), в настоящее время ITU-T (Comite Consultatif International de Telegraphique et Telephonique)

CERT	Computer Emergency Response Team (Подразделение реагирования на компьютерные неисправности)
CHAP	Challenge Handshake Authentication Protocol (Протокол аутентификации по взаимной проверке)
CIDR	Classless Inter-Domain Routing (Бесклассовая междоменная маршрутизация)
CLNP	Connectionless Network Protocol (Сетевой протокол без установления соединений)
CMIP	Common Management Information Protocol (Общий протокол управляющей информации)
CMIS	Common Management Information Services (Общая служба управляющей информации)
CMOT	Common Management Information Services and Protocol over TCP/IP (Общая служба и протокол управляюшей информации поверх TCP/IP)
CPU	Central Processing Unit (Центральный процессор, процессор)
CR	Carriage Return (Возврат каретки — название символа)
CRC	Cyclic Redundancy Check (Циклическая избыточная проверка, контроль циклическим избыточным кодом)
CSLIP	Compressed SLIP (Протокол SLIP со сжатием данных)
CSMA/CD	Carrier Sense Multiple Access with Collision Detection (Множественный доступ с контролем носителя и обнаружением конфликтов)
CSO	Computer Services Organization (Организация по обслуживанию компьютеров)
CSU	Channel Service Unit (Служебный элемент канала)
DAP	Directory Access Protocol (Протокол доступа к каталогам)
DARPA	Defense Advanced Research Projects Agency (Агентство перспективных исследовательских проектов в области обороны)
DCA	Defense Communications Agency (Агентство военных коммуникаций)
DCE	Data Circuit-terminating Equipment (Оборудование терминирования цепей данных)
DCE	Distributed Computing Environment (Распределенное компьютерное окружение)
DDN	Defense Data Network (Военные цифровые сети)
DDN NIC	Defense Data Network Network Information Center (Сетевой информационный центр военных цифровых сетей)
DE	Discard Eligibility (Frame Relay) (Приемлемое удаление в сетях Frame Relay)
DEC	Digital Equipment Corporation (название компании)
DES	Data Encryption Standard (Стандарт шифрования данных)
DEV	Deviation (Отклонение)
DFS	Distributed File Service (Распределенная файловая служба)
DHCP	Dynamic Host Configuration Protocol (Протокол динамического конфигурирования хостов)
DISA	Defense Information Systems Agency (Агентство военных информационных систем)
DIX	Digital, Intel, and Xerox Ethernet protocol (стандарт на Ethernet компаний DEC, Intel и Xerox)
DLCI	Data Link Connection Identifier (Идентификатор соединения по связи данных)
DLL	Dynamic Link Library (Динамическая библиотека связывания)
DME	Distributed Management Environment (Окружение распределенного обслуживания)

DMI	Desktop Management Interface (Интерфейс управления настольными системами)
DMTF	Desktop Management Task Force (Рабочая группа по управлению настольными системами)
DNS	Domain Name System (Система именования доменов)
DOD	Department of Defense (Министерство обороны США)
DOS	Disk Operating System (Дисковая операционная система)
DSA	Directory System Agent (Системный агент каталога)
DSAP	Destination Service Access Point (Точка доступа к службе назначения)
DSU	Data Service Unit (Элемент службы данных)
DTE	Data Terminal Equipment (Оборудование терминирования данных)
DUA	Directory User Agent (Пользовательский агент каталога)
DUAL	Diffusing Update Algorithm (Диффузионный алгоритм внесения изменений)
DXI	Data Exchange Interface (Интерфейс обмена данными)
EBCDIC	Extended Binary-Coded Decimal Interchange Code (Расширенный десятичный код обмена с двоичным кодированием)
EGP	Exterior Gateway Protocol (Внешний протокол шлюза)
EIGRP	Enhanced Internet Gateway Routing Protocol (Улучшенный протокол маршрутизации шлюза Интернета)
EOF	End of File (Конец файла)
EOR	End of Record (Конец записи)
ESMTP	Extensions to SMTP (Расширения протокола SMTP)
ESP	Encapsulating Security Payload (Инкапсулирование защищенной полезной нагрузки)
FAQ	Frequently Asked Questions (Часто задаваемые вопросы)
FCS	Frame Check Sequence (Проверочная последовательность кадров)
FDDI	Fiber Distributed Data Interface (Волоконно-оптический интерфейс данных)
FECN	Forward Explicit Congestion Notification (Frame Relay) (Явное указание на перегрузку при пересылке в сетях Frame Relay)
FIN	Final Segment (Заключительный сегмент)
FIPS	Federal Information Processing Standard (Федеральный стандарт обработки информации)
FTAM	File Transfer, Access and Management (Пересылка файлов, доступ и обслуживание)
FTP	File Transfer Protocol (Протокол пересылки файлов)
FYI	For Your Information ("К вашему сведению")
GGP	Gateway-to-Gateway Protocol (Межшлюзовый протокол)
GIF	Graphics Interchange Format (Формат обмена графикой)
GMT	Greenwich Mean Time (Время по Гринвичу)
GOSIP	Government Open Systems Interconnection Profile (Правительственный профиль взаимодействия открытых систем)
GUI	Graphical User Interface (Графический пользовательский интерфейс)
HDLC	High Level Data Link Control Protocol (Протокол управления связью данных высокого уровня)
HIPPI	High Performance Parallel Interface (Высокопроизводительный параллельный интерфейс)
HTML	Hypertext Markup Language (Язык разметки гипертекста)

HTTP	Hypertext Transfer Protocol (Протокол пересылки гипертекста)
IAB	Internet Architecture Board (Internet Activities Board) (Совет по архитектуре Интернета, ранее — Совет по работе Интернета)
IAC	Interpret As Command (Интерпретировать как команду)
IANA	Internet Assigned Numbers Authority (Авторизация присвоенных номеров Интернета — название организации)
IBM	International Business Machines (название компании)
ICMP	Internet Control Message Protocol (Протокол управляющих сообщений Интернета)
ID	Identifier (Идентификатор)
IDRP	OSI Inter-Domain Routing Protocol (Протокол междоменной маршрутизации модели OSI)
IEEE	Institute of Electrical and Electronics Engineers (Институт инженеров по электротехнике и электронике)
IEN	Internet Engineering Notes (Инженерные заметки Интернета)
IESG	Internet Engineering Steering Group (Управляющая группа технологии Интернета)
IETF	Internet Engineering Task Force (Рабочая группа технологии Интернета)
IGMP	Internet Group Management Protocol (Протокол управления группами Интернета)
IGP	Interior Gateway Protocol (Протокол внешнего шлюза)
IGRP	Internet Gateway Routing Protocol (Протокол маршрутизации шлюза Интернета, лицензия компании Cisco)
ILMI	Interim Local Management Interface (Интерфейс руководства локальным обслуживанием)
IMAP	Internet Mail Access Protocol (Протокол доступа к почте Интернета)
I/O	Input/Output (ввод/вывод)
IP	Internet Protocol (Протокол Интернета)
IPng	IP next generation (version 6) (Протокол Интернета следующего поколения, версия 6)
IPSO	IP Security Option (Варианты безопасности в IP)
IPX	Internetwork Packet eXchange (Обмен межсетевыми пакетами для NetWare)
IRQ	Interrupt Request (Вектор прерывания)
IRTF	Internet Research Task Force (Рабочая группа исследования Интернета)
ISDN	Integrated Services Digital Network (Цифровые сети с интегрированными службами)
IS-IS	Intermediate System to Intermediate System (Взаимодействие между промежуточными системами)
ISN	Initial Sequence Number (Исходный порядковый номер)
ISO	International Organization for Standardization (Международная организация по стандартизации)
ISOC	Internet Society (Сообщество Интернета)
ISODE	ISO Development Environment (Окружение разработки ISO)
ISP	Internet Service Provider (Провайдер Интернета, поставщик сетевых услуг)
ITU	International Telecommunications Union (Международный телекоммуникационный союз)
ITU-T	Telecommunication Standardization Sector of the ITU (Сектор телекоммуникационных стандартов ITU)

JPEG	Joint Photographic Experts Group (Объединенная группа экспертов по фотографии)
LAN	Local Area Network (Локальная сеть)
LAPB	Link Access Procedures Balanced (Процедура балансировки доступа к связи)
LAPD	Link Access Procedures on the D-channel (Процедура доступа к связи по D-каналу)
LF	Line Feed (Перевод строки — название символа)
LLC	Logical Link Control (Управление локальной связью)
MAC	Media Access Control (Управление доступом к носителю)
MAN	Metropolitan Area Network (городские сети)
MD5	Message Digest 5 (Протокол резюме сообщения, версия 5)
MIB	Management Information Base (Информационная база управления)
MIME	Multipurpose Internet Mail Extensions (Многоцелевые почтовые расширения Интернета)
ms	Millisecond (миллисекунда)
MSS	Maximum Segment Size (Максимальный размер сегмента)
MTA	Message Transfer Agent (Агент пересылки сообщений)
MTU	Maximum Transmission Unit (Максимальный элемент пересылки)
MX	Mail Exchanger (Система почтового обмена)
NAP	Network Access Point (Точка доступа к сети)
NCSA	National Center for Supercomputing Applications (Национальный центр по применению суперкомпьютеров)
NDIS	Network Device Interface Specification (Спецификация интерфейса сетевого устройства)
NETBIOS	Network Basic Input Output System (Базовая сетевая система ввода/вывода)
NFS	Network File System (Сетевая файловая система)
NIC	Network Information Center (Сетевой информационный центр)
NIS	Network Information System (Сетевая информационная система)
NISI	Network Information Service Infrastructure (Инфраструктура сетевой информационной службы)
NIST	National Institute of Standards and Technology (Национальный институт стандартов и технологий)
NLPID	Network Level Protocol ID (Идентификатор протокола сетевого уровня)
NNTP	Network News Transfer Protocol (Протокол пересылки сетевых новостей)
NOC	Network Operations Center (Сетевой операционный центр)
NREN	National Research and Education Network (Исследовательская и образовательная национальная сеть)
NS	Name Server (Сервер имен)
NSAP	Network Service Access Point (Точка доступа к сетевой службе)
NSF	National Science Foundation (Национальный научный фонд)
NTP	Network Time Protocol (Протокол сетевого времени)
NVT	Network Virtual Terminal (Сетевой виртуальный терминал)
ODI	Open Device Interface (Интерфейс открытых устройств)
ONC	Open Network Computing (Открытые сетевые вычисления)
OSF	Open Software Foundation (Фонд открытого программного обеспечения)

OSI	Open Systems Interconnect (Взаимодействие открытых систем)
OSPF	Open Shortest Path First (Сначала открывать кратчайший путь)
OUI	Organizationally Unique Identifier (Уникальный идентификатор организации)
PAD	Packet Assembler/Disassembler (Сборка/извлечение пакетов)
PAP	Password Authentication Protocol (Протокол аутентификации по паролям)
PC	Personal Computer (Персональный компьютер)
PDU	Protocol Data Unit (Элемент данных протокола)
PGP	Pretty Good Privacy (Повышенная секретность)
PI	Protocol Interpreter (Интерпретатор протокола)
PING	Packet Internet Groper (Пакетная служба Groper в Интернете)
POP	Point Of Presence (Точка присутствия)
POP	Post Office Protocol (Протокол почтового офиса)
PPP	Point-to-Point Protocol (Протокол "точка-точка")
PTT	Postal Telegraph and Telephone (Почтовый телефон и телеграф)
QoS	Quality of Service (Качество обслуживания)
RA	Routing Arbiter (Арбитр маршрутизации)
RARP	Reverse Address Resolution Protocol (Протокол обратного разрешения адресов)
RFC	Request For Comments (Запрос комментариев)
RIF	Routing Information Field (Информационное поле маршрутизации)
RIP	Routing Information Protocol (Протокол информации маршрутизации)
RIPE	Reseaux IP Europeens (Европейское исследовательское агентство по IP)
RMON	Remote Network Monitor (Удаленный сетевой монитор)
ROM	Read Only Memory (Память "только чтение", постоянное запоминающее устройство)
RPC	Remote Procedure Call (Вызов удаленной процедуры)
RR	Resource Record (Запись о ресурсах)
RR	Routing Registry (Регистрация маршрутов)
RST	Reset (Сброс)
RTO	Retransmission Timeout (Тайм-аут по повторной пересылке для протокола TCP)
RTT	Round-Trip Time (Время цикла)
SDEV	Smoothed Deviation (Округленное отклонение)
SDLC	Synchronous Data Link Control (Управление синхронной связью данных)
SEI	Software Engineering Institute (Институт инженеров программного обеспечения)
SGML	Standard Generalized Markup Language (Обобщенный стандартный язык разметки)
SIP	SMDS Interface Protocol (Интерфейсный протокол SMDS)
SLIP	Serial Line Interface Protocol (Протокол интерфейса последовательной линии)
SMDS	Switched Multimegabit Data Service (Коммутируемая многомегабитная служба данных)
SMI	Structure of Management Information (Структура управляющей информации)
SMTP	Simple Mail Transfer Protocol (Простой протокол пересылки почты)
SNA	Systems Network Architecture (Сетевая архитектура систем)
SNAP	Sub-Network Access Protocol (Протокол доступа к подсетям)

SNMP	Simple Network Management Protocol (Простой протокол сетевого управления)
SOA	Start of Authority (Начало авторизации)
SONET	Synchronous Optical Network (Синхронные оптические сети)
SPF	Shortest Path First (Первым — кратчайший путь)
SPI	Security Parameters Index (Индекс параметров безопасности)
SPX	Sequenced Packet Exchange (Последовательный обмен пакетами, для Netware)
SRTT	Smoothed Round-Trip Time (Округленное время цикла)
SSAP	Source Service Access Point (Точка доступа к службе источника)
SSL	Secure Sockets Layer (Уровень безопасности в socket)
SWS	Silly Window Syndrome (Синдром "бестолкового окна")
SYN	Synchronizing Segment (Сегмент синхронизации)
TCB	Transmission Control Block (Блок управления пересылкой)
TCP	Transmission Control Protocol (Протокол управления пересылкой)
TCU	Trunk Coupling Unit (Парный элемент транковой связи)
TELNET	Terminal Networking (Сетевое взаимодействие терминалов)
TFTP	Trivial File Transfer Protocol (Примитивный протокол пересылки файлов)
TLI	Transport Layer Interface (Интерфейс транспортного уровня)
TOS	Type of Service (Тип обслуживания)
TP4	OSI Transport Class 4 (Класс транспорта 4 модели OSI)
TSAP	Transport Service Access Point (Точка доступа к транспортной службе)
TTL	Time-To-Live (Время жизни, возраст)
UA	User Agent (Пользовательский агент)
UDP	User Datagram Protocol (Протокол пользовательских датаграмм)
ULP	Upper Layer Protocol (Протокол верхнего уровня)
URI	Universal Resource Identifier (Универсальный идентификатор ресурсов)
URL	Uniform Resource Locator (Единый указатель ресурсов)
URN	Uniform Resource Name (Единое имя ресурса)
UTC	Universal Time Coordinated (Универсальное время)
VCC	Virtual Channel Connection (Соединение по виртуальному каналу)
VLSM	Variable Length Subnet Masks (Маска подсети переменной длины)
VPC	Virtual Path Connection (Соединение по виртуальному пути)
W3	World Wide Web (WWW) (Всемирная паутина)
WAIS	Wide Area Information Service (Региональная информационная служба)
WAN	Wide Area Network (Региональная сеть)
WWW	World Wide Web (Всемирная паутина)
WYSIWYG	What You See is What You Get ("Что видим, то и получаем")
XDR	eXternal Data Representation (Внешнее представление данных)
XNS	Xerox Network Systems (Сетевая система компании Xerox)

Приложение

B

RFC и другие документы по TCP/IP

B.1 Возможность получения документов RFC

На момент выхода книги документы RFC можно было получить в службе каталогов и баз данных InterNIC (InterNIC Directory and Database Services), обслуживаемой компанией AT&T. Эта служба доступна по адресу:

http://www.internic.net/

при выборе *DIRECTORY AND DATABASE SERVICES* и перехода по указателю на соответствующий документ RFC. Документы RFC доступны также по адресу:

ftp://ftp.internic.net/

в каталоге */rfc*.

Полный текущий список документов RFC содержится в документе InterNIC */rfc/rfc-index.txt*.

B.2 Assigned Numbers

Комитет Internet Assigned Numbers Authority (IANA) координирует присваивание уникальных значений параметров для протоколов Интернета. IANA уполномочен на это Internet Society (ISOC) и Federal Network Council.

Периодически IANA издает RFC *Assigned Numbers* (присвоенные номера), который сообщает о текущих назначениях параметров и их значениях. Новые параметры можно сразу же получить из общедоступного архива FTP:

ftp://ftp.isi.edu/in-notes/iana/assignments

B.3 Регистрационные формы

Регистрационные формы для имен и адресов Интернета можно получить в службе регистрации InterNIC (InterNIC Registration Services), доступной по адресу:

http://www.internic.net/

при выборе *REGISTRATION SERVICES* и последующем выборе *Templates*.

B.4 Система именования доменов

Служба регистрации содержит информацию о Domain Name System (DNS) в своем архиве пересылки файлов, доступном при выборе на домашней странице этой организации *FTP Archive* с последующим выбором каталога *domain* или через:

ftp://rs.internic.net/domain/

B.5 Стандарты RFC

В документе RFC периодически публикуется официальный список стандартов и их текущий статус. Этот список является самостоятельным стандартом (STD 1). Информация в таблицах от B.1 до B.5 была взята из RFC 1920, опубликованного в марте 1996 г. и отражающего статус и состояние стандартов Интернета на момент выхода данной книги. Текущее состояние стандартов можно узнать в файле *rfc-index.txt*.

Статус протокола отражает процесс его утверждения. После предварительного рассмотрения стандарт получает статус *предложения* (proposed). После дальнейшего изучения, улучшения и пересмотра статус стандарта может быть повышен до *черновика* (draft). Статус *стандарта* (standard) присваивается после тестирования, экспериментального применения и заключительной доработки.

Статус протоколов указывается в таблицах как *обязательный* (required), *рекомендованный* (recommended), *необязательный* (elective), *с ограниченным использованием* (limited use) или как *нерекомендованный* (not recommemded).

В представленных ниже таблицах используются следующие сокращения для *статуса* документов:

НЕОБ.— необязателен,
РЕК.— рекомендован,
ОБ.— обязателен,
СТ.— стандарт,
ЭКС.— экспериментальный,
ИСТ.— исторический,
ПР.— предложенный.

Таблица В.1

Стандарты протоколов

Протокол	Название	Статус	RFC	STD
	Internet Official Protocol Standards (официальные стандарты протоколов Интернета)	ОБ.	1920	1
	Assigned Numbers (присвоенные номера)	ОБ.	1700	2
	Host Requirements-Communications (требования к хостам — коммуникации)	ОБ.	1122	3
	Host Requirements-Applications (требования к хостам — приложения)	ОБ.	1123	3
IP	Internet Protocol (протокол Интернета)	ОБ.	791	5
	Исправления: IP Subnet Extension (расширения подсетей IP)	ОБ.	950	5
	IP Broadcast Datagrams (широковещательные датаграммы IP)	ОБ.	919	5
	IP Broadcast Datagrams with Subnets (широковещательные датаграммы IP в подсетях)	ОБ.	922	5
ICMP	Internet Control Message Protocol (протокол управляющих сообщений Интернета)	ОБ.	792	5
IGMP	Internet Group Multicast Protocol (протокол многоадресной рассылки в группе Интернета)	РЕК.	1112	5
UDP	User Datagram Protocol (протокол пользовательских датаграмм)	РЕК.	768	6
TCP	Transmission Control Protocol (протокол управления пересылкой)	РЕК.	793	7
TELNET	Telnet Protocol (протокол Telnet)	РЕК.	854 855	8
FTP	File Transfer Protocol (протокол пересылки файлов)	РЕК.	959	9
SMTP	Simple Mail Transfer Protocol (простой протокол пересылки почты)	РЕК.	821	10
SMTP-SIZE	SMTP Service Ext for Message Size (размер сообщений расширенных служб SMTP)	РЕК.	1870	10

Таблица В.1 (*продолжение*)

Стандарты протоколов

Протокол	Название	Статус	RFC	STD
SMTP-EXT	SMTP Service Extensions (расширенные службы SMTP)	РЕК.	1869	10
MAIL	Format of Electronic Mail Messages (формат сообщений электронной почты)	РЕК.	822	11
CONTENT	Content Type Header Field (типы заголовочных полей "Содержание")	РЕК.	1049	11
NTPv2	Network Time Protocol (Version 2) (протокол сетевого времени, версия 2)	РЕК.	1119	12
DOMAIN	Domain Name System (система именования доменов)	РЕК.	1034 1035	13
DNS-MX	Mail Routing and the Domain System (маршрутизация почты и система доменов)	РЕК.	974	14
SNMP	Simple Network Management Protocol (протокол управления простыми сетями)	РЕК.	1157	15
SMI	Structure of Management Information (структура управляющей информации)	РЕК.	1155	16
Concise-MIB	Concise MIB Definitions (определение сокращенных MIB)	РЕК.	1212	16
MIB-II	Management Information Base-II (информационная база управления, версия II)	РЕК.	1213	17
NETBIOS	NetBIOS Service Protocols (протокол служб NetBIOS)	НЕОБ.	1001 1002	19
ECHO	Echo Protocol (протокол эхо-сообщений)	РЕК.	862	20
DISCARD	Discard Protocol (протокол отбрасывания сообщений)	НЕОБ.	863	21
CHARGEN	Character Generator Protocol (протокол генератора символов)	НЕОБ.	864	22
QUOTE	Quote of the Day Protocol (протокол "цитат дня")	НЕОБ.	865	23
USERS	Active Users Protocol (протокол активного пользователя)	НЕОБ.	866	24
DAYTIME	Daytime Protocol (протокол времени дня)	НЕОБ.	867	25
TIME	Time Server Protocol (протокол сервера времени)	НЕОБ.	868	26
TFTP	Trivial File Transfer Protocol (примитивный протокол пересылки файлов)	НЕОБ.	1350	33
TP-TCP	ISO Transport Service on Top of the TCP (транспортные службы ISO поверх TCP)	НЕОБ.	1006	35
ETHER-MIB	Ethernet MIB (база данных MIB для Ethernet)	НЕОБ.	1643	50
PPP	Point-to-Point Protocol (протокол "точка-точка")	НЕОБ.	1661	51
PPP-HDLC	PPP in HDLC Framing (PPP в кадрах HDLC)	НЕОБ.	1662	51
IP-SMDS	IP Datagrams over the SMDS Service (датаграммы поверх службы SMDS)	НЕОБ.	1209	52

Таблица В.2

Стандарты протоколов специфичных сетей

Протокол	Название	Состояние	RFC	STD
IP-ATM	Classical IP and ARP over ATM (классические IP и ARP поверх ATM)	ПР.	1577	
IP-FR	Multiprotocol over Frame Relay (многопротокольность в сетях Frame Relay)	Черновик	1490	
ATM-ENCAP	Multiprotocol Encapsulation over ATM (многопротокольная инкапсуляция поверх ATM)	ПР.	1483	
IP-TR-MC	IP Multicast over Token-Ring LANs (многоадресные рассылки IP через локальные сети Token-Ring)	ПР.	1469	
IP-FDDI	Transmission of IP and ARP over FDDI Net (пересылка IP и ARP в сетях FDDI)	СТ.	1390	36
IP-HIPPI	IP and ARP on HIPPI (IP и ARP в HIPPI)	ПР.	1374	
IP-X.25	X.25 and ISDN in the Packet Mode (пакетный режим в сетях X.25 и ISDN)	Черновик	1356	
IP-FDDI	Internet Protocol on FDDI Networks (протокол Интернета в сетях FDDI)	Черновик	1188	
ARP	Address Resolution Protocol (протокол разрешения адресов)	СТ.	826	37
RARP	A Reverse Address Resolution Protocol (протокол обратного разрешения адресов)	СТ.	903	38
IP-ARPA	Internet Protocol on ARPANET (протокол Интернета в ARPANET)	СТ.	BBN 1822	39
IP-WB	Internet Protocol on Wideband Network (протокол Интернета в широкополосных сетях)	СТ.	907	40
IP-E	Internet Protocol on Ethernet Networks (протокол Интернета в сетях Ethernet)	СТ.	894	41
IP-EE	Internet Protocol on Exp. Ethernet Nets (протокол Интернета в экспериментальных сетях Ethernet)	СТ.	895	42
IP-IEEE	Internet Protocol on IEEE 802 (протокол Интернета в сетях IEEE 802)	СТ.	1042	43
IP-DC	Internet Protocol on DC Networks (протокол Интернета в сетях DC)	СТ.	891	44
IP-HC	Internet Protocol on Hyperchannel (протокол Интернета в Hyperchannel)	СТ.	1044	45
IP-ARC	Transmitting IP Traffic over ARCNET Nets (пересылка трафика IP в сетях ARCNET)	СТ.	1201	46
IP-SLIP	Transmission of IP over Serial Lines (пересылка трафика IP через последовательные линии связи)	СТ.	1055	47
IP-NETBIOS	Transmission of IP over NETBIOS (пересылка трафика IP поверх NETBIOS)	СТ.	1088	48
IP-IPX	Transmission of 802.2 over IPX Networks (пересылка трафика 802.2 через сети IPX)	СТ.	1132	49

Таблица 8.3

Варианты telnet

Протокол	Название	Вариант	Состояние	Статус	RFC	STD
TOPT-BIN	Binary Transmission (пересылка двоичных данных)	0	СТ.	РЕК.	856	27
TOFT-ECHO	Echo (эхо-сообщения)	1	СТ.	РЕК.	857	28
TOPT-RECN	Reconnection (повторное соединение)	2	ПР.	НЕОБ.		
TOPT-SUPP	Suppress Go Ahead (поддержка команды "Дальше")	3	СТ.	РЕК.	858	29
TOPT-APRX	Approx Message Size Negotiation (согласование предварительного размера сообщения)	4	ПР.	НЕОБ.		
TOPT-STAT	Status (статус)	5	СТ.	РЕК.	859	30
TOPT-TIM	Timing Mark (метка времени)	6	СТ.	РЕК.	860	31
TOPT-REM	Remote Controlled Trans and Echo (удаленная управляемая пересылка и эхо-обмен)	7	ПР.	НЕОБ.	726	
TOPT-OLW	Output Line Width (выходная ширина строки)	8	ПР.	НЕОБ.		
TOPT-OPS	Output Page Size (выходной размер страницы)	9	ПР.	НЕОБ.		
TOPT-OCRD	Output Carriage-Return Disposition (размещение выходных символов перевода каретки)	10	ПР.	НЕОБ.	652	
TOPT-OHT	Output Horizontal Tabstops (размещение выходных меток для горизонтальных табуляций)	11	ПР.	НЕОБ.	653	
TOPT-OHTD	Output Horizontal Tab Disposition (размещение выходных горизонтальных табуляций)	12	ПР.	НЕОБ.	654	
TOPT-OFD	Output Formfeed Disposition (размещение выходных символов перевода формата)	13	ПР.	НЕОБ.	655	

Таблица В.3 (*продолжение*)

Варианты telnet

Протокол	Название	Вариант	Состояние	Статус	RFC	STD
TOPT-OVT	Output Vertical Tabstops (размещение выходных меток для вертикальных табуляций)	14	ПР.	НЕОБ.	656	
TOPT-OVTD	Output Vertical Tab Disposition (размещение выходных вертикальных табуляций)	15	ПР.	НЕОБ.	657	
TOPT-OLD	Output Linefeed Disposition (размещение выходных символов перевода строки)	16	ПР.	НЕОБ.	658	
TOPT-EXT	Extended ASCII (расширенный набор символов ASCII)	17	ПР.	НЕОБ.	698	
TOPT-LOGO	Logout (выход из системы)	18	ПР.	НЕОБ.	727	
TOPT-BYTE	Byte Macro (байтовые макросы)	19	ПР.	НЕОБ.	735	
TOPT-DATA	Data Entry Terminal (терминал ввода данных)	20	ПР.	НЕОБ.	1043	
TOPT-SUP	SUPDUP	21	ПР.	НЕОБ.	736	
TOPT-SUPO	SUPDUP Output (вывод SUPDUP)	22	ПР.	НЕОБ.	749	
TOPT-SNDL	Send Location (послать текущую позицию)	23	ПР.	НЕОБ.	779	
TOPT-TERM	Terminal Type (тип терминала)	24	ПР.	НЕОБ.	1091	
TOPT-EOR	End of Record (конец записи)	25	ПР.	НЕОБ.	885	
TOPT-TACACS	TACACS User Identification (идентификация пользователя методом TACACS)	26	ПР.	НЕОБ.	927	
TOPT-OM	Output Marking (выходные маркеры)	27	ПР.	НЕОБ.	933	
TOPT-TLN	Terminal Location Number (номер позиции X-терминала)	28	ПР.	НЕОБ.	946	

Таблица В.3 (*продолжение*)

Варианты telnet

Протокол	Название	Вариант	Состояние	Статус	RFC	STD
TOPT-3270	Telnet 3270 Regime (режим совместимости telnet с терминалами 3270)	29	ПР.	НЕОБ.	1041	
TOPT-X.3	X.3 PAD	30	ПР.	НЕОБ.	1053	
TOPT-NAWS	Negotiate About Window Size (согласование размера окна)	31	ПР.	НЕОБ.	1073	
TOPT-TS	Terminal Speed (скорость терминала)	32	ПР.	НЕОБ.	1079	
TOPT-RFC	Remote Flow Control (удаленное управление потоком)	33	ПР.	НЕОБ.	1372	
TOPT-LINE	Linemode (строчный режим)	34	Черновик	НЕОБ.	1184	
TOPT-XDL	X Display Location (положение на дисплее X)	35	ПР.	НЕОБ.	1096	
TOPT-ENVIR	Telnet Environment Option (параметры окружения telnet)	36	ИСТ.	Нет	1408	
TOPT-AUTH	Telnet Authentication Option (параметры аутентификации telnet)	37	ЭКС.	НЕОБ.	1416	
TOPT-ENVIR	Telnet Environment Option (параметры окружения telnet)	39	ПР.	НЕОБ.	1572	
TOPT-EXTOP	Extended-Options-List (расширенный список параметров)	255	СТ.	РЕК.	861	32

Таблица В.4

Черновики стандартов

Протокол	Название	Статус	RFC
COEX-MIB	Coexistence between SNMPvl & SNMPv2 (сосуществование стандартов SNMPvl и SNMPv2)	НЕОБ.	1908
SNMPv2-MIB	MIB for SNMPv2 (базы данных MIB для SNMPv2)	НЕОБ.	1907
TRANS-MIB	Transport Mappings for SNMPv2 (транспортное отображение для SNMPv2)	НЕОБ.	1906
OPS-MIB	Protocol Operations for SNMPv2 (операционный протокол для SNMPv2)	НЕОБ.	1905
CONF-MIB	Conformance Statements for SNMPv2 (соответствие операторов для SNMPv2)	НЕОБ.	1904

Таблица В.4 (*продолжение*)

Черновики стандартов

Протокол	Название	Статус	RFC
CONV-MIB	Textual Conventions for SNMPv2 (соглашения для текстов в SNMPv2)	НЕОБ.	1903
SMIV2	SMI for SNMPv2 (SMI для SNMPV2)	НЕОБ.	1902
CON-MD5	Content-MD5 Header Field (содержимое полей MD5 в заголовке)	НЕОБ.	1864
OSPF-MIB	OSPF Version 2 MIB (базы данных MIB по OSPF версии 2)	НЕОБ.	1850
STR-REP	String Representation of Distinguished Names (строковые представления различающихся имен)	НЕОБ.	1779
X.500syn	X.500 String Representation of Standard Attribute Syntaxes (представление строк X.500 в стандартном синтаксисе атрибутов)	НЕОБ.	1778
X.500lite	X.500 Lightweight Directory Access Protocol (облегченный протокол доступа к каталогам X.500)	НЕОБ.	1777
BGP-4-APP	Application of BGP-4 (приложения для BGP-4)	НЕОБ.	1772
BGP-4	Border Gateway Protocol 4 (протокол граничного шлюза, версия 4)	НЕОБ.	1771
PPP-DNCP	PPP DECnet Phase IV Control Protocol (управляющий протокол для соединений "точка-точка" в сетях DECnet Phase IV)	НЕОБ.	1762
RMON-MIB	Remote Network Monitoring MIB (базы данных MIB удаленного сетевого мониторинга)	НЕОБ.	1757
802.5-MIB	IEEE 802.5 Token Ring MIB (базы данных MIB в сетях IEEE 802.5 Token Ring)	НЕОБ.	1748
BGP-4-MIB	BGP-4 MIB (базы данных MIB для протокола BGP-4)	НЕОБ.	1657
POP3	Post Office Protocol, Version 3 (протокол почтового офиса, версия 3)	НЕОБ.	1725
RIP2-MIB	RIP Version 2 MIB Extension (расширение MIB для протокола RIP версии 2)	НЕОБ.	1724
RIP2	RIP Version 2 — Carrying Additional Info. (протокол RIP версии 2 — перенос дополнительной информации)	НЕОБ.	1723
RIP2-APP	RIP Version 2 Protocol App. Statement (протокол RIP версии 2 — прикладные операторы)	НЕОБ.	1722
SIP-MIB	SIP Interface Type MIB (типы интерфейсов SIP в MIB)	НЕОБ.	1694
	Def Man Objs Parallel-printer-like (описание основных объектов, подобных параллельному принтеру)	НЕОБ.	1660
	DefMan Objs RS-232-like (описание основных объектов, подобных интерфейсу RS-232)	НЕОБ.	1659
	Def Man Objs Character Stream (описание основных объектов символьного потока)	НЕОБ.	1658
SMTP-8BIT	SMTP Service Ext or 8bit-MIMEtransport (служебные расширения SMTP или 8-битовый транспорт, совместимый с MIME)	НЕОБ.	1652

Таблица В.4 (*продолжение*)

Черновики стандартов

Протокол	Название	Статус	RFC
OSI-NSAP	Guidelines for OSI NSAP Allocation (руководство по распределению OSI NSAP)	НЕОБ.	1629
OSPF2	Open Shortest Path First Routing V2 (маршрутизация методом "Открывать первым кратчайший путь" версии 2)	НЕОБ.	1583
ISO-TS-ECHO	Echo for ISO-8473 (эхо-сообщение для ISO-8473)	НЕОБ.	1575
DECNET-MIB	DECNET MIB (базы данных MIB для сетей DECNET)	НЕОБ.	1559
	Message Header Ext. of Non-ASCII Text (расширение заголовка сообщения для не-ASCII текстов)	НЕОБ.	1522
MIME	Multipurpose Internet Mail Extensions (многоцелевые почтовые расширения Интернета)	НЕОБ.	1521
802.3-MIB	IEEE 802.3 Repeater MIB (MIB-повторители для IEEE 802.3)	НЕОБ.	1516
BRIDGE-MIB	BRIDGE-MIB (базы данных MIB для мостов)	НЕОБ.	1493
NTPv3	Network Time Protocol (Version 3) (протокол сетевого времени, версия 3)	НЕОБ.	1305
IP-MTU	Path MTU Discovery (исследование MTU по пути следования)	НЕОБ.	1191
FINGER	Finger Protocol (протокол программы Finger)	НЕОБ.	1288
BOOTP	Bootstrap Protocol (протокол загрузки)	РЕК.	951, 1497
NICNAME	Whols Protocol (протокол коротких имен)	НЕОБ.	954

Таблица В.5

Предложенные стандарты

Протокол	Название	Статус	RFC
WHOIS++M	How to Interact with a Whois++ Mesh (как взаимодействовать с базой данных Whois++)	НЕОБ.	1914
WHOIS++A	Architecture of Whois++ Index Service (архитектура службы индексов Whois++)	НЕОБ.	1913
DSN	Delivery Status Notifications (доставка уведомлений о статусе)	НЕОБ.	1894
EMS-CODE	Enhanced Mail System Status Codes (расширенные коды статуса почтовой системы)	НЕОБ.	1893
MIME-RPT	Multipart/Report (многофрагментные сообщения/отчеты)	НЕОБ.	1892
SMTP-DSN	SMTP Delivery Status Notifications (уведомление о доставке статуса в SMTP)	НЕОБ.	1891
RTP-AV	RTP Audio/Video Profile (профиль аудио/видео RTP)	НЕОБ.	1890
RTP	Transport Protocol for Real-Time Apps (транспортный протокол для приложений реального времени)	НЕОБ.	1889
DNS-IPv6	DNS Extensions to Support IPv6 (расширение DNS для поддержки протокола IPv6)	НЕОБ.	1886
ICMPv6	ICMPv6 for IPv6 (ICMPv6 для IPv6)	НЕОБ.	1885

Таблица В.5 (*продолжение*)

Предложенные стандарты

Протокол	Название	Статус	RFC
IPv6-Addr	IPv6 Addressing Architecture (адресная архитектура IPv6)	НЕОБ.	1884
IPv6	IPv6 Specification (спецификация IPv6)	НЕОБ.	1883
HTML	Hypertext Markup Language - 2.0 (язык разметки гипертекста, версия 2)	НЕОБ.	1866
SMTP-Pipe	SMTP Serv. Ext. for Command Pipelining (сервисное расширение SMTP для пересылки команд в канале)	НЕОБ.	1854
MIME-Sec	MIME Object Security Services (служба безопасности объектов MIME)	НЕОБ.	1848
MIME-Encyp	MIME: Signed and Encrypted (MIME: подпись и шифрование)	НЕОБ.	1847
WHOIS++	Architecture of the WHOIS++ service (архитектура службы WHOIS++)	НЕОБ.	1835
	Binding Protocols for ONC RPC Version 2 (протокол связывания для ONC RPC версии 2)	НЕОБ.	1833
XDR	External Data Representation Standard (стандарт внешнего представления данных)	НЕОБ.	1832
RPC	Remote Procedure Call Protocol V. 2 (протокол вызова удаленных процедур, версия 2)	НЕОБ.	1831
	ESP DES-CBC Transform (преобразование ESP DES-CBC)	НЕОБ./ОБ.	1829
	IP Authentication Using Keyed MD5 (аутентификация IP по алгоритму MD5 с использованием ключей)	НЕОБ./ОБ.	1828
ESP IP	Encapsulating Security Payload (инкапсуляция защищенной полезной нагрузки)	НЕОБ./ОБ.	1827
IPv6-AH	IP Authentication Header (заголовок аутентификации IP)	НЕОБ./ОБ.	1826
	Security Architecture for IP (архитектура безопасности IP)	НЕОБ./ОБ.	1825
RREQ	Requirements for IP Version 4 Routers (требования к маршрутизаторам протокола IP версии 4)	НЕОБ.	1812
URL	Relative Uniform Resource Locators (относительный единый указатель ресурсов)	НЕОБ.	1808
CLDAP	Connectionless LDAP (LDAP без создания соединений)	НЕОБ.	1798
OSPF-DC	Ext. OSPF to Support Demand Circuits (расширения OSPF для поддержки цепей по требованию)	НЕОБ.	1793
TMUX	Transport Multiplexing Protocol (протокол мультиплексирования транспорта)	НЕОБ.	1692
TFTP-Opt	TFTP Options (варианты TFTP)	НЕОБ.	1784
TFTP-Blk	TFTP Blocksize Option (параметры размеров блоков TFTP)	НЕОБ.	1783
TFTP-Ext	TFTP Option Extension (расширенные возможности TFTP)	НЕОБ.	1782
OSI-Dir	OSI User Friendly Naming ... (удобное пользователям именование в OSI)	НЕОБ.	1781
MIME-EDI	MIME Encapsulation of EDI Objects (инкапсуляция в MIME объектов EDI)	НЕОБ.	1767

Таблица 8.5 (*продолжение*)

Предложенные стандарты

Протокол	Название	Статус	RFC
Lang-Tag	Tags for Identification of Languages (теги идентификации языков)	НЕОБ.	1766
XNSCP	PPP XNS IDP Control Protocol (управляющий протокол PPP XNS IDP)	НЕОБ.	1764
BVCP	PPP Banyan Vines Control Protocol (управляющий протокол "точка-точка" компании Banyan Vines)	НЕОБ.	1763
Print-MIB	Printer MIB (база данных MIB принтеров)	НЕОБ.	1759
ATM-SIG	ATM Signaling Support for IP over ATM (поддержка сигналов в IP поверх ATM)	НЕОБ.	1755
IPNG	Recommendation for IP Next Generation (рекомендации по IP следующего поколения)	НЕОБ.	1752
802.5-SSR	802.5 SSR MIB using SMIv2 (802.5 SSR MIB с использованием SMIv2)	НЕОБ.	1749
SDLCSMIv2	SNADLC SDLC MIB using SMIv2 (SNADLC SDLC MIB с использованием SMIv2)	НЕОБ.	1747
BGP4/IDRP	BGP4/IDRP for IP/OSPF Interaction (BGP4/IDRP для взаимодействий IP/OSPF)	НЕОБ.	1745
AT-MIB	Appletalk MIB (базы данных MIB для Apple Talk)	НЕОБ.	1742
MacMIME	MIME Encapsulation of Macintosh files (инкапсуляция в MIME файлов Macintosh)	НЕОБ.	1740
URL	Uniform Resource Locators (единый указатель ресурсов)	НЕОБ.	1738
POP3-AUTH	POPS AUTHentication Command (команды аутентификации в POP3)	НЕОБ.	1734
IMAP4-AUTH	IMAP4 Authentication Mechanisms (механизм аутентификации IMAP4)	НЕОБ.	1731
IMAP4	Internet Message Access Protocol V4 (протокол сообщений доступа Интернета, версия 4)	НЕОБ.	1730
PPP-MP	PPP Multilink Protocol (протокол PPP для нескольких связей)	НЕОБ.	1717
RDBMS-MIB	RDMS MIB - using SMIv2 (RDMS MIB с использованием SMIv2)	НЕОБ.	1697
MODEM-MIB	Modem MIB - using SMIv2 (модем MIB с использованием SMIv2)	НЕОБ.	1696
ATM-MIB	ATM Management Version 8.0 using SMIv2 (управляющий протокол ATM версии 8.0 с использованием SMIv2)	НЕОБ.	1695
SNANAU-MIB	SNA NAUs MIB using SMIv2 (SNA NAU MIB с использованием SMIv2)	НЕОБ.	1665
PPP-TRANS	PPP Reliable Transmission (надежная пересылка в PPP)	НЕОБ.	1663
BGP-4-IMP	BGP-4 Roadmap and Implementation (цели и реализация BGP-4)	НЕОБ.	1656
	Postmaster Convention X.400 Operations (операции преобразования Postmaster в X.400)	НЕОБ.	1648

Таблица В.5 (*продолжение*)

Предложенные стандарты

Протокол	Название	Статус	RFC
TN3270-En	TN3270 Enhancements (улучшения в TN3270)	НЕОБ.	1647
PPP-BCP	PPP Bridging Control Protocol (протокол управления мостами PPP)	НЕОБ.	1638
UPS-MIB	UPS Management Information Base (информационная база управления источниками резервного питания)	НЕОБ.	1628
AAL5-MTU	Default IP MTU for use over ATM AAL5 (значение по умолчанию MTU из IP для использования поверх ATM AAL5)	НЕОБ.	1626
PPP-SONET	PPP over SONET/SDH (PPP поверх SONET/SDH)	НЕОБ.	1619
PPP-ISDN	PPP over ISDN (PPP поверх ISDN)	НЕОБ.	1618
DNS-R-MIB	DNS Resolver MIB Extensions (расширение MIB для определителя DNS)	НЕОБ.	1612
DNS-S-MIB	DNS Server MIB Extensions (расширение MIB для сервера DNS)	НЕОБ.	1611
FR-MIB	Frame Relay Service MIB (MIB для служб Frame Relay)	НЕОБ.	1604
PPP-X25	PPP in X.25 (протокол PPP в сетях X.25)	НЕОБ.	1598
OSPF-NSSA	The OSPF NSSA Option (варианты OSPF NSSA)	НЕОБ.	1587
OSPF-Multi	Multicast Extensions to OSPF (расширение для многоадресных рассылок в OSPF)	НЕОБ.	1584
SONET-MIB	MIB SONET/SDH Interface Type (тип интерфейса MIB для SONET/SDH)	НЕОБ.	1595
RIP-DC	Extensions to RIP to Support Demand Cir. (расширение RIP для поддержки цепей по требованию)	НЕОБ.	1582
	Evolution of the Interfaces Group of MIB-II (эволюция группы интерфейсов MIB-II)	НЕОБ.	1573
PPP-LCP	PPP LCP Extensions (расширения PPP LCP)	НЕОБ.	1570
X500-MIB	X.500 Directory Monitoring MIB (MIB мониторинга каталогов X.500)	НЕОБ.	1567
MAIL-MIB	Mail Monitoring MIB (MIB мониторинга почты)	НЕОБ.	1566
NSM-MIB	Network Services Monitoring MIB (MIB мониторинга сетевых служб)	НЕОБ.	1565
CIPX	Compressing IPX Headers Over WAN Media (сжатые заголовки IPX поверх носителя региональных сетей)	НЕОБ.	1553
IPXCP	PPP Internetworking Packet Exchange Control (управление межсетевым обменом пакетов в PPP)	НЕОБ.	1552
DHCP-BOOTP	Interoperation Between DHCP and BOOTP (взаимодействие между DHCP и BOOTP)	НЕОБ.	1534
DHCP-BOOTP	DHCP Options and BOOTP Vendor Extensions (расширения вариантов DHCP и BOOTP от разработчиков)	НЕОБ.	1533
BOOTP	Clarifications and Extensions BOOTP (выверенный и расширенный протокол BOOTP)	НЕОБ.	1532

Таблица В.5 (*продолжение*)

Предложенные стандарты

Протокол	Название	Статус	RFC
DHCP	Dynamic Host Configuration Protocol (протокол динамического конфигурирования хоста)	НЕОБ.	1541
SRB-MIB	Source Routing Bridge MIB (MIB для моста с маршрутизацией от источника)	НЕОБ.	1525
CIDR-STRA	CIDR Address Assignment... (присваивание адресов CIDR)	НЕОБ.	1519
CIDR-ARCH	CIDR Architecture... (архитектура CIDR)	НЕОБ.	1518
CIDR-APP	CIDR Applicability Statement (прикладные операторы CIDR)	НЕОБ.	1517
	802.3 MAU MIB	НЕОБ.	1515
HOST-MIB	Host Resources MIB (MIB ресурсов хоста)	НЕОБ.	1514
	Token Ring Extensions to RMON MIB (расширение Token Ring для RMON MIB)	НЕОБ.	1513
FDDI-MIB	FDDI Management Information Base (база управляющей информации для FDDI)	НЕОБ.	1512
KERBEROS	Kerberos Network Authentication Ser (V5) (сервер сетевой аутентификации Kerberos, версия 5)	НЕОБ.	1510
GSSAPI	Generic Security Service API: C-bindings (основная служба безопасности API: связывание с C)	НЕОБ.	1509
GSSAPI	Generic Security Service Application... (основная служба безопасности приложений)	НЕОБ.	1508
DASS	Distributed Authentication Security... (распределенная аутентификация безопасности)	НЕОБ.	1507
	X.400 Use of Extended Character Sets (использование расширенного набора символов X.400)	НЕОБ.	1502
HARPOON	Rules for Downgrading Messages... (правила разделения сообщений)	НЕОБ.	1496
Mapping	MHS/RFC-822 Message Body Mapping (отображение тела сообщения MHS/RFC-822)	НЕОБ.	1495
Equiv	X.400/MIME Body Equivalences (соответствие тел сообщений X.400/MIME)	НЕОБ.	1494
IDPR	Inter-Domain Policy Routing Protocol (протокол междоменной политики маршрутизации)	НЕОБ.	1479
IDPR-ARCH	Architecture for IDPR (архитектура IDPR)	НЕОБ.	1478
PPP/Bridge MIB	Bridge PPP MIB (MIB мостов PPP)	НЕОБ.	1474
PPP/IP MIB	IP Network Control Protocol of PPP MIB (управляющий сетевой протокол IP для PPP MIB)	НЕОБ.	1473
PPP/SEC MIB	Security Protocols of PPP MIB (протокол безопасности PPP MIB)	НЕОБ.	1472
PPP/LCP MIB	Link Control Protocol of PPP MIB (протокол управления связью PPP MIB)	НЕОБ.	1471
X25-MIB	Multiprotocol Interconnect on X.25 MIB (MIB многопротокольного взаимодействия в сетях X.25)	НЕОБ.	1461
PEM-KEY	PEM — Key Certification (сертификационные ключи PEM)	НЕОБ.	1424

Таблица В.5 (*продолжение*)

Предложенные стандарты

Протокол	Название	Статус	RFC
PEM-ALG	PEM — Algorithms, Modes, and Identifiers (алгоритмы, режимы и идентификаторы PEM)	НЕОБ.	1423
PEM-CKM	PEM — Certificate-Based Key Management (обслуживание сертификационных ключей PEM)	НЕОБ.	1422
PEM-ENC	PEM — Message Encryption and Auth (аутентификация и шифрование сообщений PEM)	НЕОБ.	1421
SNMP-IPX	SNMP over IPX (SNMP поверх IPX)	НЕОБ.	1420
SNMP-AT	SNMP over AppleTalk (SNMP поверх AppleTalk)	НЕОБ.	1419
SNMP-OSI	SNMP over OSI (SNMP поверх OSI)	НЕОБ.	1418
FTP-FTAM	FTP-FTAM Gateway Specification (спецификация шлюза FTP-FTAM)	НЕОБ.	1415
IDENT-MIB	Identification MIB (идентификация MIB)	НЕОБ.	1414
IDENT	Identification Protocol (протокол идентификации)	НЕОБ.	1413
DS3/E3-MIB	DS3/E3 Interface Type (тип интерфейса DS3/E3)	НЕОБ.	1407
DS1/E1-MIB	DS1/E1 Interface Type (тип интерфейса DS1/E1)	НЕОБ.	1406
BGP-OSPF	BGP OSPF Interaction (взаимодействие BGP и OSPF)	НЕОБ.	1403
	Route Advertisement In BGP2 And BGP3 (объявление маршрута в BGP2 и BGP3)	НЕОБ.	1397
SNMP-X.25	SNMP MIB Extension for X.25 Packet Layer (расширение MIB протокола SNMP для уровня пакетов X.25)	НЕОБ.	1382
SNMP-LAPB	SNMP MIB Extension for X.25 LAPB (расширение MIB протокола SNMP для X.25 LAPB)	НЕОБ.	1381
PPP-ATCP	PPP AppleTalk Control Protocol (управляющий протокол PPP AppleTalk)	НЕОБ.	1378
PPP-OSINLCP	PPP OSI Network Layer Control Protocol (управляющий протокол сетевого уровня PPP OSI)	НЕОБ.	1377
TABLE-MIB	IP Forwarding Table MIB (пересылка таблиц MIB в IP)	НЕОБ.	1354
TOS	Type of Service in the Internet (тип обслуживания в Интернете)	НЕОБ.	1349
PPP-AUTH	PPP Authentication (аутентификация в PPP)	НЕОБ.	1334
PPP-LINK	PPP Link Quality Monitoring (мониторинг качества связи PPP)	НЕОБ.	1333
PPP-IPCP	PPP Control Protocol (управляющий протокол PPP)	НЕОБ.	1332
	X.400 1988 to 1984 downgrading (градации в X.400 с 1988 по 1984 гг.)	НЕОБ.	1328
	Mapping between X.400 (1988) (отображение между X.400 от 1988 г.)	НЕОБ.	1327
TCP-EXT	TCP Extensions for High Performance (расширение TCP для обеспечения высокой производительности)	НЕОБ.	1323

Таблица 8.5 (*продолжение*)

Предложенные стандарты

Протокол	Название	Статус	RFC
FRAME-MIB	Management Information Base for Frame (информационная база управления для кадра)	НЕОБ.	1315
NETFAX	File Format for the Exchange of Images (формат файла для обмена изображениями)	НЕОБ.	1314
IARP	Inverse Address Resolution Protocol (обратный протокол разрешения адресов)	НЕОБ.	1293
FDDI-MIB	FDDI-MIB (база данных MIB для FDDI)	НЕОБ.	1285
	Encoding Network Addresses (декодирование сетевого адреса)	НЕОБ.	1277
	Replication and Distributed Operations (репликация и распределенные операции)	НЕОБ.	1276
	COSINE and Internet X.500 Schema (COSINE и схема X.500 для Интернета)	НЕОБ.	1274
BGP-MIB	Border Gateway Protocol MIB (Version 3) (MIB протокола граничного шлюза версии 3)	НЕОБ.	1269
ICMP-ROUT	ICMP Router Discovery Messages (сообщения исследования маршрутизаторов ICMP)	НЕОБ.	1256
IPSO	DoD Security Options for IP (параметры безопасности для IP Министерства обороны США)	НЕОБ.	1108
OSI-UDP	OSI TS on UDP (OSI TS в UDP)	НЕОБ.	1240
STD-MIBs	Reassignment of Exp MIBs to Std MIBs (переприсваивание расширенных MIB стандартным MIB)	НЕОБ.	1239
IPX-IP	Tunneling IPX Traffic through IP Nets (туннель трафика IPX в сети IP)	НЕОБ.	1234
GINT-MIB	Extensions to the Generic-Interface MIB (расширение для MIB с общим интерфейсом)	НЕОБ.	1229
IS-IS	OSI IS-IS for TCP/IP Dual Environments (OSI IS-IS для двойного окружения TCP/IP)	НЕОБ.	1195
IP-CMPRS	Compressing TCP/IP Headers (сжатие заголовков TCP/IP)	НЕОБ.	1144
NNTP	Network News Transfer Protocol (протокол пересылки сетевых новостей)	НЕОБ.	977

Приложение

C

Центры сетевой информации и другие службы

C.1 Регистрация

Прежде чем организация сможет подключить свою сеть к Интернету, ей нужно получить один или несколько блоков IP-адресов от провайдера или непосредственно от службы регистрации Интернета. Организация должна зарегистрировать свое имя домена и идентифицировать серверы имен доменов (DNS). Ей может потребоваться *номер автономной системы*, который является уникальным целым числом, присвоенным сети данной организации.

C.1.1 Основная служба регистрации NIC

Основная служба регистрации Интернета (Internet Registration Service) в настоящее время финансируется Национальным научным фондом (National Science Foundation — NSF). Эта служба обеспечивает всемирную координацию и делегирует права службам регистрации Северной и Южной Америки. Адрес службы:

Network Solutions
Attn: InterNIC Registration Services
505 Huntmar Park Drive
Herndon, Virginia 22070

Через электронную почту: hostmaster@internic.net

Регистрация хостов и доменов, а также изменение регистрационной информации могут быть выполнены по электронной почте. Как отмечено в приложении B, формы регистрации доступны через WWW по адресу:

http://www.internic.net/

или через FTP:

ftp://ftp.internic.net/templates

C.1.2 Европейская служба NIC

Основная европейская служба NIC:

RIPE Network Coordination Centre (RIPE NCC)
(Registry for the European Region)

Электронная почта: hostmaster@ripe.net, ncc@ripe.net

Телефон: +31 20 592 5065

Факс: +31 20 592 5090

Почтовый адрес: RIPE NCC
Kruislaan 409
1098 SJ Amsterdam
The Netherlands

Сетевой координационный центр RIPE:

http//www.ripe.net/

C.1.3 Азиатско-Тихоокеанский NIC

NIC для Азиатско-Тихоокеанского региона:

Asia Pacific Network Information Center
c/o Internet Initiative Japan, Inc.
Sanbancho Annex Bldg.
1-4 Sanbancho, Chiyoda-ku
Tokyo 102, Japan

Электронная почта: ip-request@rs.apnic.net

Телефон: + 81-3-5276-3973

Факс: + 81-3-5276-6239

Сетевой информационный центр этого региона:

http://www.apnic.net/

ftp://archive.apnic.net/apnic/docs/

Эти три главных сетевых информационных центра (Network Information Centers — NIC) делегируют права регистрации адресов национальным организациям и провайдерским NIC в пределах своих регионов.

C.2 Поиск других NIC

Служба данных NIC компании AT&T периодически публикует список других NIC по адресу:

http://ds.internic.net/pub/niclocator/

Список обслуживается рабочей группой инфраструктуры сетевой информационной службы (Network Information Services Infrastructure — NISI), созданной в рамках Internet Engineering Task Force (IETF).

C.3 Поиск администраторов через WHOIS

Регистрационная информация об организации включает имена административных и технических сотрудников для контактов и сведения о способах обращения к ним.

Эта информация доступна в интерактивной базе данных, к которой можно обращаться через приложение *whois*. Ниже приводится запрос сведений о домене *yale.edu*. Первый ответ дает нам главную справку, YALE-DOM, используемую для получения дополнительной информации о домене.

```
> whois -h rs.internic.net yale.edu
Yale University (YALE-DOM)                                    YALE.EDU
Yale University (YALE)              YALE.EDU          128.36.0.1, 130.132.1.1

The InterNIC Registration Services Host contains ONLY Internet Information
(Networks, ASN's, Domains, and POC's).
Please use the whois server at nic.ddn.mil for MILNET Information.

> whois -h rs.internic.net yale-dom
Yale University (YALE-DOM)
  Yale University Computing & Information Systems
  Mail Stop 2112
  New Haven, CT 06520

  Domain Name: YALE. EDU

  Administrative Contact:
      Paolillo, Joseph (JP218) joseph_paolillo@yale.edu
      (203) 432-6673
  Technical Contact, Zone Contact:
      Long, Morrow H. (HML1) LONG-MORROW@CS.YALE.EDU
      (203) 432-1254
```

```
Record last updated on 15-Dec-93.
Record created on 17-Mar-87.

Domain servers in listed order:

SERV1.NET.YALE.EDU          130.132.1.9
SERV2.NET.YALE.EDU          130.132.1.10
SERV3.NET.YALE.EDU          130.132.1.11
YALE.EDU                    128.36.0.1, 130.132.1.1
NIC.NEAR.NET                192.52.71.4

The InterNIC Registration Services Host contains ONLY Internet Information
(Networks, ASN's, Domains, and POC's).
Please use the whois server at nic.ddn.mil for MILNET Information.
```

C.4 Идентификаторы регистрации IPv6

Internet Assigned Numbers Authority (IANA) координирует использование адресов IPv6. Текущие идентификаторы регистрации для адресов провайдеров IPv6:

Региональная регистрация	Идентификатор регистрации
Мультирегиональный (IANA)	*10000*
RIPE NCC	01000
INTERNIC	11000
APNIC	10100

C.5 Функции безопасности CERT

Координационный центр CERT (Computer Emergency Response Team — подразделение реагирования на компьютерные неисправности), основанный в 1988 г., располагается в Институте разработки программного обеспечения (Software Engineering Institute — SEI) университета Карнеги-Меллона (Питсбург, Пенсильвания).

CERT публикует заметки о проблемах безопасности, обнаруженных в операционных системах или программных пакетах, и предоставляет руководства по их устранению. CERT координирует действия по защите от взлома сайтов Интернета. Информация CERT доступна по адресу:

http://www.sei.cmu.edu/technology/cert.cc.html

ftp://cert.org/

Связаться с CERT можно через:

CERT Coordination Center
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213-3890

Электронная почта: cert@cert.org

Телефон: + 1-412-268-7090 (24-часовое обслуживание)

Факс: +1-412-268-6989

Рекомендации CERT публикуются в группе новостей:

comp.security.announce

и распространяются через рассылочный почтовый список:

cert-advisory-request@cert.org

Приложение

D

Маски подсети переменной длины

D.1 Введение

Формат адресов Интернета причиняет много хлопот сетевым администраторам хотя бы потому, что 32-разрядное адресное пространство слишком мало и ограничено.

Компьютеры работают с адресами, используя побитовое деление, и вполне могут воспринимать 16-разрядные номера сетей, 7-разрядные номера подсетей и 9-разрядные номера хостов. Но пользователям не очень удобны такие битовые фрагменты.

К еще большему беспорядку приводит *запись байтов* адреса десятичными числами, например 130.15.1.2. Когда границы подсети не попадают в целые байты, требуются некоторые вычисления для выделения из адреса: 1) подсети и 2) адреса хоста.

Это приложение поможет упростить работу с масками подсети, когда они не попадают в байтовые границы. Мы рассмотрим несколько примеров масок подсети для сети класса B с адресом 130.15. В таблице 5.2 приведен полный список масок подсети для сетей этого класса. *Хотя мы не включили в примеры подсети с одними нулями, нужно помнить, что такой вариант успешно используется на многих сайтах.*

D.1.1 Маска подсети из семи бит

Когда в адресной части для подсети меньше 8 бит, можно выбрать вариант с небольшим числом подсетей, но большим количеством хостов. Например:

Биты подсети	Количество подсетей	Биты хоста	Количество хостов
7	128	9	510
6	64	10	1022

Для 7-битовой подсети первый хост в первой подсети имеет адрес (в двоичном представлении и записи с точками):

```
10000010 00001111 00000010 0000001
  130   .   15   .   2    .   1
```

Часть адреса для хостов напечатана полужирным шрифтом. Последний хост первой подсети имеет следующий адрес:

```
10000010 00001111 00000011 1111110
  130   .   15   .   3    .   254
```

Таким образом, в записи с точками первая подсеть имеет адреса:

от 130.15.2.1 до 130.15.2.255
от 130.15.3.0 до 130.15.3.254

Все адреса, начинающиеся с 130.15.2 или 130.15.3, находятся в этой же самой подсети. Допустим и адрес хоста 130.15.2.255. Этот адрес заканчивается *байтом* из одних единиц, но *поле хоста* имеет один 0 в предыдущем байте. Точно так же легален адрес 130.15.3.0, хотя он и завершается всеми нулями, но не имеет нулевого поля хоста.

Вторая подсеть будет включать адреса:

от 130.15.4.1 до 130.15.4.255
от 130.15.5.0 до 130.15.5.254

Принцип уже должен быть ясен. Каждая подсеть будет содержать смежную пару четных и нечетных номеров. Новые подсети начинаются с каждого четного номера.

D.1.2 Маска подсети из шести бит

Рассмотрим 6-битовые подсети. В нашем случае первый хост первой подсети имеет адреса:

10000010 00001111 00000**100 0000001**
130 . 15 . 4 . 1

Последний хост первой подсети имеет адрес:

10000010 00001111 00000**111 1111110**
130 . 15 . 7 . 254

Это означает, что при записи с точками к первой подсети будут относиться адреса:

от 130.15.4.1 до 130.15.4.255
от 130.15.5.0 до 130.15.5.255
от 130.15.6.0 до 130.15.6.255
от 130.15.7.0 до 130.15.7.254

Все адреса, начинающиеся с 130.15.4, 130.15.5, 130.15.6 и 130.15.7, находятся в одной подсети. Как и прежде, допустимы адреса хостов 130.15.4.255, 130.15.5.255 и 130.15.6.255. Эти адреса завершаются *байтом* из всех единиц, но *поле хоста* содержит не только единицы, поскольку имеет ноль в предыдущем байте. Точно так же законны адреса 130.15.5.0, 130.15.6.0 и 130.15.7.0. Хотя они и завершаются нулевым байтом, но не имеют нулевого поля хоста.

Вторая подсеть будет включать адреса:

от 130.15.8.1 до 130.15.8.255
от 130.15.9.0 до 130.15.9.255
от 130.15.10.0 до 130.15.10.255
от 130.15.11.0 до 130.15.11.254

Зависимость прослеживается и в этом случае. Каждая подсеть будет представлена четырьмя смежными номерами. Новые подсети начинаются с номеров, кратных четырем.

В 5-разрядных подсетях первая подсеть будет содержать адреса от 130.15.8.1 до 130.15.15.254, а новые подсети — начинаться с номеров, кратных восьми. Теперь, когда мы разобрались с небольшими масками подсетей, можно перейти к большим.

D.1.3 Подсети из 9-ти бит

Начнем с сети 130.15.1. При 9-битовой подсети ее первый хост будет иметь адрес:

10000010 00001111 00000001 0**0000001**
130 . 15 . 1 . 1

Адрес последнего хоста подсети:

10000010 00001111 00000001 0**1111110**
130 . 15 . 1 . 126

При записи с точками подсеть будет содержать адреса:

от 130.15.1.1 до 130.15.1.126

Первый хост следующей подсети будет иметь адрес:

10000010 00001111 00000001 1**0000001**
130 . 15 . 1 . 129

Последний хост этой подсети сможет адресоваться как:

10000010 00001111 00000001 1**1111110**
130 . 15 . 1 . 254

К подсети будут относиться адреса:

от 130.15.1.129 до 130.15.1.254

Первый хост следующей подсети приобретет адрес:

10000010 00001111 00000010 **00000001**
130 . 15 . 2 . 1

Последний хост следующей подсети получит адрес:

10000010 00001111 00000010 **01111110**
130 . 15 . 2 . 126

Таким образом, к следующей подсети будут относиться адреса:

от 130.15.2.1 до 130.15.2.126

Прослеживается зависимость и в этом случае. Последний байт используется для конструирования двух подсетей, каждая со 126 адресами. Номера хостов для первой из них располагаются в диапазоне от 1 до 126. Номера хостов второй подсети: от 129 до 254.

D.1.4 10-битовые подсети

Начнем с более простого случая для сети 130.15.1. Первый хост будет иметь адрес:

10000010 00001111 00000001 **00000001**
130 . 15 . 1 . 1

Последний хост этой подсети получит адрес:

10000010 00001111 00000001 **00111110**
130 . 15 . 1 . 62

Записанная с точками, подсеть будет содержать 62 адреса:

от 130.15.1.1 до 130.15.1.62

Адрес первого хоста следующей подсети:

10000010 00001111 00000001 **01000001**
130 . 15 . 1 . 65

Последний хост второй подсети:

10000010 00001111 00000001 **01111110**
130 . 15 . 1 . 126

В записи с точками это будет подсеть из 62 адресов:

от 130.15.1.65 до 130.15.1.126

Последний байт служит для конструирования четырех подсетей, из которых каждая будет иметь 62 адреса. Последний байт будет разделен на следующие диапазоны:

от 1 до 62
от 65 до 126
от 129 до 190
от 193 до 254

D.2 Маски подсетей с переменной длиной

Очень трудно выбрать одну-единственную маску подсети для организации. Многие сети предприятий сочетают различное коммуникационное оборудование — линии дальней связи, Frame Relay, локальные сети офиса и мелких подразделений организации. К счастью, сегодня можно присвоить адреса более эффективным способом, используя маски подсетей переменной длины. Другими словами, применение нескольких масок различного размера позволит удовлетворить требования каждой из подсетей организации.

Единственной причиной того, что этот способ не применялся ранее, было отсутствие пересылки информации о масках подсетей между маршрутизаторами в старых протоколах маршрутизации. Например, классический маршрутизатор протокола RIP обеспечивал обмен сообщениями со следующим содержанием:

- Сеть назначения, подсеть или хост
- Метрика счетчика попадания до точки назначения

Элементы таблиц маршрутизации не содержали никакой информации о масках подсетей. Реализации учитывают лишь ситуацию, когда во всей сети используется единственная маска. Организации с адресом класса B обычно выбирали 8 бит для номеров подсетей и 8 бит для номеров хостов, что навсегда ограничивало их 254 подсетями по 254 хоста в каждой.

RIP версии 2, Open Shortest Path First (OSPF), и Cisco Enhanced Internet Gateway Routing Protocol (EIGRP) поддерживают маски переменной длины. Это означает, что маршрутизаторы включают в описание каждой точки назначения маску подсети.

Мы продолжим рассматривать пример сети класса B (130.15.0.0). Самый легкий способ работать с масками переменной длины — это отделить диапазоны номеров для каждого размера.

D.2.1 Присваивание маски линии "точка-точка"

Начнем со связи "точка-точка" (Point-to-Point). Хотя в некоторых сайтах не присваивают IP-адреса линиям "точка-точка", многие маршрутизаторы обеспечивают такую возможность, и мы рассмотрим сначала именно этот вариант. Для любой цепи "точка-точка" необходимо только два адреса. 14-битовая маска будет наиболее пригодной для этого случая. Если адреса начинаются от 130.15.251, то мы получаем 64 подсети:

от 130.15.251.1 до 130.15.251.2
от 130.15.251.5 до 130.15.251.6
от 130.15.251.9 до 130.15.251.10
...
от 130.15.250.253 до 130.15.250.254

Если же использовать 14-битовые маски для всех адресов в диапазоне

от 130.15.251.0 до 130.15.255.255

то мы получим пятикратное увеличение, т.е. 320 подсетей.

D.2.2 Локальная сеть малого офиса

Предположим, что организация имеет 100 филиалов и каждому из них требуется от 30 до 40 адресов. Чтобы обезопасить себя, выбираем 10-битовую маску подсети, которая поддержит 62 хоста на каждом сайте. Для адресов от 130.15.101 мы получим четыре подсети:

от 130.15.101.1 до 130.15.101.62
от 130.15.101.65 до 130.15.101.126
от 130.15.101.129 до 130.15.101.190
от 130.15.101.193 до 130.15.101.254

Если требуется 100 подсетей, нужно применить 10-битовые маски к диапазону адресов:

от 130.15.101.0 до 130.15.125.255

Следует зарезервировать несколько больший диапазон, чтобы обеспечить добавление сайтов в будущем.

D.2.3 Большая локальная сеть

Наконец, предположим, что существует шесть больших локальных сетей. Мы хотим обеспечить соединение каждой из них с 500 хостами. Подойдет 7-битовая маска подсети (см. раздел D.1.1). Типичная 7-битовая подсеть содержит диапазон адресов, подобный следующим:

от 130.15.2.1 до 130.15.2.255
от 130.15.3.0 до 130.15.3.254

Если нужно 6 таких локальных сетей, можно применить 7-битовые маски к диапазону:

от 130.15.2.0 до 130.15.13.255

Лучше резервировать больший диапазон, чтобы учесть будущие потребности.

D.2.4 Резюме

Маски переменной длины поддерживают эффективное выделение IP-адресов. Первым шагом в их применении должен быть анализ сети и определение необходимых размеров. Затем выделяется диапазон номеров для использования с каждым размером маски. Полезно оставлять небольшие промежутки между диапазонами адресов, чтобы учесть будущее расширение сети.

Библиография

Albitz, Paul, and Cricket Liu, *DNS and BIND*, O'Reilly & Associates, Sebastopol, Calif., 1993.

American National Standards Institute, *Fiber Distributed Data Interface (FDDI) — Token-Ring Physical Layer Protocol (PHY)*, ANS X3. 148- 1988, (also ISO 9314-1, 1989).

———, *Fiber Distributed Data Interface (FDDI – Token-Ring Media Access Control (MAC)*, ANS X3.139-1987, (also ISO 9314-2, 1989).

———, *T1.602 – Telecommunications – ISDN – Data Link Layer Signalling Specification for Application at the Network Interface*, 1990.

———, *T1.606 – Frame Relaying Bearer Service – Architectural Framework and Service Description*, 1990.

———, *T1SI/90-175 – Addendum to.696 – Frame Relaying Bearer Service – Architectural Framework and Service Description*, 1990.

———, *T1SI/90-214 – DSSI – Core Aspects of Frame Protocol for Use with Frame Relay Bearer Service – Architectural Framework and Service Description*, 1990.

Bellcore TA-TSV-00160, *Exchange Access SMDS Service Generic Requirements*, December 1990.

Bellovin, S., and M. Merritt, "Limitations of the Kerberos Authentication System," *Computer Communications Review*, October 1990.

Black, Uyless D., "Data Communications," *Networks, and Distributed Processing*, Reston, 1983.

Bolt, Beranek, and Newman, A History of the ARPANET: *The First Decade*, Technical Report, 1981.

Borman, D., "Implementing TCP/IP on a Cray Computer," *Computer Communication Review*, April 1989.

Brand, R., *Coping with the Threat of Computer Security Incidents: A Primer from Prevention through Recovery*, at cert.sei.cmu.edu in|pub|info|primer, June 1990.

Callon, Ross, "An Overview of OSI NSAP Addressing in the Internet," *ConneXions, The Interoperability Report*, December 1991.

CCITT Recommendation I.22, *Framework for providing additional packet mode bearer services*, Blue Book, ITU, Geneva, 1988.

CCITT Recommendation X.25, *Interface between data terminal equipment (DTE) and data-cir-cuit-terminating equipment (DCE) for terminals operating in the packet mode on public data networks*, 1980 and 1984.

CCITT Recommendation X.400, *Message Handling System*, 1984 and 1988.

CCITT Recommendation X.500, *The Directory*, 1988.

Cerf, V., "A History of the ARPANET," *ConneXions, The Interoperability Report*, October 1989.

——— and R. Kahn, "A Protocol for Packet Network Intercommunication," *IEEE Transactions on Communication*, May 1974.

Cheswick, B., "The Design of a Secure Internet Gateway," *Proc. Of the Summer Usenix Conference*, Anaheim, Calif., June 1990.

Cheswick, William R., and Steven M. Bellovin, *Firewalls and Internet Security*, Addison-Wesley, Reading, Mass., 1994.

Cisco Systems, StrataCom, Digital Equipment Corporation, *Frame Relay Specification with Extensions*, Draft, 1990.

Cisco Systems, *Gateway System Manual*, 1991.

Coltun, Rob, "OSPF: An Internet Routing Protocol," *ConneXions*, August 1989.

Comer, Douglas E., *Internetworking with TCP / IP, Volume I, Principles, Protocols, and Architecture*, 2d ed., Prentice-Hall, Englewood Cliffs, N. J., 1991.

——— and David L. Stevens, *Internetworking with TCP / IP. Volume II. Design, Implementation, and Internals*, Prentice-Hall, Englewood Cliffs, N. J., 1991.

Cooper, J., *Computer and Communications Security: Strategies for the 1990s*, McGraw-Hill, New York, 1989.

Deering, S., "IP Multicasting," *ConneXions*, February 1991.

Dern, Daniel P., "Standards for Interior Gateway Routing Protocols," *ConneXions*, July 1990.

Digital Equipment Corporation, Intel Corporation, and XEROX Corporation, *The Ethernet: A Local Area Network Data Link Layer and Physical Layer Specification*, September 1980.

Frey, Donnalyn, and Rick Adams, *!%@:: A Directory of Electronic Mail Addressing and Networks*, 2d ed., O'Reilly & Associates, Sebastopol, Calif., 1989.

FRICC, *Program Plan for the National Research and Education Network*, Federal Research Internet Coordinating Committee, U. S. Department of Energy, Office of Scientific Computing Report ER-7, May 1989.

FTP Software, *PC / TCP Kernel Installation and Reference Guide*, Version 2.05 for DOS, 1990.
———, *PC / TCP User's Guide*, Version 2.05 for DOS, 1990.
Garcia-Luna-Aceves, J. J., *A Unified Approach to Loop-Free Routing using Distance Vectors or Link States*, ACM 089791-332-9/89/0009/0212, pp. 212-223, 1989.
———, "Loop-Free Routing using Diffusing Computations," *IEEE / ACM Transactions on Networking*, vol. 1, no. 1, 1993.
GOSIP, *U. S. Government Open Systems Interconnection Profile Version 2.0*, Advanced Requirements Group, National Institute of Standards and Technology (NIST), April 1989.
Green, James Harry, *The Dow Jones-Irwin Handbook of Telecommunications*, Dow Jones-Irwin, Homewood, I11., 1986.
Hedrick, Charles L., *Introduction to Administration of an Internet-based Local Network*, Rutgers, The State University of New Jersey, 1988, at cs.rutgers.edu, in/runet/tcp-ip-admin. doc.
———, *Introduction to the Internet Protocols*, Rutgers, The State University of New Jersey, 1987, host cs.rutgers.edu, /runet/tcp-ip- intro.doc.
Hoffman, L., *Rogue Programs: Viruses, Worms, and Trojan Horses*, Van Nostrand Reinhold, New York, 1990.
Huitema, Christian, "Routing in the Internet,"Prentice-Hall PTR, Englewood Cliffs, N. J., 1995.
IBM GG24-3442, *IBM AS / 400 TCP / IP Configuration and Operation*, 1991.
IBM GG24-3696, *Managing TCP / IP Networks Using NetView and the SNMP Interface*, 1991.
IBM GG24-3816, *High-Speed Networking Technology, An Introductory Survey*, 1992.
IBM SC31-6081, *TCP / IP Version 2 Release 2 for VM: User's Guide*, 1991.
IBM SC31-6084, *TCP / IP Version 2 Release 2 for VM: Programmer's Reference*, 1991.
IBM, *Vocabulary for Data Processing, Telecommunications, and Office Systems*, 1981.
Institute of Electrical and Electronics Engineers, *Draft Standard P802. IA – Overview and Architecture*, 1989.
———, *Local Area Networks – CSMA / CD Access Method*, ANSI/IEEE 802.3, (ISO 8802-3).
———, *Local Area Networks – Distributed Queue Dual Bus (DQDB) Subnetwork of a Metropolitan Area Network (MAN)*, ANSI/IEEE 802.6 (ISO DIS 8802-6, 1991).
———, *Local Area Networks – Higher Layers and Interworking*, ANSI/IEEE 802.1, 1990 (ISO DIS 8802-1D, 1990).
———, *Local Area Networks – Logical Link Control*, ANSI/IEEE 802.2, 1989 (ISO 8802-2, 1989).
———, *Local Area Networks – Network Management*, Draft IEEE 802. 1 B, 1990.
———, *Local Area Networks – Token-Bus Access Method*, ANSI/IEEE 802.4, (ISO 8802-3).
———, *Local Area Networks – Token Ring Access Method*, ANSI/IEEE 802.5, 1989 (ISO 8802-5,1989).
International Organization for Standardization, *Information Processing Systems — Common Management Information Protocol (CMIP)*, ISO 9596, 1990.
———, *Information Processing Systems — Common Management Information Service (CMIS)*, ISO 9595, 1990.
———, *Information Processing Systems – Data Communications – Addendum to the Network Service Definition*, ISO 8348 ADI.
———, *Information Processing Systems – Data Communications – High-Level Data Link Control Procedures — Consolidation of Classes of Procedures*, ISO 7809.
———, *Information Processing Systems – Data Communications – High-Lever Data Link Control Procedures — Consolidation of Elements of Procedures*, ISO 4335.
———, *Information Processing Systems – Data Communications – High-Lever Data Link Control Procedures — Frame Structure*, ISO 3309.
———, *Information Processing Systems – Data Communications – Network Service Definition*, ISO 8348.
———, *Information Processing Systems – Data Communications – Protocol for Providing the Connectionless-Mode Network Service*, ISO 8473.
———, *Information Processing Systems – Open Systems Interconnection – Basic Connection Oriented Session Protocol Specification*, ISO 8327.
———, *Information Processing Systems – Open Systems Interconnection – Basic Connection Oriented Session Service Definition*, ISO 8326.
———, *Information Processing Systems – Open Systems Interconnection – Connection Oriented Presentation Protocol Specification*, ISO 8823.
———, *Information Processing Systems – Open Systems Interconnection – Connection Oriented Presentation Service Definition*, ISO 8822.
———, *Information Processing Systems – Open Systems Interconnection – Connection Oriented Transport Protocol*, ISO 8073.
———, *Information Processing Systems – Open Systems Interconnection – Intermediate System to Intermediate System Intra-Domain Routing Exchange Protocol for use in Conjunction with the Protocol for Providing the Connectionless-Mode Network Service*, ISO DIS 10589.
———, *Information Processing Systems – Open Systems Interconnection – Message Handling System*, ISO 10021/CCITT X.400.
———, *Information Processing Systems – Open Systems Interconnection – Protocol Specification for the Association Control Service Element*, ISO 8650.
———, *Information Processing Systems – Open Systems Interconnection – Remote Operations: Model, Notation, and Service Definition*, ISO 9072-1.
———, *Information Processing Systems – Open Systems Interconnection – Remote Operations: Protocol Specification*. ISO 9066-2.
———, *Information Processing Systems – Open Systems Interconnection – Service Definition for the Association Control Service Element*, ISO 8649.
———, *Information Processing Systems – Open Systems Interconnection – Specification of Abstract Syntax Notation One (ASN.1)*, ISO 8824.
———, *Information Processing Systems – Open Systems Interconnection – Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)*, ISO 8825.

———, *Information Processing Systems – Open Systems Interconnectin – Transport Service Definition*, ISO 8072.
———, *OSI Routing Framework*, ISO TC97/SC6/N4616, June 1987.
Jacobson, V.,"Berkeley TCP Evolution from 4.3-Tahoe to 4.3-Reno," *Proceedings of the Eighteenth Internet Engineering Task Force.*
———, "Congestion Avoidance and Control," *ACM SIGCOMM-88*, August 1988.
Jain, R., K. Ramakrishnan, and D-M Chiu, *Congestion Avoidance in Computer Networks With a Connectionless Network Layer*, Technical Report, DEC-TR-506, Digital Equipment Corporation, 1987.
Kapoor, Atul, *SNA, Architecture, Protocols, and Implementation*, McGraw-Hill, New York, 1992.
Karn, P., and C. Partridge, "Improving Round-Trip Time Estimates in Reliable Transport Protocols," *Proceedings of the ACM SIGCOMM*, 1987.
Kernighan, Brian W., and Dennis M. Ritchie, *The C Programming Language*, 2d ed., Prentice-Hall, Englewood Cliffs, N. J., 1988.
Kessler, Gary C., and Train, David A., *Metropolitan Area Networks*, McGraw-Hill, New York, 1992.
———, *ISDN*, McGraw-Hill, New York, 1990.
Kochan, Stephen G., and Patrick H. Wood (consulting eds.), *UNIX Networking*, 1989.
Laquey, T. L., *User's Directory of Computer Networks*, Digital Press, Bedford, Mass., 1989.
Lippis, Nick, and James Herman, "Widening Your Internet Horizons," *ConneXions*, October 1991.
Liu, Cricket, Jerry Peek, Russ Jones, Bryan Buus, and Adrian Nye, *Managing Internet Information Services*, O'Reilly & Associates, Sebastopol, Calif., 1995.
Malamud, Carl, *DEC Networks and Architectures*, McGraw-Hill, New York, 1989.
———, *STACKS — The INTEROP Book*, Prentice-Hall, Englewood Cliffs, N. J., 1991.
McKenney, P., "Congestion Avoidance," *ConneXions*, February 1991.
Medin, Milo, "The Great IGP Debate – Part Two: The Open Shortest Path First (OSPF) Routing Protocol," *ConneXions*, October 1991.
Mills, D., and H-W. Braun, "The NSFNET Backbone Network," *Proceedings of the ACM SIGCOMM*, 1987.
Mogul, Jeffrey C., "Efficient Use Of Workstations for Passive Monitoring of Local Area Networks," *Proceedings of SIGCOMM '90 Symposium on Communications Architectures and Protocols*, September 1990.
Narten, T., "Internet Routing," *Proceedings of the ACM SIGCOMM*, 1989.
Nemeth, Evi, Garth Snyder, and Scott Seebass, *UNIX System Administration Handbook*, Prentice-Hall, Englewood Cliffs, N. J., 1989.
Odlyzko, A. M., "The future of integer factorization," *CryptoBytes* (The technical newsletter of RSA Laboratories), 1994.
Perlman, Radia, and Ross Callon, "The Great IGP Debate – Part One: IS-IS and Integrated Routing," *ConneXions*, October 1991.
Pfleeger, C., *Security in Computing*, Prentice-Hall, Englewood Cliffs, N. J., 1989.
Postel, J. B., "Internetwork Protocol Approaches," *IEEE Transactions on Communications*, 1980.
———, C. A. Sunshine, and D. Chen, "The ARPA Internet Protocol," *Computer Networks*, vol. 5, no. 4, July 1981.
Quarterman, John S., "The Matrix," *Computer Networks and Conferencing Systems Worldwide*, Digital Press, Bedford, Mass., 1990.
——— and Hoskins, J. C., "Notable Computer Networks," *Communications of the ACM*, October, 1986.
Romkey, John, "The Packet Driver," *ConneXions*, July 1990.
Rose, Marshall T., *The Little Black Book: Mail Bonding with OSI Directory Services*, Prentice-Hall, Englewood Cliffs, N. J., 1990.
———, *The Open Book: A Practical Perspective on OSI*, Prentice-Hall, Englewood Cliffs, N. J., 1990.
———, *The Simple Book: An Introduction to Management of TCP / IP-based Internets*, Prentice-Hall, N. J., 1990.
Sackett, George C., *IBM's Token-Ring Networking Handbook*, McGraw-Hill, New York, 1993.
St. Amand, Joseph V., *A Guide to Packet-Switched, Value-Added Networks*, Macmillan, New York, 1986.
Schwartz, Michael F., "Resource Discovery and Related Research at the University of Colorado," *ConneXions*, May 1991.
Seeley, D., "A Tour of the Worm," *Proceedings of 1989 Winter USENIX Conference*, Usenix Association, San Diego, Calif., February 1989.
Sijan, Karanjit, and Hare, Chris, *Internet Firewalls and Network Security*, New Riders Publishing.
Simmons, G. J., ed., *Contemporary Cryptology*, IEEE, 1991.
Spafford, E., "The Internet Worm Program: An Analysis, " *Computer Communication Review*, vol. 19, no. 1, ACM SIGCOMM, January 1989.
Stallings, William, *Data and Computer Communications*, Macmillan, New York, 1984.
———, *Handbook of Computer Communications Standards*, Department of Defense Protocol Standards, 1988.
Stern, Hal, *Managing NFS and NIS*, O'Reilly and Associates, Sebastopol, Calif., 1991.
Stevens, W. Richard, *TCP / IP Illustrated*, vol. 1, Addison Wesley, Reading, Mass., 1994.
———, *UNIX Network Programming*, Prentice-Hall, Englewood Cliffs, N. J., 1990.
Stoll, C., *The Cuckoo's Egg*, Doubleday, New York, 1989.
Tannenbaum, Andrew S., Computer Networks, Prentice-Hall, Englewood Cliffs, N. J., 1981.
Vitalink, *Building and Managing Multivendor Networks using Bridge and Router Technologies*, 1990.
Tsuchiya, Paul F., "Inter-domain Routing in the Internet," *ConneXions*, January 1991.
XEROX, *Internet Transport Protocols, Report XSIS 028112*, Xerox Corporation, 1981.
X/Open specification, *X/Open CAE Specification: Protocols for X/Open Internetworking: XNFS*, X/Open Company, Ltd., 1991.

Глоссарий

Abstract Syntax Notation One (ASN.1) Первая абстрактная синтаксическая нотация. Язык для определения типов данных ASN.1. Используется в стандартах OSI и спецификациях TCP/IP для управления сетью.

Access Control Управление доступом. Возможность указать привилегию конечного пользователя для доступа к компьютерным данным.

Acknowledgment Подтверждение. Протокол TCP требует, чтобы данные были подтверждены, прежде чем они будут рассматриваться как благополучно переданные по сети.

Active Open Активное открытие. Действие приложения для инициализации соединения TCP.

Address Mask Адресная маска. 32-разрядный двоичный номер для идентификации части IP-адреса, используемой для номера сети или подсети. Каждый бит в полях сети и подсети установлен в 1.

Address Resolution Protocol (ARP) Протокол разрешения адресов. Протокол динамического исследования физического адреса системы по заданному IP-адресу.

Agent Агент. В протоколе Simple Network Management Protocol процесс в устройстве, ответственный за получение и отправление запросов, равно как и отправление сообщений-прерываний.

American National Standards Institute (ANSI) Американский национальный институт стандартов. Организация, ответственная за координацию стандартизации в США. Является членом ISO.

AppleTalk Сетевой протокол, разработанный компанией Apple Computer для своего оборудования.

Application Programming Interface (API) Интерфейс программирования приложений. Набор программ для расширения возможностей компьютера. В программировании для TCP/IP применяются API: socket и Transport Layer Interface.

Archie Сервер, который собирает и индексирует место размещения файлов в общедоступных архивах пересылки, а также поддерживает пользовательские поисковые средства.

ARPANET Первая в мире сеть с коммутацией пакетов, многие годы использовавшаяся как магистраль Интернета.

ASCII American National Standard Code for Information Interchange (Американский национальный стандартный код для информационного обмена). Для определения символа ASCII требуется семь из восьми битов октета.

Asynchronous Transfer Mode Асинхронный режим пересылки. Технология на основе коммутации для пересылки информации в 53-октетных ячейках. ATM может использоваться для пересылки данных, аудио и видео.

Authentication Аутентификация (проверка подлинности). Идентификация партнера по коммуникации.

Authentication Header (AH) Аутентификационный заголовок. Заголовок уровня IP, описывающий источник данных и обеспечивающий целостность пересылки данных. Обычно AH вставляется после главного заголовка IP перед аутентифицируемой информацией.

Autonomous System (AS) Автономная система. Набор маршрутизаторов, управляемый одним администратором и использующий общий протокол Interior Gateway Protocol. Раньше определялась как одна или несколько сетей, имеющих единую политику внешней маршрутизации.

Bandwidth Полоса пропускания, доля производительности сетевого носителя. Количество данных, которые могут быть посланы по сетевой связи. Обычно измеряется в битах за секунду.

Basic Encoding Rules (BER) Базовые правила кодирования в формат пересылки типов данных, специфицированных по ASN.1.

Baud Бод. Единица скорости передачи сигналов, равная количеству изменений сигнала за секунду. Для цифровых сигналов (с двумя состояниями) бод равен бит/с.

Berkeley Software Distribution (BSD) Программный дистрибутив Беркли. Программное обеспечение Unix от Калифорнийского университета в Беркли, имеющее поддержку TCP/IP.

Best Current Practices (BCP) Лучший текущий способ применения. Классификация полезного RFC, который не определяет стандарт протокола.

Big Endian Стиль "тупоконечников". Формат для хранения или пересылки данных, в котором наиболее значимый байт (или бит) стоит первым.

BIND Software Программное обеспечение BIND. Программный сервер именования доменов от Калифорнийского университета Беркли.

Bootstrap Protocol (BOOTP) Загрузочный протокол. Используется для загрузки в системы сетевой конфигурационной информации.

Border Gateway protocol (BGP) Протокол граничного шлюза. Протокол объявления о нескольких сетях, которых можно достичь в пределах автономной системы. BGP обеспечивает совместное использование этой информации несколькими автономными системами. BGP заменяет более старый протокол EGP и предлагает множество улучшений.

Bounce Отбрасывание, возвращение назад. Возвращение части почты, которая не может быть доставлена в точку назначения.

Bridge Мост. Устройство, которое соединяет два или более физических сегментов локальной сети и пересылает кадры, имеющие адреса источника и назначения в различных сегментах.

Broadcast Широковещательная рассылка. Кадр связи, адресованный всем системам одной сетевой связи.

Brouter Мост-маршрутизатор. Устройство, объединяющее функции моста и маршрутизатора. Некоторая часть трафика выбирается для маршрутизации, а оставшаяся часть обслуживается мостом.

Buffer Буфер. Область хранения входных или выходных данных.

Canonical Name Каноническое имя. Уникальное истинное имя хоста.

Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Множественный доступ с контролем несущей и обнаружением конфликтов. Простой протокол Media Access Control (управления доступа к носителю). Все станции прослушивают носитель. Станция, желающая переслать данные, ожидает освобождения носителя. Когда две стации передают данные одновременно, обе пересылки отменяются и возобновляются через случайный интервал времени.

Cipher-Block Chaining Формирование цепочки малозначимых блоков. Популярный вариант шифрования DES. Блок уже зашифрованных данных используется в алгоритме шифрования следующего блока данных.

Classless Inter-Domain Routing (CIDR) Бесклассовая междоменная маршрутизация. Метод, позволяющий сетевой части IP-адреса состоять из определенного количества бит.

Common Management Information Protocol (CMIP) Общий протокол управления информацией. Главный протокол сетевого управления в OSI.

Common Management Information Services and Protocol over TCP/IP (CMOT) Общие протокол и службы управления информацией поверх TCP/IP. Историческая (не рекомендуемая) спецификация для использования протоколов управления OSI в сетях TCP/IP.

Confidentiality Конфиденциальность. Защита информации от взлома.

Connection Соединение. Логическая коммуникация между пользователями TCP.

Core Gateway Основной шлюз. Исторически — маршрутизатор магистрали Интернета. Основные шлюзы распространяли сведения о достижимости среди автономных систем, подключенных к магистрали Интернета.

Cracker Взломщик. Некто, пытающийся проникнуть в компьютерную систему, часто с преступными намерениями.

Cyclic Redundancy Check (CRC) Циклическая избыточная проверка. Значение, полученное в результате выполнения математической функции над битами кадра и добавленное в конец этого кадра. CRC повторно вычисляется при получении кадра. Если результат отличается от добавленного в конец значения, кадр отбрасывается.

Data Circuit-terminating Equipment (DCE) Оборудование терминирования цепи данных. Оборудование для соединения DTE с линией или сетью.

Data Encryption Standard (DES) Стандарт шифрования данных. Симметричный протокол шифрования, официально санкционированный правительством США. Существуют различные варианты DES (см. Cipher Block Chaining).

Data Terminal Equipment (DTE) Оборудование терминирования данных. Источник или точка назначения для данных. Часто обозначает терминал или компьютер, подключенный к региональной сети.

DECnet Лицензированный протокол компании Digital Equipment Corporation. Версии протокола именуются номерами фаз, например Phase IV и Phase V.

Directory Access Protocol (DAP) Протокол доступа к каталогам. Клиент/серверный протокол для доступа к службе каталогов X. 500.

Directory System Agent (DSA) Системный агент каталога. Сервер, который принимает запросы от пользовательского агента каталогов (Directory User Agent) и извлекает информацию из базы данных. DSA взаимодействует с пользовательским агентом каталогов по протоколу X. 500 Directory Access Protocol.

Directory User Agent (DUA) Пользовательский агент каталогов. Клиент, позволяющий пользователю отправлять запросы к серверу каталогов X.500. DUA взаимодействует с DSA по протоколу X.500 Directory Access Protocol.

Distributed Computing Environment (DCE) Распределенное вычислительное окружение (среда). Набор технологий, избранных Open Software Foundation для поддержки распределенных вычислений.

Distributed File Service (DFS) Распределенная файловая служба. Одобренная Open Software Foundation технология файлового сервера.

Distributed Management Environment (DME) Распределенное управляющее окружение (среда). Набор технологий, избранных Open Software Foundation для управления сетями и системами.

DIX Ethernet Версия Ethernet, разработанная компаниями Digital, Intel и Xerox.

Domain Name System (DNS) Система именования доменов. Множество распределенных баз данных, обеспечивающих информацию, подобную трансляции имени системы в IP-адрес или в сведения о месте размещения сервера почтового обмена.

DS1 Кадр и спецификация интерфейса для синхронных линий T1.

DS3 Кадр и спецификация интерфейса для синхронных линий T3.

Encryption Шифрование. Преобразование информации в форму, которая не может быть понята без владения секретным ключом (ключом шифрования).

Encapsulating Security Payload (ESP) Инкапсуляция защищенной полезной нагрузки. Протокол обеспечения конфиденциальности датаграмм IP (по выбору, аутентификации и целостности данных). ESP может использоваться между парой хостов, между парой маршрутизаторов или между хостом и маршрутизатором.

Exterior Gateway Protocol (EGP) Протокол граничного шлюза. Маршрутизаторы соседних автономных систем используют этот протокол для опознания множества сетей, которые могут быть достигнуты в пределах автономных систем или через каждую из них. EGP понемногу вытесняется протоколом BGP.

eXternal Data Representation (XDR) Внешнее представление данных. Разработанный компанией Sun Microsystems стандарт описания типов данных, использующий их как параметры при кодировании данных перед пересылкой.

Fiber Distributed Data Interface (FDDI) Волоконно-оптический интерфейс распределенных данных. Стандарт для высокоскоростной пересылки данных по двойному волоконно-оптическому кольцу.

File Transfer, Access and Management (FTAM) Пересылка файлов, доступ и управление. Протокол OSI для пересылки данных и сетевого управления. FTAM разрешает пользователям копировать целые файлы или части файлов, например отдельные записи.

File Transfer Protocol (FTP) Протокол пересылки файлов. Протокол TCP/IP, разрешающий пользователям копировать файлы между системами и выполнять файловые операции, например переименование или удаление файлов.

Finger Дословно — перст. Программа для вывода информации об одном или нескольких удаленных пользователях.

Flow Control Управление потоком. Механизм, позволяющий приемнику ограничивать количество данных, пересылаемых отправителем за единицу времени. Предотвращает переполнение буферов памяти приемника.

For Your Information (FYI) Для вашего сведения. Набор документов, содержащих полезную информацию, подобную ответам на часто задаваемые вопросы о стеке TCP/IP. FYI публикуются как документы RFC.

Fragmentation Фрагментация. Деление датаграммы на части. Выполняется, когда датаграмма слишком велика для данной сетевой технологии и не может обычным способом достичь точки назначения.

Frame Кадр. Элемент данных протокола уровня связи данных.

Frame Check Sequence (FCS) Контрольная последовательность кадра. Математическая функция, применяемая к битам кадра, результат которой добавляется в конец кадра. FCS повторно вычисляется при получении кадра. Если результат будет отличаться от добавленного в конец кадра значения, то такой кадр отбрасывается.

Frequently Asked Questions (FAQ) Часто задаваемые вопросы. Документ в форме вопросов и ответов, который обобщает информацию для группы новостей или рассылочного списка.

Gateway Шлюз. Маршрутизатор IP. Многие документы RFC используют термин "*шлюз*" вместо термина "*маршрутизатор*".

Gateway-to-Gateway Protocol (GGP) Протокол межшлюзового обмена. Ранее использовался для обмена информацией между основными маршрутизаторами Интернета.

Gopher Протокол, обеспечивающий доступ клиентов к серверу посредством серии меню.

Government Open Systems Interconnection Profile (GOSIP) Правительственный профиль взаимодействия открытых систем. Спецификация набора протоколов OSI, рекомендованных для компьютерного оборудования правительственных учреждений.

Graphics Interchange Format (GIF) Формат обмена графикой. Популярный формат для графических файлов изображений.

High Level Data Link Control Protocol (HDLC) Протокол управления связями данных высокого уровня. Стандарт, являющийся основой для нескольких протоколов уровня связи данных.

High Performance Parallel Interface (HIPPI) Высокопроизводительный параллельный интерфейс. Высокоскоростная коммуникационная технология, описанная в стандарте ANSI. Устройства связываются по HIPPI на небольшие расстояния при скорости обмена 800 или 1600 Мбит/с.

Hypertext Markup Language (HTML) Язык разметки гипертекста. Служит для записи гипертекстовых документов, в которых тегами определяются элементы форматирования, например заголовки, абзацы или списки.

Initial Sequence Number (ISN) Начальный порядковый номер. Определяется во время установки соединения TCP. Посылаемые по соединению октеты данных будут нумероваться от ISN.

Integrated Services Digital Network (ISDN) Цифровые сети с интеграцией служб. Телефонная технология пересылки оцифрованных данных и речевых сигналов.

Interior Gateway Protocol (IGP) Протокол внутреннего шлюза. Любой протокол маршрутизации, используемый внутри автономной системы.

Intermediate System to Intermediate System Protocol (IS-IS) Взаимодействие промежуточных систем. Протокол для маршрутизации трафиков OSI и IP.

International Organization for Standardization (ISO) Международная организация стандартизации. Учреждение для развития международной торговли и совместных исследований в науке и технике.

International Telecommunications Union (ITU) Международный телекоммуникационный союз. Учреждение по координации работы национальных организаций по коммуникационным стандартам и сотрудничеству в этой области.

International Telecommunications Union Telecommunication Standardization Sector (ITU-T) Сектор телекоммуникаций ITU. Осуществляет контроль над исследовательскими группами и публикует "Рекомендации" для международных коммуникационных стандартов. Прежде назывался CCITT.

International Telegraph and Telephone Consultative Committee (CCITT) Международный консультативный комитет по телеграфии и телефонии. Старое название организации, созданной для упрощения объединения средств обслуживания коммуникаций в международные сети.

internet интернет. Несколько сетей, связанных маршрутизаторами IP и воспринимаемых пользователями как единая сеть.

Internet Интернет. Самая большая всемирная сеть. Работа Интернета основана на стеке протоколов TCP/IP.

Internet Architecture Board (IAB) Совет по архитектуре Интернета. Ранее назывался Internet Activities Board. Группа сообщества Интернета (Internet Society), ответственная за разработку и отбор протоколов для использования в Интернете и за назначение состояния и статуса для уже принятых протоколов.

Internet Assigned Numbers Authority (IANA) Авторизация присвоенных номеров Интернета. Организация, ответственная за управление присваиванием значений различных параметров, например общеизвестных портов, адресов многоадресной рассылки, идентификаторов терминалов и идентификаторов систем.

Internet Control Message Protocol (ICMP) Протокол управляющих сообщений Интернета. Необходим для реализации IP. ICMP определяет сообщения об ошибках, которые будут посланы при отбрасывании датаграммы или при обнаружении системной перегрузки. Кроме того, ICMP обеспечивает несколько полезных служб запросов.

Internet Engineering Notes (IEN) Инженерные заметки Интернета. Исходный набор документов о возможностях стека TCP/IP. Эти документы доступны по интерактивным запросам к Network Information Center (NIC).

Internet Engineering Steering Group (IESG) Управляющая группа технологии Интернета. Координирует действия рабочих групп IETF и выполняет техническое рецензирование разрабатываемых стандартов.

Internet Engineering Task Force (IETF) Рабочая группа технологии Интернета. Несколько рабочих групп из добровольцев, развивающих и реализующих протоколы Интернета.

Internet Group Management Protocol (IGMP) Протокол управления группами Интернета. Часть спецификации многоадресной рассылки. IGMP используется для пересылки информации о членстве в группе.

Internet Protocol (IP) Протокол интернета. Протокол уровня 3 из стека TCP/IP, ответственный за транспортировку датаграмм через интернет.

Internet Research Task Force (IRTF) Рабочая группа исследования Интернета. Руководимая IAB группа по долговременным исследованиям в области протоколов Интернета.

Internet Service Provider (ISP) Провайдер, провайдер Интернета, поставщик сетевых услуг. Организация, оказывающая коммерческие услуги по соединению с Интернетом.

Internet Society (ISOC) Сообщество Интернета. Международная организация, созданная для расширения и технического усовершенствования Интернета.

IP Address IP-адрес. 32-разрядное число, идентифицирующее сетевой интерфейс.

IP Datagram Датаграмма IP. Единица данных, маршрутизируемая в IP.

IP Security Option Варианты безопасности в IP. В версии 4 необязательное поле в заголовке IP, содержащее метку безопасности. Этот вариант был разработан для военных и правительственных учреждений.

ISO Development Environment (ISODE) Окружение разработки ISO. Исследования по обеспечению работы протоколов OSI поверх TCP/IP.

Joint Photographic Experts Group (JPEG) Объединенная группа экспертов по фотографии. Спецификация алгоритма сжатия изображения.

Kerberos Служба аутентификации, разработанная в Массачусетском технологическом институте. Kerberos использует шифрование для скрытия паролей от злоумышленников и исключения неправомочного доступа к файлам или службам.

Link Связь. Носитель (среда), через который взаимодействуют сетевые узлы, использующие протокол связи данных.

Little Endian Стиль "остроконечников". Формат для хранения или пересылки данных, когда менее значительный байт (или бит) располагается первым.

Local Area Network (LAN) Локальная сеть (ЛС). Сеть, предназначенная для обслуживания области в несколько квадратных километров (или менее) и состоящая из одной подсети.

Logical Byte Логический байт. Точно установленное количество бит. При пересылке файла иногда необходимо точно определить логический размер байта, чтобы сохранить целостность пересылаемых данных.

Logical Link Control (LLC) Управление логической связью. Протокол уровня 2 (связи данных), управляющий обменом данными между двумя системами, которые связаны одним физическим сетевым сегментом или расположены в сегментах, связанных одним или несколькими мостами.

MAC Address MAC-адрес. Физический адрес интерфейса локальной сети.

MAC Protocol Протокол MAC. Протокол Media Access Control (управления доступа к носителю) определяет правила по управлению способностью систем передавать и получать данные из носителя.

Mail Exchanger Служба почтового обмена. Система доставки почты в сеть организации.

Mail Gateway Почтовый шлюз. Система, выполняющая трансляцию между различными протоколами пересылки электронной почты.

Management Information Base (MIB) База управляющей информации. Множество всех определений для объектов управления сети. Также конфигурация, статус и информация о производительности, которая может быть извлечена из сетевого устройства.

Maximum Segment Size Максимальный размер сегмента. Максимально допустимый размер для секции данных любого сегмента, пересылаемого по конкретному соединению.

Maximum Transmission Unit (MTU) Максимальный элемент пересылки. Самая большая датаграмма, которая может быть послана через конкретный сетевой носитель, например Ethernet или Token-Ring.

Media Access Control (MAC) Управление доступом к носителю. Протокол управления доступом станции к сети. Например CSMA/CD определяет правила MAC для посылки и получения данных в локальной сети.

Message Digest 5 (MD5) Резюме сообщения версии 5. Алгоритм, комбинирующий секретный ключ с сообщением или файлом и формирующий 16-октетный ответ. Цель — обнаружить искажение в информации при пересылке.

Message Transfer Agent (MTA) Агент пересылки сообщений. Средство для перемещения сообщений (например, электронной почты) между компьютерами.

Metropolitan Area Network (MAN) Городские сети (не самый удачный термин, но есть еще WAN, LAN и Global network.— *Прим. пер.*). Технология высокоскоростных сетей, охватывающих территорию большого города. Протокол для таких сетей определен в IEEE 802.6.

Multicast IP Address IP-адрес многоадресной рассылки. Может быть принят несколькими хостами. Датаграммы, посланные по такому адресу, доставляются всем хостам группы.

Multihomed Host Многоадресный хост. Хост с несколькими сетевыми интерфейсами и, следовательно, несколькими IP-адресами.

Multipurpose Internet Mail Extensions (MIME) Многоцелевое почтовое расширение Интернета. Расширения для почты Интернета, позволяющие включать в сообщения несколько частей, каждая из которых может содержать различные типы данных, например текст, изображение, звуковые файлы или прикладные данные.

National Education and Research Network (NREN) Национальная сеть для образования и исследований. Разрабатываемая сеть высокой производительности для будущей магистральной основы Интернета.

National Institute of Standards and Technology (NIST) Национальный институт стандартов и технологий. Организация по стандартам США, разрабатывающая коммуникационные стандарты. NIST ранее назывался National Bureau of Standards (Национальное бюро стандартов).

National Science Foundation Network (NSFnet) Сеть национального научного фонда. Используемая как часть современной магистрали Интернета.

Neighbors Сосед, ближайший сосед. Узел, подключенный к той же самой сетевой связи.

NETBIOS Сетевой программный интерфейс и протокол, разработанные для IBM-совместимых персональных компьютеров.

Network Address Сетевой адрес. 32-разрядный адрес IP-системы.

Network File System (NFS) Сетевая файловая система. Набор протоколов, разработанных компанией Sun Microsystems. NFS позволяет клиентам монтировать удаленные каталоги в локальной файловой системе и использовать удаленные файлы как локальные.

Network Information Center (NIC) Сетевой информационный центр. Организация Интернета, присваивающая имена и адреса, а также обеспечивающая другую служебную информацию.

Network Information Service (NIS) Сетевая информационная служба. Набор протоколов, предложенных компанией Sun Microsystems для службы каталогов сетевой информации.

Network Service Access Point (NSAP) Точка доступа к сетевой службе. Идентификатор для разделения хоста OSI и точки транспортного уровня, куда, собственно, и направляется трафик данного хоста.

Network Virtual Terminal (NVT) Сетевой виртуальный терминал. Набор правил, определяющих очень простое взаимодействие с виртуальным терминалом. NVT используется в начале сеанса *telnet*, но далее происходит переход на более сложные правила взаимодействия.

Nonrepudiation Исключение отмены. Способность доказать, что источник послал определенные данные, даже если он позже попробует отрицать этот факт.

Open Shortest Path First (OSPF) Первым открывать кратчайший путь. Прекрасно масштабируемый протокол маршрутизации интернета, распределяющий трафик на основе выбора из нескольких маршрутов по сведениям о топологии интернета.

Open Software Foundation (OSF) Организация открытого программного обеспечения. Консорциум компьютерных компаний по разработке стандартной технологии открытых систем. Технологиями OSF являются пользовательский интерфейс MOTIF и Distributed Computing Environment (DCE).

Open Systems Interconnection (OSI) Взаимодействие открытых систем. Набор стандартов ISO по коммуникации данных.

Packet Пакет. Первоначально — элемент данных в сетях с коммутацией пакетов. В настоящее время — термин, определяющий элемент данных протокола (Protocol Data Unit) для коммуникационного протокола любого уровня.

Packet Assembler/Disassembler (PAD) Сборка/извлечение пакета. Программное обеспечение для преобразования между трафиками потока терминала и форматом пакета X.25.

Page Structure Структура страницы. Организация файла, поддерживаемая в FTP для использования со старыми компьютерами компании Digital Equipment Corporation.

Passive Open Пассивное открытие. Операция сервера TCP/IP для подготовки к получению запросов от клиентов.

Pathname Имя пути. Символьная строка, вводимая пользователем в файловую систему для идентификации файла.

Payload Полезная нагрузка. Информация, переносимая в элементе данных протокола (Protocol Data Unit).

Physical Address Физический адрес. Адрес интерфейса сети.

Point-to-Point Protocol (PPP) Протокол "точка-точка". Протокол для пересылки данных по последовательной линии связи. PPP поддерживает аутентификацию, конфигурацию связи и возможности мониторинга связи, а также обеспечивает мультиплексирование трафика нескольких протоколов в одной связи.

Port Number Номер порта. 2-октетное двоичное число, идентифицирующее высокоуровневый доступ к TCP или UDP.

Post Office Protocol (POP) Протокол почтового офиса. Протокол для загрузки клиенту электронной почты с сервера (обычно в настольную систему).

Protocol Data Unit (PDU) Элемент данных протокола. Основной термин для единицы протокола (например, для заголовка и данных), используемый для протоколов любого уровня.

Protocol Interpreter (PI) Интерпретатор протокола. Средство выполнения функций FTP. В FTP определены две роли для PI: пользователь и сервер.

Protocol State Состояние протокола. Определенный этап в развитии протокола: состояние информационное, экспериментальное или историческое.

Protocol Status Статус протокола. Уровень требований.

Proxy ARP Прокси (посредник, агент) ARP. Использование маршрутизатора для ответа на запросы ARP. Это допустимо, когда запрашивающий хост предполагает локальное размещение точки назначения, хотя фактически точка назначения расположена вне зоны действия маршрутизатора.

Push Service Служба выталкивания. Служба TCP, позволяющая приложению указать, что некоторые данные должны быть доставлены с максимально возможной скоростью.

Receive Window Приемное окно. Допустимый диапазон порядковых номеров, которые отправитель может передавать по соединению за заданное время.

Record Structures Структура записи. Общая структура файлов данных. Во время пересылки файла его структура рассматривается как последовательность записей, разделенных маркерами End-of-Record.

Remote Network Monitor (RMON) Удаленный сетевой монитор. Устройство сбора информации о сетевом трафике.

Remote Procedure Call (RPC) Вызов удаленной процедуры. Протокол вызова приложением процедуры, которая будет выполняться сервером. Сервер возвращает выходные параметры и код завершения.

Request For Comments (RFC) Запрос комментария. Документ, описывающий протокол Интернета или связанные с ним сведения. Документы RFC интерактивно доступны на различных сетевых информационных центрах (Network Information Center).

Reseaux IP Europeens (RIPE) Европейский исследовательский центр IP. Координационный центр регистрации сетей для Европы.

Resolver Определитель. Программное обеспечение доступа клиента к базе данных Domain Name System.

Retransmission Timeout Тайм-аут повторной пересылки (ретрансляции). Если на сегмент не получен ACK в период тайм-аута повторной пересылки, то TCP повторно отправит сегмент.

Reverse Address Resolution Protocol (RARP) Протокол обратного разрешения адресов. Протокол для исследования компьютером своего IP-адреса посредством широковещательной рассылки в сети.

Round-Trip Time (RTT) Время цикла. Время между посылкой сегмента TCP и получением ACK для этого сегмента.

Router Маршрутизатор. Система, пересылающая дальше трафик уровня 3, не адресованный явно на эту систему. Применяется для соединения отдельных локальных и региональных сетей в интернет и пересылки трафика между этими сетями.

Routing Information Field (RIF) Поля информации о маршрутизации. Поле в кадре Token-Ring для идентификации пути к точке назначения, находящейся через один или несколько мостов.

Routing Information Protocol (RIP) Протокол информации о маршрутизации. Простой протокол для обмена информацией между маршрутизаторами. Исходная версия была частью набора протокола XNS.

Routing Policy Политика маршрутизации. Набор источников и точек назначения, на которые автономная система желает маршрутизировать трафик.

Routing Registry Реестр маршрутизации. База данных с информацией о маршрутах, используемая для пересылки данных через две или большее число автономных систем.

Security Association Ассоциация безопасности. Коммуникация, защищенная определенным набором параметров безопасности.

Security Gateway Шлюз безопасности. Система, которая обеспечивает защиту датаграмм, посланных между внутренними системами (не доверенных внешним системам).

Segment Сегмент. Элемент данных протокола (Protocol Data Unit), состоящий из заголовка TCP и, возможно, некоторых данных.

Send Window Выходное окно. Диапазон порядковых номеров между последним октетом данных, который уже был послан, и левым краем приемного окна.

Sequence Number Порядковый номер. 32-разрядное поле в заголовке TCP. Если сегмент содержит данные, то порядковый номер связан с первым октетом данных.

Serial Line Interface Protocol (SLIP) Протокол интерфейса последовательной линии. Очень простой протокол для пересылки датаграмм IP по последовательной линии связи.

Service Provider Провайдер, провайдер Интернета, поставщик сетевых услуг. Организация, которая обеспечивает соединение по TCP/IP для своих клиентов. Некоторые провайдеры поддерживают клиентов малой локальной географической области, в то время как другие имеют национальную или международную область действия.

Shortest Path First Первым — кратчайший путь. Алгоритм маршрутизации, при выборе маршрута использующий сведения о сетевой топологии.

Silly Window Syndrome Синдром "бестолкового окна". Неэффективный перенос данных, приводящий к сообщению приемником о малом освобождении окна и, соответственно, пересылке отправителем такого же малого сегмента. Эта проблема легко устраняется с помощью алгоритма, описанного в RFC 1122.

Simple Mail Transfer Protocol (SMTP) Простой протокол пересылки почты. Протокол TCP/IP для обмена почтой между системами.

Simple Network Management Protocol (SNMP) Простой протокол сетевого управления. Протокол, обеспечивающий станциям управления мониторинг сетевых устройств и получение от них сообщений о прерывани[illegible]

Smoothed Deviation Сглаженное отклонение (девиация). Количественная оценка отклонения для усредненного времени цикла, используемая в вычислениях тайм-аута повторной пересылки TCP.

Smoothed Round-Trip Time (SRTT) Усредненное время цикла. Оценка текущего времени цикла сегмента и его подтверждения (ACK). Используется при вычислении значения тайм-аута повторной пересылки TCP.

Socket Address Адрес socket. Полный адрес коммуникации TCP/IP, состоящий из 32-разрядного адреса сети и 16-разрядного номера порта.

Socket Descriptor Дескриптор (описатель) socket. Целое число, используемое приложением для идентификации соединения. Применяется в программном интерфейсе socket из BSD.

Source Quench Подавление источника. Сообщение ICMP, посылаемое перегруженной системой источникам трафика.

Source Route Маршрутизация от источника. Последовательность IP-адресов, идентифицирующая путь пересылки датаграммы. Может содержаться в заголовке датаграммы IP.

Standard Generalized Markup Language (SGML) Обобщенный стандартный язык разметки. Мощный язык разметки для описания элементов в перемещаемых документах.

Stub Network Тупиковая сеть. Сеть, которая не переносит трафик между другими сетями.

Subnet Address Адрес подсети. Определенное количество бит локальной части IP-адреса, идентифицирующее множество систем, объединенных общей связью.

Subnet Mask Маска подсети. 32-разрядное число, в котором единицами отмечены позиции IP-адреса, связанные с сетью и подсетью.

Switched Multimegabit Data Service (SMDS) Мультимегабитовая служба коммутации данных. Служба пересылки данных, разработанная Bellcore, с протоколом, основанным на IEEE 802.6 (Metropolitan Area Network).

SYN Сегмент, используемый в начале соединения TCP. Каждый партнер посылает SYN, содержащий начальную точку для отсчета порядковых номеров сегментов и, при необходимости, размер самого большого принимаемого сегмента.

Synchronous Data Link Protocol (SDLC) Протокол синхронной связи данных. Подобен HDLC, который является частью набора коммуникационных протоколов SNA компании IBM. SDLC используется для одноточечных и многоточечных коммуникаций.

Synchronous Optical Network (SONET) Синхронная оптическая сеть. Телефонный стандарт пересылки информации по волоконно-оптическим каналам.

Systems Network Architecture (SNA) Архитектура сетевых систем. Набор протоколов коммуникаций данных, созданный и используемый компанией IBM.

T1 Цифровая телефонная служба со скоростью пересылки в 1,544 Мбит/с. Использует кадры DS1.

T3 Цифровая телефонная служба со скоростью пересылки в 44,746 Мбит/с. Использует кадры DS3.

Telnet Прикладной протокол TCP/IP, обеспечивающий подключение терминала к одному хосту для регистрации на других хостах и для взаимодействия с их приложениями.

Time-To-Live (TTL) Время жизни, возраст. Предельный временной интервал на существование датаграммы в интернете. TTL обычно определяется как максимальное число попаданий на участки сети, которые сможет пересечь датаграмма до своего отбрасывания (удаления).

Tn3270 Этот вариант используется в *telnet* для поддержки эмуляции терминала IBM 3270.

Token-Ring Технология локальных сетей, основанная на топологии кольца. Станции пересылают по кольцу специальное сообщение, называемое маркером, или жетоном. Текущий маркер имеет право передавать данные в течение ограниченного периода времени.

Transmission Control Block (TCB) Блок управления пересылкой. Структура данных для хранения информации о текущих коммуникациях TCP или UDP.

Transmission Control Protocol (TCP) Протокол управления пересылкой. Обеспечивает достоверную, ориентируемую на создание соединения передачу данных между двумя приложениями.

Transport Class 4 (OSI TP4) Транспортный класс 4. Протокол транспортного уровня OSI, работающий подобно TCP.

Transport Layer Interface (TLI) Интерфейс транспортного уровня. Интерфейс программирования приложений, предложенный компанией AT&T и взаимодействующий с протоколами TCP/IP и OSI.

Transport Service Access Point (TSAP) Точка доступа к транспортной службе. Идентификатор, указывающий протокол верхнего уровня, куда следует доставить элемент данных протокола (Protocol Data Unit).

Trivial File Transfer Protocol (TFTP) Простейший протокол пересылки файлов. Один из основных протоколов TCP/IP, используемый для загрузки и выгрузки файлов. Типичным применением является инициализация бездисковых рабочих станций или загрузка программного обеспечения из контроллера в промышленные роботы.

Trojan Horse Троянский конь. Программа, выполняющая вроде бы полезную работу, но содержащая и тайную утилиту, которую злоумышленник может использовать для доступа к данным или для взлома компьютера.

Trunk Coupling Unit (TCU) Элемент магистрального (транкового) соединения. Аппаратное средство для соединения Token-Ring с магистральным кольцом.

Unicast Address Адрес одноадресной рассылки. Присваивается единственному интерфейсу.

Uniform Resource Locator (URL) Единый указатель ресурсов. Идентификатор элемента, который может быть извлечен браузером WWW. Указывает на определенное место в сети.

Uniform Resource Name (URN) Единое имя ресурса. Идентификатор элемента, который может быть извлечен браузером WWW. Указывает на определенное имя, отображаемое на несколько мест, откуда допустимо извлекать данный элемент.

Universal Resource Identifier (URI) Универсальный идентификатор ресурса. Идентификатор элемента, который может быть извлечен браузером WWW. Может быть Uniform Resource Locator или Uniform Resource Name.

Universal Time Coordinated Универсальное время. Ранее называлось временем по Гринвичу.

Urgent Service Служба срочной доставки. Служба TCP, позволяющая приложению отметить данные как срочные, которые должны быть обработаны принимающим приложением как можно скорее.

Usenet Тысячи групп новостей (похожих на электронные доски объявлений), информация которых доступна в Интернете.

User Agent (UA) Пользовательский агент. Приложение электронной почты, помогающее конечному пользователю создавать, хранить и отправлять исходящие сообщения, а также просматривать, сохранять и отвечать на поступающие сообщения.

User Datagram Protocol Протокол пользовательских датаграмм. Простой протокол, позволяющий приложению послать отдельное сообщение другим приложениям. Доставка не гарантируется, и сообщения могут быть получены в ином порядке, чем были отправлены.

Virtual Circuit Виртуальная цепь. Термин из сетей с коммутацией пакетов. Может совместно использоваться несколькими пользователями, хотя для каждого из них она будет выглядеть как соединение между конечными точками.

Virus Вирус. Программа, цепляющаяся к другим, законным программам. Обычно разрушает локальные данные или вредит выполнению программ.

Wide Area Network (WAN) Региональная сеть. Охватывает большую географическую область. Типичные технологии WAN — это "точка-точка", X.25 и Frame Relay.

Well-Known Port Общеизвестный порт. Порт TCP или UDP, использование которого определено Internet Assigned Numbers Authority.

World Wide Web Множество серверов Интернета, обеспечивающих клиентам доступ к различным типам информации, включая документы с графическими изображениями, звуковыми файлами и ссылками на другие документы.

Worm Червь. Программа, копирующая себя на других сетевых участках.

X11 Разработанная в Массачусетском технологическом институте оконная система.

X.121 Стандарт CCITT, определяющий назначение номеров системам, подключенным к сети X.25. Эти номера используются для идентификации удаленной системы, чтобы запрос данных мог быть послан по виртуальной цепи.

X.25 Стандарт CCITT для соединения компьютеров в сеть, которая обеспечивает надежную пересылку данных на основе виртуальных цепей.

X.400 Ряд протоколов, определенных CCITT для пересылки сообщений и межперсонального обмена. Эти протоколы были позже одобрены ISO.

Xerox Network System (XNS) Сетевая система компании Xerox. Набор сетевых протоколов, разработанных в Xerox Corporation.

X/Open Консорциум компьютерных производителей по разработке единого окружения для прикладных приложений.

X-Window System Набор протоколов, разработанных в Массачусетском технологическом институте (MIT). Они разрешают пользователям взаимодействовать с приложениями, находящимися на различных компьютерах. Входные и выходные данные для каждого приложения возникают в окне на дисплее пользователя. Размещение и размер окон определяются пользователем.

Второе издание популярного справочника полностью переработано и расширено с целью предоставить читателю наиболее полное описание средств разработки, конфигурирования, использования и обслуживания сетей TCP/IP и соответствующих служб.

Книга написана увлекательно и доступно. Она содержит дополнительные материалы о нескольких протоколах Интернета, используемых серверами и браузерами WWW, а также рассматривает все последние изменения в этой области. В книгу включены главы о новом стандарте безопасности IP и протоколе IP следующего поколения, известном как IPng или IPv6. Рисунки и таблицы наглядно показывают влияние средств безопасности IP и IPng на существующие сетевые среды.

Издание содержит следующие дополнительные разделы:

- Безопасность IP и IPv6
- Описание средств WWW, новостей Интернета и приложений для работы с gopher
- Подробное описание серверов имен доменов (DNS), маски подсети и бесклассовой маршрутизации в Интернете
- Таблицы и протоколы маршрутизации
- Руководство по реализации средств безопасности для каждого из протоколов и приложений
- Примеры диалогов с новыми графическими инструментами

Новое издание бестселлера по TCP/IP станет незаменимым помощником для разработчиков сетей и приложений, для сетевых администраторов и конечных пользователей.

Computing McGraw-Hill

McGraw-Hill

A Division of The McGraw-Hill Companies

Издательство "Лори"
www.lory-press.ru