ПОЛНЫЙ ОБЗОР ТЕХНОЛОГИЙ ИНТЕРНЕТА

Стандарты и протоколы ИНТЕРНЕТА

- отраслевые стандарты
- обзор протоколов
- подробные ссылки

Дилип Найк

РУССКАЯ РЕДАКЦИЯ

Microsoft Press

Стандарты и протоколы Интернета

Наконец-то! Все стандарты и протоколы Интернета в одной книге!

«Стандарты и протоколы Интернета» — не очередная книга для «чайников» и не справочник по ресурсам Интернета. В этом уникальном издании Вы найдете все о базовых технологиях Интернета. По объему информации книга сравнима с техническими словарями, однако здесь термины и протоколы упорядочены по функциональным признакам и сопровождены подробными описаниями.

Весь Интернет в одной книге:

- организации, создающие стандарты протоколов Интернета;
- основные транспортные механизмы и протоколы;
- принципы сетевого взаимодействия;
- криптографические стандарты и их применение;
- службы каталогов, а также механизмы обмена сообщениями и передачи файлов;
- инструментальные средства для управления сетями и повышения производительности;
- детальное описание спецификаций World Wide Web;
- текстовые, аудио- и видеоконференции;
- поисковые машины, электронная коммерция и сжатие данных;
- язык VRML.

«Стандарты и протоколы Интернета» незаменима для программистов, аналитиков, разработчиков, руководителей проектов, сотрудников службы компьютерной безопасности, Web-мастеров и всех тех, кто по роду своей деятельности связан с Интернетом.

Об авторе

Дилип Найк (Dilip C. Naik) *имеет сертификат MCSE, степень мастера в электроинженерии и более 14 лет стажа работы в различных компаниях на трех континентах, включая 8 лет в Microsoft.*

Приглашаем на Web-узел издательства: www.rusedit.ru

ISBN 5-7502-0102-3

9 785750 201020

РУССКАЯ РЕДАКЦИЯ

Памяти моего отца, К. Д. Найка

INTERNET
Standards and Protocols

Dilip C. Naik

Microsoft® Press

Стандарты и протоколы

ИНТЕРНЕТА

Дилип Найк

РУССКАЯ РЕДАКЦИЯ

УДК 004.738.5.057.4
ББК 32.973.202
Н20

Найк Дилип

Н20 Стандарты и протоколы Интернета/Пер. с англ. — М.: Издательский отдел «Русская Редакция» ТОО «Channel Trading Ltd.», 1999. — 384 с.: ил.

ISBN 5–7502–0102–3

Книга подробно освещает основные современные стандарты и протоколы, используемые в сети Интернет, причем термины упорядочены не по алфавиту, а по функциональным признакам. Снабжена иллюстрациями, облегчающими понимание материала, а также множеством ссылок на официальные источники стандартов. Книга состоит из 16 глав, трех приложений и указателя.

Издание адресовано широкой аудитории: как новичкам, осваивающим новые области, так и тем, кто профессионально разбирается в технологиях Интернета и желает упорядочить имеющиеся знания.

При работе над этой книгой использованы технологии и оборудование для построения информационных радиосетей, предоставленные **Компанией ArtCommunications Ltd.**, тел.: **943-8446, 943-8438, 212-0696, http://www.artcoms.ru/**.

Подготовлено к печати издательским отделом «Русская Редакция» ТОО «Channel Trading Ltd.» по лицензионному договору с Microsoft Corporation, Редмонд, Вашингтон, США.

ISBN 1–57231–692–6 (англ)
ISBN 5–7502–0102–3

Оглавление

ЧАСТЬ 4

World Wide Web 223

ЧАСТЬ 5

Мультимедиа 291

Приложения 319

От автора

Один человек не способен переработать весь материал, собранный в этой книге. Я глубоко признателен Джону Пирсу (John Pierce), редактору из Microsoft Press, который поддерживал меня во время работы над книгой. Огромное спасибо Дэвиду Кларку (David Clark) из Microsoft Press и, естественно, всем тем, кто усердно трудился над этой книгой. Особо хочу отметить Барба Раньяна (Barb Runyan), Шерил Пеннер (Cheryl Penner), Трэвиса Бивена (Travis Beaven), Линду Эбенштейн (Linda Ebenstein), Стюарта Гринмана (Stuart Greenman) и Памелу Хэфи (Pamela Hafey). Я признателен моим коллегам, которые делились идеями и соображениями, вселяли в меня уверенность своими горячими уверениями, что книга такого рода актуальна и найдет своего читателя, а также помогали проверять рукопись.

Об авторе

Дилип Найк (Dilip C. Naik) имеет степень мастера в электроинженерии, а также сертификат MCSE (Microsoft Certified Systems Engineer). Он уже более 15 лет занимается компьютерами и примерно 10 из них работает в компании Microsoft, где освоил самые разные профессии — от разработки программного обеспечения и до управления проектами. Дилип успел пожить на четырех континентах, причем на трех из них трудился не покладая рук. Его девиз: «Есть мозги — можешь научиться». Эта книга — плод кропотливого труда Дилипа в новом для него качестве писателя.

Введение

Технологии Интернета непрерывно развиваются, и «идти в ногу» с новинками — непростая задача. Книга «Стандарты и протоколы Интернета» как раз и поможет Вам в этом. Автору пришлось достаточно потрудиться, чтобы охватить весь материал, ведь некоторые из рассмотренных здесь вопросов весьма объемны, например о протоколе TCP можно писать отдельный труд. Я надеюсь, что книга будет интересна самым разным читателям:

- и новичкам, пытающимся разобраться во все более популярных во всех областях, в том числе и в бизнесе, технологиях;
- и тем, кто уже изучил одну из областей Интернета и желает кратко ознакомиться с другими технологиями;
- и профессионалам, которым необходимо просто обновить знания или отыскать источник информации для дальнейшего углубленного изучения;
- и специалистам, которым по роду их деятельности нужны общие знания об Интернете. Это и те, кто готовится к собеседованию для устройства на работу, и менеджеры, которым приходится общаться с профессионалами по информационным технологиям, и даже сами профессионалы, отвечающие на звонки назойливых пользователей.

Эта книга концептуально весьма напоминает словарь. Здесь описаны многие термины, относящиеся к Интернету, но упорядочены они не по алфавиту, а по функциональным признакам. Например, одна глава посвящена электронной коммерции, другая — службам каталогов и третья — мультимедиа. При подборе материала я придерживался гибкого подхода. Некоторые темы покажутся Вам устаревшими, но я включил их потому, что без них нет Интернета. Другие, наоборот, описывают новые технологии. Пользуясь правом автора, наиболее важным, с моей точки зрения, темам я посвятил несколько страниц, а не несколько абзацев, как это принято в словарях.

В справочниках информация, доступная из разных источников, просто резюмируется. В этом смысле данная книга — не исключение. При

наличии времени, ресурсов и, конечно же, желания любой читатель, используя ссылки, указанные здесь, сумеет отыскать более подробные материалы. Ценность же этого издания в том, что оно содержит информацию, хотя и краткую, но собранную изо всех доступных источников. Вы держите в руках единое компактное руководство, и надеюсь, воспользуетесь им в полной мере.

Структура книги

Часть 1 «Основы передачи данных» посвящена базовым стандартам для различных типов сетей — от современных локальных до глобальных и Интернета. Здесь описаны физический, канальный и сетевой уровень модели ISO/OSI. О самой модели рассказано в главе 1 «Модель взаимодействия открытых систем». В части 1 рассмотрены также вопросы обеспечения безопасности: защита данных и сохранность паролей уже стали базовой сетевой услугой и больше не обеспечиваются приложением.

В части 2 «Поиск информации» описаны механизмы поиска информации. Некоторые из них, например служба WAIS, считались весьма перспективными, но сейчас «отходят в тень» (если уже не списаны вовсе). Другие технологии, упомянутые здесь, например протокол LDAP, имеют огромный потенциал и, вероятно, получат широкое распространение и поддержку.

В части 3 «Обмен информацией» рассказано о различных стандартах и протоколах, разработанных для обмена информацией. Это электронная почта и протоколы передачи файлов.

Часть 4 «World Wide Web» посвящена технологиям, имеющим отношение к WWW. Сейчас Web одновременно служит инструментом и поиска и обмена информацией. В связи со значительными успехами в разработке средств для Web и всеобщим интересом к этим вопросам, Web посвящена отдельная глава.

Наконец, часть 5 «Мультимедиа» описывает использование в Интернете средств мультимедиа, например аудио, видео и VRML.

По сути, эта книга о технологиях, а не о программных продуктах. Там, где это уместно, для примера я упоминаю некоторые продукты, реализующие конкретные технологии. Если какой-либо продукт упомянут в этой книге, а другой — нет, не стоит делать выводов об их качестве или популярности. Это всего лишь авторский произвол.

Создатели стандартов

Развитием программных протоколов и интерфейсов, используемых в Интернете, занимаются многие организации. Некоторые из них я перечислил ниже, но этот список ни в коей мере нельзя считать исчерпывающим.

Сообщество Интернета

Сообщество Интернета (Internet Society, ISOC) — это группа профессионалов и экспертов, координирующая его жизнедеятельность и развитие. Основная деятельность ISOC — разработка стандартов и протоколов совместно с другими организациями.

IAB

IAB (Internet Architecture Board, ранее — Internet Activities Board) — техническая консультативная группа в составе ISOC. Она наблюдает за архитектурой и развитием протоколов Интернета, создает стандарты, управляет серией документов RFC (Request for Comments) и готовит различные периодические издания. Также IAB сотрудничает с другими организациями, занимающимися техническими вопросами и стандартами, связанными с Интернетом. В составе IAB есть две основные подчиненные группы — IETF (Internet Engineering Task Force) и IRTF (Internet Research Task Force).

IETF и IESG

IETF — Рабочая группа инженеров Интернета — разрабатывает протоколы. Она состоит из отдельных рабочих групп, которые обновляют существующие стандарты и создают новые. Члены IETF обмениваются материалами с IRTF и рекомендуют стандарты для IESG (Internet Engineering Steering Group) — Группы управления инженерами Интернета, действующей совместно с IAB.

В IETF разрабатываются девять направлений: приложения, межсетевые службы, управление сетями, функциональные требования, маршрутизация, безопасность, служебные приложения, транспорты и услуги пользователям. Для каждого создана одна или несколько рабочих групп, ответственных за создание стандарта или разрешение проблемы. Рабочую группу обычно утверждают на неформальном заседании BOF (Birds of Feather), проводимом на съезде IETF.

IRTF и IRSG

IRTF — Исследовательская группа Интернета — проблемное подразделение IAB. В нее включены группы, которые занимаются протоколами, приложениями, архитектурой и технологиями Интернета. Ею руководят председатель IRTF и IRSG (Internet Research Steering Group) — Группа управления исследованиями Интернета, в которую входят руководители различных отделов и другие ученые.

Обязанности IETF и IRTF на первый взгляд похожи, однако IRTF проводит более долгосрочные исследования (людей меньше, структура неформальна, но исследования глубже), тогда как IETF решает краткосрочные задачи создания стандартов (больше людей и бюрократических операций, необходимых для формализации стандартов).

InterNIC

InterNIC (Internet Network Information Center) — Центр сетевой информации Интернета — регистрирует имена доменов Интернета и управляет базой данных этих имен. Этот процесс сейчас пытаются модернизировать. Одно из возможных решений — создание дополнительных центров регистрации. Кроме того, еще не завершены несколько судебных процессов по вопросам допустимости сбора оплаты за регистрацию и назначения полномочной организации по сбору этой оплаты.

IANA

IANA (Internet Assigned Numbers Authority) контролирует распределение в Интернете числовых параметров протокола IP, гарантируя, что каждый домен получает уникальное значение. Помимо IP-адресов, IANA служит центральным реестром для других чисел и данных, относящихся к Интернету. IANA находится в Институте информационных наук (Information Sciences Institute) при Университете Южной Калифорнии (University of Southern California).

Консорциум W3C

Тим Бернерс-Ли (Tim Berners-Lee) предложил модель WWW, будучи сотрудником Европейской лаборатории физики частиц (CERN). Сейчас он работает в Массачусетском технологическом институте (Massachusetts Institute of Technology, MIT) и занимает пост директора консорциума W3C.

World Wide Web Consortium (W3C) — это международный, не зависящий от производителей, промышленный консорциум, созданный на коммерческой основе и сотрудничающий с производителями и создате-

лями стандартов с целью развития Web-протоколов Интернета, таких, как HTTP, HTML и URL.

В США консорциум W3C существует на базе Лаборатории компьютерных наук Массачусетского технологического института (MIT Laboratory for Computer Science, MIT LCS), в Европе — на базе Национального исследовательского института информатики и автоматизации (Institut National de Recherche en Informatique et en Automatique, INRIA) и в Азии — на базе Keio University Shonan Fujisawa.

Другие организации

Помимо ISOC и взаимосвязанных с ним организаций, на развитие Интернета и других отраслей компьютерной индустрии оказывают влияние многие организации. Это — Международная организация стандартизации (International Organization for Standardization, ISO), Американский национальный институт стандартов (American National Standards Institute, ANSI), Европейская ассоциация производителей компьютеров (European Computer Manufacturers Association, ECMA), Институт инженеров по электротехнике и электронике (Institute of Electrical and Electronics Engineers, IEEE), Open Group, Фонд открытого программного обеспечения (Open Software Foundation, OSF), X/Open и другие. Помимо прочего, в ISO входит целый ряд рабочих групп, которые ведут исследования по другим направлениям, например описывают форматы бумаги и типы фотопленок, хотя сама ISO не занимается электротехническими и инженерными аспектами исследуемых проблем.

Коммерческие компании

Кроме стандартизирующих организаций, занимающихся разработкой протоколов Интернета, значительное влияние оказывают корпорации, использующие Интернет в коммерческих целях. Так, Microsoft, Netscape и Sun Microsystems принимают значительное участие в развитии протоколов и интерфейсов Интернета. Иногда эти интерфейсы становятся стандартами де-факто, иногда — не получают распространения, а иногда — передаются стандартизирующим организациям.

Например, Microsoft передала свои технологии ActiveX в ведение Active Group (подразделение Open Group). Теперь их дальнейшей судьбой занимается комитет из двенадцати производителей. Компания Netscape сотрудничала с ECMA для создания стандартизированной версии JavaScript, названной EcmaScript. Sun Microsystems в настоящее время работает над принятием Java в качестве стандарта ISO.

Документы RFC и Internet-Drafts

Техническая документация, касающаяся Интернета, существует в двух основных видах: RFC и Internet-Drafts. Первые появились еще в 1969 году при создании сети ARPANET и почитаются сообществом разработчиков Интернета как основные. В документах серии RFC подробно описаны сетевые протоколы и интерфейсы, а также раскрыты другие темы, имеющие отношение к Интернету. Помимо этого, в RFC иногда встречаются заметки с конференций и иногда даже шутки (традиционно публикуются RFC к Рождеству и к 1 апреля).

Документы серии RFC пронумерованы. Например, RFC 2000 — это «Internet Official Protocol Standards» от IAB. Если в документ вносятся изменения, то ему присваивается новый номер. Например, RFC 2000 заменяет (выводит из употребления) следующие RFC: 1920, 1880, 1800, 1780, 1720, 1610, 1600, 1540, 1500, 1410, 1360, 1280, 1250, 1200, 1140, 1130, 1100 и 1083. Когда Вы держите в руках документ RFC, убедитесь, это самая последняя версия.

Документы RFC относятся к разным категориям, которые регулярно уточняются. Степени «зрелости» RFC таковы (в порядке убывания): стандарт, черновой стандарт, предложенный стандарт, экспериментальный, информационный или исторический. Также все RFC ранжируются по необходимости использования: необходимый, рекомендованный, факультативный, ограниченного использования и нерекомендованный.

Internet-Drafts — другая серия технических документов об Интернете, используемых в основном различными рабочими группами IETF. Эти черновые документы имеют силу только в течение шести месяцев, после чего их обновляют, заменяют, или выводят из обращения.

Документы Internet-Drafts не нумеруются, как RFC, им присвоены уникальные имена файлов, часто в виде *draft-<Автор>-<Рабочая_группа>-<Тема>-<Версия>.txt*.

Компания, которая имеет собственные протоколы или интерфейсы, помимо RFC и Internet-Drafts может публиковать данные на своем Web-узле или использовать другие каналы информирования разработчиков.

Ссылки

http://info.isoc.org/index.html

http://www.isi.edu/iab

http://www.ietf.org

http://www.irtf.org

http://www.internic.net

http://www.iso.ch

http://www.iana.org/iana

http://www.w3.org

http://www.ietf.org/1id-abstracts.html

http://ds.internic.net/rfc

http://ds.internic.net/rfc/rfcXXX.txt (где XXX — номер RFC)

http://www.isi.edu/rfc-editor

RFC 1800, «Internet Official Protocol Standards»

RFC 1796, «Not All RFCs Are Standards»

Часть 1

Основы передачи данных

ГЛАВА 1

Модель взаимодействия открытых систем

В этой главе описана семиуровневая модель взаимодействия открытых систем (Open Systems Interconnection, OSI), предложенная Международной организацией по стандартизации (International Organization for Standardization, ISO). Модель ISO/OSI предполагает, что все сетевые приложения можно подразделить на семь уровней, для каждого из которых созданы свои стандарты и общие модели. В результате задача сетевого взаимодействия делится на меньшие и более легкие задачи, обеспечивается совместимость между продуктами разных производителей и упрощается разработка приложений за счет создания отдельных уровней и использования уже существующих реализаций.

Семиуровневая модель показана на рис. 1-1.

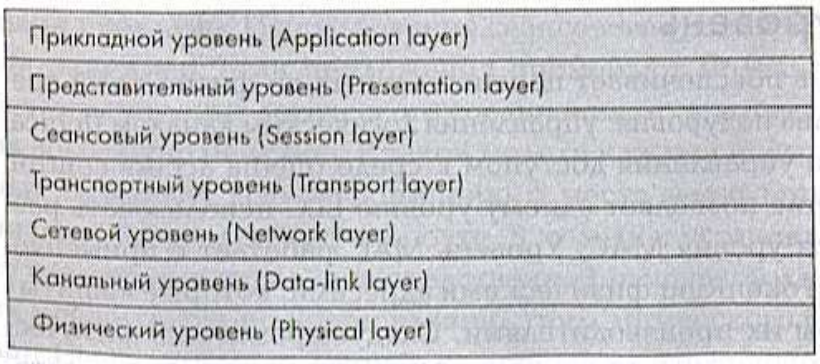


Рис. 1-1. *Семиуровневая модель ISO/OSI*

Теоретически, каждый уровень должен взаимодействовать с аналогичным уровнем удаленного компьютера. На практике каждый из них, за исключением физического, взаимодействует с выше- и нижележащими уровнями — предоставляет услуги вышележащему и пользуется услугами нижележащего. Учтите, в реальной ситуации на одном компьютере независимо друг от друга иногда выполняется несколько реализаций одного уровня. Например, компьютер может иметь несколько сетевых

адаптеров стандарта Ethernet или адаптеры стандартов Ethernet и Token-Ring и т. д.

В идеале архитектура ISO/OSI должна была появиться первой, а уже потом — все остальные разработки, созданные коммерческими, исследовательскими и стандартизирующими организациями, причем каждая — для конкретного уровня архитектуры. На самом же деле многие технологии появились раньше семиуровневой модели. Кроме того, некоторые новые технологии, разработанные позже модели ISO/OSI, не полностью с ней совместимы. Но, несмотря на столь сложный путь развития, эта модель обеспечивает достаточную совместимость, а значит, она выгодна.

Давайте поподробнее рассмотрим каждый из семи уровней и их применение.

Физический уровень

Физический уровень описывает физические свойства (например, электромеханические характеристики) среды и сигналов, переносящих информацию. Это физические характеристики кабелей и разъемов, уровни напряжения и электрического сопротивления и т. д., в том числе, например, спецификация кабеля «неэкранированная витая пара» (unshielded twisted pair, UTP). Подробное описание физического уровня выходит за рамки этой книги.

Канальный уровень

Канальный уровень обеспечивает перенос данных по физической среде. Он поделен на два подуровня: управления логическим каналом (logical link control, LLC) и управления доступом к среде (media access control, MAC). Такое деление позволяет одному уровню LLC использовать различные реализации уровня MAC. Уровень MAC работает с применяемыми в Ethernet и TokenRing физическими адресами, которые «вшиты» в сетевые адаптеры их производителями. Следует различать физические и логические (например, IP-) адреса. С последними работает сетевой уровень. В главе 2 «Основные транспорты и каналы» Вы найдете описание некоторых реализаций канального уровня, например Ethernet и TokenRing.

Сетевой уровень

В отличие от канального уровня, имеющего дело с физическими адресами, сетевой уровень работает с логическими адресами. Он обеспечивает подключение и маршрутизацию между двумя узлами сети.

Сетевой уровень предоставляет транспортному уровню услуги с установлением логического соединения (connection-oriented), например X.25, или без установления оного (connectionless), например IP (Internet Protocol). Одна из основных функций сетевого уровня — маршрутизация.

К протоколам сетевого уровня относятся IP и ICMP (Internet Control Message Protocol).

Транспортный уровень

Транспортный уровень предоставляет услуги, аналогичные услугам сетевого уровня. Надежность гарантируют лишь некоторые (далеко не все) реализации сетевых уровней, поэтому ее относят к числу функций, выполняемых транспортным уровнем. Транспортный уровень должен существовать хотя бы потому, что иногда все три нижних уровня (физический, канальный и сетевой) предоставляет оператор услуг связи. В этом случае, используя соответствующий протокол транспортного уровня, потребитель услуг может обеспечить требуемую надежность услуг.

TCP (Transmission Control Protocol) — широко распространенный протокол транспортного уровня.

Сеансовый уровень

Сеансовый уровень обеспечивает установление и разрыв сеансов и управление ими. Сеанс — это логическое соединение между двумя конечными пунктами. Использование сеансового уровня не всегда необходимо; например, если приложения применяют модель передачи данных без установления логического соединения, то протокол сеансового уровня абсолютно ненужен. В такой модели каждый посылаемый пакет данных содержит всю информацию о месте назначения — аналогично письму, отправляемому по почте. В модели с установлением логического соединения перед непосредственной отправкой данных выполняются мероприятия по установлению этого логического соединения (канала). После передачи данных — дополнительные действия по завершению сеанса. Наилучший пример этой модели — телефонный звонок. При наборе номера Вы устанавливаете логическое соединение, в результате на другом конце провода звонит телефон. Когда один из собеседников говорит «аллё», начинается передача данных. После того как один из абонентов вешает трубку, телефонная компания выполняет некоторые действия для разрыва соединения.

Сеансовый уровень следит также за очередностью передачи данных. Эту функцию называют «управление диалогом» (dialog management).

Вот примеры протоколов сеансового, представительного и прикладного уровней — SMTP (Simple Mail Transfer Protocol), FTP (File Transfer Protocol) и Telnet.

Представительный уровень

Представительный уровень позволяет двум стекам протоколов «договориться» о синтаксисе (представлении) передаваемых друг другу данных. Поскольку гарантий одинакового представления информации нет, то этот уровень при необходимости переводит данные из одного вида в другой.

Прикладной уровень

Прикладной уровень — высший в модели ISO/OSI. На этом уровне выполняются конкретные приложения, которые пользуются услугами представительного уровня (и косвенно — всех остальных). Это может быть обмен электронной почтой, пересылка файлов или любое другое сетевое приложение.

На рис. 1-2 показана модель ISO/OSI и некоторые протоколы соответствующих уровней.

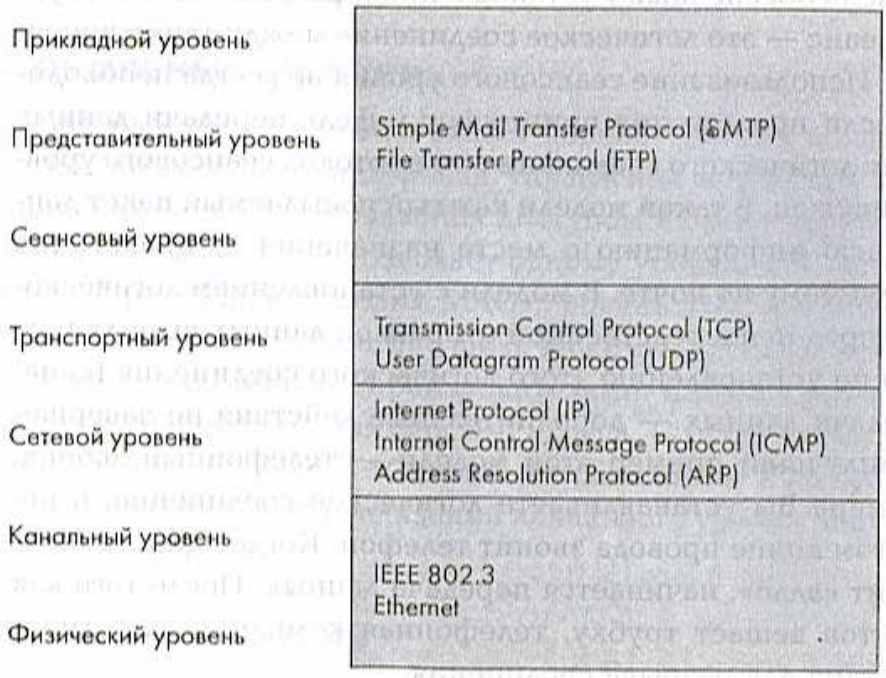


Рис. 1-2. *Семиуровневая модель ISO/OSI и некоторые протоколы*

Терминология

Сильно упрощая, можно сказать, что Интернет в основном состоит из устройств семи видов. Большинство из них работает на физическом, канальном и сетевом уровнях модели ISO/OSI. В нашей книге используются названия этих устройств, поэтому я уделю немного времени их описанию.

Повторители

Повторители (repeaters) работают на физическом уровне модели ISO/OSI и обычно применяются в локальных сетях (local area network, LAN) для увеличения длины сегментов. Они просто воспроизводят принятый сигнал, усиливая его мощность. Также повторители используются для увеличения длины сегмента в сетях топологии «шина». На рис. 1-3 изображена типовая схема: благодаря повторителю длина сетевой шины увеличивается за счет сегмента 2.

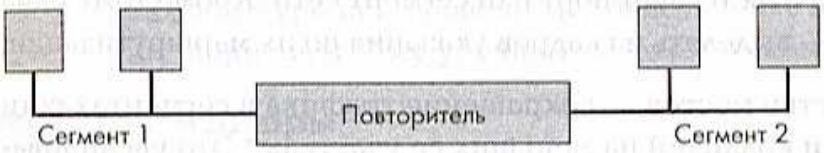


Рис. 1-3. *Повторитель и сегменты*

Повторители прозрачны; другими словами, остальные устройства (компьютеры, маршрутизаторы и т. д.) не способны обнаружить их присутствие. Повторитель может переводить данные из одной физической среды в другую, например из Ethernet в FDDI (Fiber Distributed Data Interface), причем оба сегмента, соединенные повторителем, должны применять одинаковые реализации уровня LLC — например, Ethernet или TokenRing, но не их сочетание. Это происходит потому, что повторители не умеют пользоваться услугами канального уровня.

Мосты

Мосты (bridges) работают на канальном уровне, который, как говорилось выше, включает подуровень MAC. Каждый узел сети имеет плату сетевого интерфейса (например, Ethernet или TokenRing), имеющую уникальный MAC-адрес. Мосты выделяют MAC-адреса из принимаемых кадров данных и избирательно пересылают эти кадры в соответствующие порты (рис. 1-4).

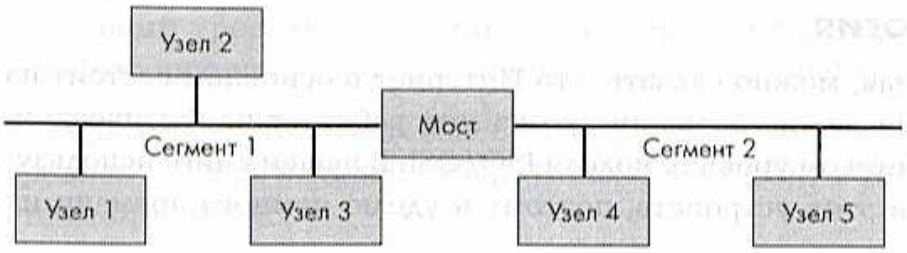


Рис. 1-4. *Пример работы мостов*

Мост игнорирует кадры, передаваемые между узлами, расположенными по одну сторону от него. Например, кадры, отправляемые от узла 1 к узлам 2 и 3, мост пересылать не будет. Эта операция распространяется только на кадры, отправляемые узлу, находящемуся по другую сторону моста, например от узла 1 к узлу 4.

Мост может иметь несколько портов для соединения более двух сегментов LAN. Пересылая пакеты, он составляет таблицу, в которой каждому MAC-адресу соответствует порт или сегмент сети. Кроме того, одна из функций моста — выделять из кадров указания по их маршрутизации.

Важное преимущество мостов — сокращение трафика в сегментах сети за счет локализации коллизий на меньших ее участках.* Это увеличивает время передачи полезных данных и снижает количество коллизий в одном сегменте.

Мост способен работать как накопительно-передающее устройство — принимать кадр целиком перед отправкой его в нужный порт. Попутно выясняется целостность кадров при помощи содержащейся в них *циклической контрольной суммы* (cyclical redundancy check, CRC). Поврежденные кадры не пересылаются. Это снижает нагрузку на сеть, поскольку плохой кадр проходит только один сегмент сети и не попадает в другие.

Мосты бывают двух типов: *локальный* (local bridge) соединяет два локальных сегмента LAN; *удаленный* (remote bridge) — два сегмента LAN через глобальную сеть (wide area network, WAN). Мост работает аналогично маршрутизатору, разница только в том, что мост устанавливает соединение на канальном уровне, а маршрутизатор — на сетевом.

Мосты могут переводить данные между разными уровнями MAC. Например, из одного порта принять кадр Ethernet, а в другой порт отослать кадр TokenRing, что показано на рис. 1-5.

---

* Коллизия — одновременная передача данных двумя сетевыми адаптерами в одну сетевую шину, что приводит к взаимным помехам и потере данных. — *Прим. перев.*

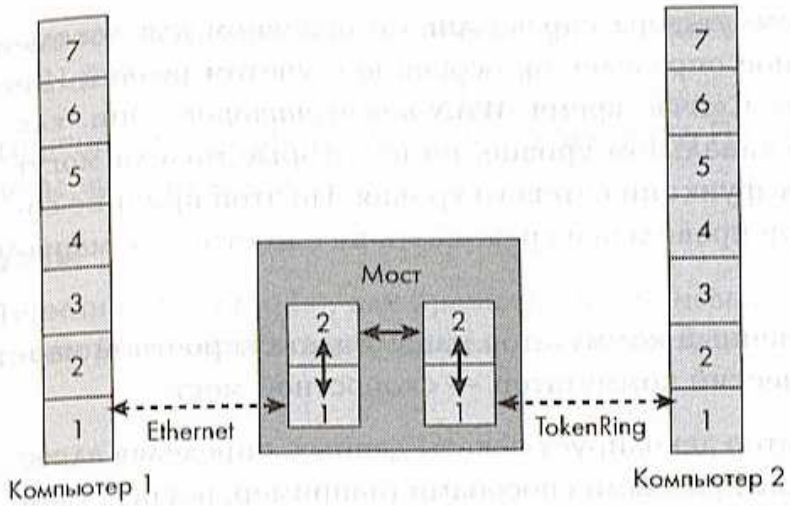


Рис. 1-5. *Мост, переводящий данные из одной среды в другую*

На рис. 1-5 изображено, как компьютер 1 посылает кадр мосту. Путь кадра проходит через физический уровень (уровень 1), подуровни MAC и LLC канального уровня (уровень 2). Далее данные передаются другим канальному и физическому уровням, а затем — компьютеру 2.

Маршрутизаторы

Маршрутизаторы (routers) работают на сетевом уровне. В отличие от мостов, пересылающих пакеты на основе таблиц физических адресов (например, адресов Ethernet), маршрутизаторы пересылают пакеты, используя таблицы логических адресов (например, IP-адресов). Маршрутизатор — ужасно сложная штука, и крупные компании, продавая их, получают приличные прибыли. В некоторых изданиях можно встретить термин *многопротокольный маршрутизатор* (multiprotocol router), который означает, что маршрутизатор понимает несколько протоколов сетевого уровня, например TCP и IPX (Internetwork Packet Exchange).

Коммутаторы

В толковании термина *коммутатор* (switch) есть некоторые разногласия. Согласно классическому определению, на канальном уровне коммутатор работает почти как мост. Суть отличия моста от коммутатора в том, что первый действует как накопительно-передающее устройство, а второй нет. Коммутатор сразу после декодирования адреса назначения отправляет кадр в соответствующий порт. Передача начинается сразу, даже если окончание кадра еще принимается. Преимущество такой схемы — высокая скорость. Недостаток в том, что коммутатор пересылает все кадры, даже поврежденные. Говоря формально, этот тип коммутатора называется *LAN-коммутатором*, но в литературе его часто именуют просто коммутатором.

Такое определение коммутатора справедливо в основном для локальных сетей. Современное определение, особенно с учетом реалий Интернета, в корне иное. Сейчас время *WAN-коммутаторов*. Они, как правило, работают на канальном уровне, но некоторые модели могут частично использовать функции сетевого уровня. По этой причине современный коммутатор правильнее сравнивать не с мостом, а с маршрутизатором.

Подведем итог: современный коммутатор напоминает скоростной маршрутизатор, а классический коммутатор — скоростной мост.

Современный коммутатор декодирует пакеты данных, определяя адрес сетевого уровня, который разными способами (например, посредством протокола ARP [Address Resolution Protocol]) сопоставляется с портом коммутатора. Последующие пакеты данных от того же отправителя к тому же получателю коммутируются уже на канальном уровне, в то время как маршрутизаторы делают это на сетевом уровне. Кроме того, коммутаторы не принимают участия в протоколах маршрутизации, например RIP (Routing Information Protocol). Еще одно преимущество современных коммутаторов в том, что они пригодны и для других приложений, например для виртуальных сетей (Virtual LAN).

Шлюзы

Шлюз (gateway) — это, как правило, аппаратное и программное обеспечение, соединяющее две разные сети, в которых используются разные протоколы. Обычно шлюзы работают на сетевом и более высоких уровнях. Так называемые *прикладные шлюзы* (application gateway) при пересылке данных из одной сети в другую выполняют трансляцию протоколов. Пример этого — почтовый шлюз, конвертирующий два разных протокола электронной почты. Иногда термин «шлюз» применяют, описывая ситуацию, когда не требуется трансляция протоколов, а данные просто пересылаются из одной сети в другую. В этом случае шлюз — это аппаратное или программное обеспечение, непосредственно связывающее две сети. Шлюз характеризуется наличием нескольких адресов сетевого уровня, например нескольких IP-адресов.

Хосты

Хост (host) — это компьютер, на котором работает сетевой протокол, например TCP/IP. Обычно хост имеет некоторое прикладное программное обеспечение, передающее и принимающее пакеты. Хост обменивается данными с другими хост-компьютерами, и значительная доля дея-

тельности в Интернете обусловлена управлением информационными потоками между хост-компьютерами.

Типичные примеры хостов: маршрутизаторы, ПК, серверы, прокси-серверы, шлюзы и т. д.

Узлы

Термином *узел* (node), как правило, кратко называют «мост», «маршрутизатор», «коммутатор», «шлюз» или «хост».

ГЛАВА 2

Основные транспорты и каналы

В этой главе описаны основные из используемых в Интернете транспортные протоколы и системы каналообразования, а также модемы и сетевые стандарты.

Модемы

Модем (сокращение от «модулятор/демодулятор») преобразует цифровые сигналы в аналоговые и наоборот и позволяет компьютерам (использующим цифровые сигналы) передавать информацию по обычным телефонным линиям (в которых применяются аналоговые сигналы). Модем-отправитель переводит цифровые сигналы компьютера в аналоговый вид, а затем отправляет их в телефонную линию. Модем-приемник выполняет обратное преобразование сигнала (из аналогового вида в цифровой) и передает его компьютеру.

Международный консультативный комитет по телефонии и телеграфии (Comité Consultatif International Télégraphique et Téléphonique, CCITT) установил некоторые стандарты передачи данных, в том числе протоколы модемной, сетевой и факсимильной связи. В 1993 году CCITT переименован в Международный союз электросвязи (International Telecommunication Union, ITU). Ниже в таблице описаны некоторые (но не все!) модемные протоколы ITU. Здесь не упомянуты, например, реализации собственных протоколов некоторых производителей. Сокращение «bps» означает *bits per second* (бит/с). Один символ состоит из 8 бит, плюс стартовый и стоповый биты, значит, для передачи одного символа надо 10 бит.

Протокол	Скорость передачи данных, бит/с
V.34	28 800, 26 400, 24 000, 21 600, 19 200, 16 800 или 14 400
V.32bis	14 400, 12 000, 9 600 или 7 200
V.32	9 600, 4 800 или 2 400

Модем, инициирующий сеанс связи, пытается использовать наиболее быстрый из доступных протоколов, а если модем-приемник не поддерживает его, то обращается к следующему протоколу. В случае с перечисленными ранее протоколами модуляции порядок их применения таков: сначала вызывающий модем пытается использовать протокол V.34, если это не удается, то применяется V.32bis, а уже потом — V.32. После подбора конкретного протокола, модем пытается передавать данные на максимально возможной для этого протокола скорости, а при возникновении большого числа ошибок или помех снижает ее.

Также модемы используют протоколы обнаружения и коррекции ошибок, которые позволяют выявить искаженные данные, а в некоторых случаях даже вычислить исходные данные без повторной их передачи. После выбора протокола модуляции аналогичным способом подключаются алгоритмы обнаружения и коррекции ошибок. Вот некоторые из них: MNP (Microcom Networking Protocol) уровней 1, 2, 3 и 4; V.42, также известный как LAPM (Link Access Protocol for Modems). Порядок применения этих протоколов следующий: V.42, MNP 4, MNP 3, MNP 2 и MNP 1. Учтите, что использование модемом протоколов обнаружения и коррекции ошибок не позволяет на более высоких уровнях совсем отказаться от применения протоколов контроля ошибок. Так, например, в протоколе передачи файлов контроль ошибок необходим, поскольку данные могут быть потеряны между модемом и компьютером, например, из-за переполнения буфера, и такую ошибку невозможно отследить средствами самого модема.

Кроме того, модемы пытаются сжимать передаваемые данные. В таблице перечислены основные стандарты передачи данных.

Стандарт передачи	Описание
MNP 3	Удаляет старт-стоповые биты, делая передачу синхронной; данные организованы в блоки
MNP 4	Данные организованы в блоки (как в MNP 3), длина которых зависит от качества связи. Более эффективен, чем MNP 3
MNP 5	Аналогичен MNP 4, но добавлена компрессия данных. Эффективность зависит от данных
V.42bis	Эффективность зависит от данных

Последовательность применения: V.42bis, MNP 5, MNP 4, MNP 3, а далее без компрессии.

Модемы на 56 кбит/с

Самая последняя модель модемов — те, что передают данные со скоростью 56 кбит/с по коммутируемой телефонной сети общего пользования (public switched telephone network, PSTN). Сеть PSTN изначально представляла собой полностью аналоговую систему, передающую аналоговый голосовой сигнал от одного конца телефонного соединения к другому. Постепенно на некоторых участках стали использовать цифровые линии, а в точках стыка аналоговые сигналы путем их дискретизации переводились в цифровые. В первую очередь это коснулось магистральных линий между телефонными станциями. Сейчас цифровыми стали почти все отрезки PSTN, за исключением так называемой «последней мили» от локальной телефонной станции до потребителя.

Классический модем генерирует аналоговый сигнал, который PSTN превращает в цифровой посредством дискретизации (на частоте примерно 8 000 Гц). Далее цифровой сигнал передается на сервер или поставщику услуг Интернета (Internet service provider, ISP), при этом может произойти цифро-аналого-цифровое преобразование. Модемы же на 56 кбит/с предполагают полностью цифровое соединение местной телефонной станции с сервером или ISP. Это проиллюстрировано на рис. 2-1.

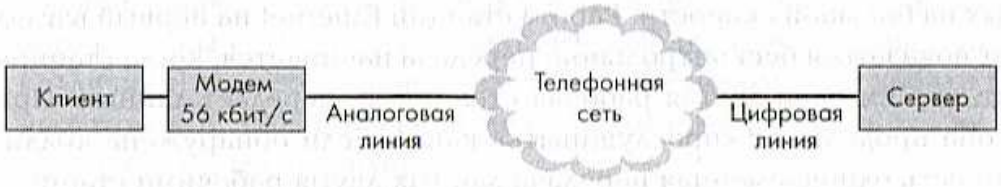


Рис. 2-1. *Подключение модема на 56 кбит/с*

Скорость 56 кбит/с возможна благодаря отсутствию дополнительного аналого-цифрового преобразования при котором иногда возникают погрешности дискретизации. Теоретически, максимальная скорость модема составляет 64 кбит/с, но на практике она ниже из-за того, что емкость телефонной линии ограничена федеральными инструкциями, и более того, передача некоторых цифровых сигналов затруднена, поскольку соответствующие им аналоговые частоты в отдельных линиях затухают или вообще фильтруются. Кроме того, модемы на 56 кбит/с асимметричны, то есть обеспечивают скорость передачи до 56 кбит/с в одном направлении и до 28 кбит/с — в другом. Это вызвано тем, что в одном направлении происходит одно аналого-цифровое преобразование, а в другом уже два.

Существует две основных реализации этой технологии: *K56flex* от компании Rockwell Semiconductor Systems и *x2* от U.S.Robotics (ныне часть 3Com). Модемы от U.S.Robotics построены на цифровых сигнальных процессорах. Элементарные операции они выполняют аппаратно, а все остальные — программно, а значит, модернизировать их несколько проще, чем модемы на микросхемах Rockwell Semiconductor Systems. Эти технологии не совместимы между собой, поэтому два взаимодействующих узла должны использовать одну из них. Сегодня разработан новый стандарт V.90, объединяющий обе эти технологии. Но о нем говорить еще рано, хотя пробные испытания V.90 показали, что не все такие модемы эквивалентны, а производительность канала (в терминах скорости передачи) сильно зависит от используемой аппаратуры и программного обеспечения. И хотя работа по согласованию двух упомянутых технологий начата, очевидно, что завершится она позже издания этой книги.

Ethernet

В этом разделе термин Ethernet используется в самом общем смысле, отражая все его значения. Разработанный компанией Xerox, стандарт Ethernet широко используется в локальных сетях, что весьма удобно при работе с приложениями, требующими периодической отправки данных на большой скорости. Работа станций Ethernet на первый взгляд может показаться бесконтрольной: передача начинается, когда станция решит, что ни одна другая рабочая станция не передает данные. При этом она продолжает «прослушивать» канал. Если обнаружена коллизия, то есть одновременная передача данных двумя рабочими станциями, то обе они прерывают процесс и перед повторной попыткой выжидают произвольный интервал времени. Если и далее возникают коллизии, то применяются другие алгоритмы. В одном из них время ожидания возрастает экспоненциально. Такая схема регулирования трафика называется *Множественный доступ с контролем несущей и обнаружением коллизий* (Carrier Sense Multiple Access with Collision Detection, CSMA/CD).

Сетевые адаптеры стандарта Ethernet выпускают многие производители. В тексте самого стандарта каждому производителю выделен диапазон адресов. Благодаря этому все адаптеры Ethernet имеют уникальные аппаратные адреса. Длина кадров при передаче данных варьируется, что показано на рис. 2-2.

Рис. 2-2. *Кадр Ethernet*

В каждом кадре Ethernet указан адрес как отправителя, так и получателя. Тем не менее технология Ethernet широковещательная, то есть каждая станция видит все передаваемые по соединительному кабелю кадры. Предполагается, что «честная» рабочая станция игнорирует кадры, адресованные не ей, но у сетевых адаптеров существует так называемый *смешанный режим* работы (promiscuous mode), когда все принимаемые станцией кадры, в том числе и «чужие», передаются программному обеспечению верхнего уровня. Перевести станцию в этот режим могут как законные пользователи (например, для работы анализаторов протоколов), так и сетевые хакеры.

Обычно в сетях Ethernet используют топологии «шина» или «звезда». Для соединения нескольких шин (или сетей) применяются коммутаторы и мосты Ethernet. Они пересылают кадры, адресованные рабочим станциям в другой сети. Коммутаторы и мосты уменьшают зоны, в которых возникают коллизии, и позволяют более эффективно использовать ресурсы сети для осмысленного трафика (того, что не приводит к коллизиям). Для соединения узлов Ethernet используют кабели различных типов. В таблице перечислены некоторые из них, а также максимальные длины сегментов.

Тип кабеля	Максимальная длина сегмента, м
10BaseT	100
10Base2	185
10Base5	500

TokenRing

Технология *передачи маркера* (token ring) содержит два основных стандарта. Первый — сеть TokenRing от IBM, для компании IBM она остается основной технологией локальных сетей. Второй — стандарт IEEE 802.5, разработанный Институтом инженеров в области электроники и электротехники (Institute of Electrical and Electronics Engineers, IEEE). Он практически повторяет стандарт IBM TokenRing, однако, есть некоторые

отличия. Основное в том, что спецификация IBM TokenRing требует топологию «звезда» и проводку на витой паре, а IEEE 802.5 не оговаривает этого.

Локальная сеть стандарта IEEE 802.5 основана на топологии «кольцо», а для определения очередности передачи информации используется короткое сообщение — *маркер* (token). (Кадр в такой сети состоит из маркера и данных, например, пользовательских или управляющих.) Маркер циркулирует по кольцу, и только владеющая им рабочая станция может передавать данные. Рабочая станция, получившая маркер, но не желающая пересылать данные, просто отправляет его дальше. Сравните, сетевой стандарт Ethernet не имеет такого механизма, то есть рабочие станции Ethernet не предотвращают коллизии, а обнаруживают их во время передачи. В сети же с передачей маркера коллизий возникнуть не может. «Занятый» маркер (он имеет пометку «занято» и используется для переноса полезной информации) циркулирует до тех пор, пока не достигнет адресата. Затем он следует обратно к передающей рабочей станции, которая удаляет сообщение, снимает пометку и отправляет в кольцо уже «свободный» маркер.

Сеть IBM TokenRing использует точно такой же маркер, который в силу другой топологии проходит по иному пути. Вместо объединения в физическое кольцо, узлы подключаются в форме звезды к одному концентратору — MultiStation Access Unit (MSAU), еще известному как Multistation Access Unit (MAU) или Smart Multistation Access Unit (SMAU). В однонаправленном логическом (хоть и не физическом) кольце MSAU передает маркер от станции к станции по звездообразному маршруту. Каждый MSAU соединяет до 8 станций, а для удлинения логического кольца можно соединить до 33 MSAU.

В сети с передачей маркера заложены некоторые функции отказоустойчивости. Например, один компьютер, назначенный на роль монитора, способен «отлавливать» маркеры, бесконечно циркулирующие из-за неполадок с передающим устройством, или обеспечивать существование не более одного маркера в кольце. Кроме того, все компьютеры кольца участвуют в локализации неполадок сети благодаря процессу *испускания маяка* (beaconing), когда заподозривший неполадку посылает по сети специальный сигнал — *маяк* (beacon). Компьютер испускает маяк до тех пор, пока не примет его от предыдущего компьютера. Через некоторое время подавать сигнал будет только компьютер, непосредственно следующий за местом неисправности. Поскольку маяк однозначно определяет пославший его компьютер, то однозначно определя-

ется и предыдущий компьютер. Эта информация полезна для переконфигурирования сети и исключения из кольца поврежденного участка. Например, устройства MSAU могут использовать испускание маяка для выявления неисправных или выключенных станций и поврежденных участков сети (в том числе из-за кабельной системы), а потом исключать их из кольца.

В обоих стандартах маркер состоит из 3 байт:

- байт признака начала маркера;
- байт контроля доступа, определяющий приоритет и занятость маркера. Значение приоритета используется для присвоения некоторым станциям сети более высокого приоритета (а, следовательно, лучшей способности передавать данные). Каждой станции определен приоритет, и только имеющие приоритет равный или более высокий, чем приоритет маркера, смогут его использовать для передачи данных;
- байт признака конца маркера.

Локальные сети с передачей маркера применяются там, где требуются гарантированная минимальная скорость передачи данных и некоторые возможности выявления и устранения неисправностей. Обычно такие сети обеспечивают скорости передачи данных от 4 до 16 Мбит/с.

FDDI

Оптоволоконная технология FDDI (Fiber Distributed Data Interface) все чаще применяется в сетях — в основном для магистральных систем, соединяющих несколько локальных сетей. Благодаря скорости передачи данных 100 Мбит/с, FDDI также используется для высокоскоростной связи между большими и/или быстрыми компьютерами.

Стандарт FDDI, принятый ANSI (American National Standards Institute) и ISO, определяет физический уровень (модели ISO/OSI) с использованием оптоволоконного кабеля в локальной сети топологии «двойное кольцо». (Стандарты ANSI и ISO полностью совместимы между собой.) Двойные кольца обеспечивают отказоустойчивость. Каждая станция FDDI подключена к двум кольцам, так называемым первичному и вторичному. При одиночном сбое возможно изолировать отказавшую станцию. Несколько сбоев иногда приводят к разбиению кольца на изолированные фрагменты, не способные к взаимодействию друг с другом. Также стандарт FDDI определяет протокол доступа к среде, в котором используется механизм передачи маркера. Каждая станция FDDI имеет уникальный 6-байтный адрес.

Линии T1/T3

Линии T1/T3 традиционно применяют для соединения двух значительно удаленных локальных сетей. Данные по T1 проходят со скоростью 1,544 Мбит/с, а по T3 — до 44,763 Мбит/с. Линия T1 состоит из 24 каналов по 64 кбит/с, их можно использовать для разных целей. Например, по одним передавать голосовую информацию, а по другим какую-то еще, в том числе графику и видео. В настоящее время линии T1 стали использоваться для подключения серверов Интернета к поставщикам услуг, особенно небольшими организациями, содержащими свои собственные Web-узлы.

ISDN

Широко применяемая операторами общественной телефонной связи (telephone carrier), Цифровая сеть интегрированных услуг (Integrated Services Digital Network, ISDN) — это семейство созданных CCITT протоколов, ориентированных на создание полностью цифровой всемирной сети передачи данных. Линия от подписчика до местной коммутационной станции, магистральные линии между коммутационными станциями и местная линия к адресату — цифровые, поэтому ISDN не требует ни одного аналого-цифрового преобразования. Кроме того, сеть ISDN обеспечивает бóльшую полосу пропускания, чем обычная (аналоговая) телефонная сеть, и позволяет одновременно пересылать голосовые и другие данные (например, компьютерные, музыкальные или видео). Еще одно преимущество сети ISDN — высокая скорость установления соединения, она в 5 или 6 раз выше, чем для обычных телефонных линий.

В стандарт ISDN входят два различных протокола — BRI (Basic Rate Interface) и PRI (Primary Rate Interface). При установке необходимо выбрать один из них. Стандарт PRI можно рассматривать как эквивалент линий T1. Но более распространен в сетях ISDN стандарт BRI, который содержит три раздельных канала:

- D-канал на 16 кбит/с (управляющий), для передачи управляющих и сигнальных данных ISDN;
- B-канал на 64 кбит/с (опорный), для передачи голосовой или другой информации;
- второй B-канал для передачи голоса или других данных.

Телефонные компании Северной Америки пока не полностью поддерживают сигнализацию ISDN. В результате некоторые B-каналы ISDN

гарантируют лишь 56 кбит/с, а не все 64 кбит/с. Информация в линиях ISDN передается в несжатом виде, поскольку до сих пор для них нет единого стандарта компрессии данных. Пару B-каналов можно использовать по-разному, например для одновременной передачи голоса и данных или для передачи одних и тех же данных в два разных места. Для увеличения пропускной способности в одном направлении иногда объединяют два B-канала в один, что известно как *разуплотнение* канала (inverse multiplexing).

Принятый стандарт разуплотнения является расширением протокола PPP (Point-to-Point Protocol) и называется *многоканальным PPP* (multilink PPP). (Протокол PPP подробно описан в главе 3.) Однако изделия (в частности, ISDN-модемы), в которых реализован этот стандарт, пока еще не очень популярны. Многие производители ISDN-модемов предлагают собственные решения для разуплотнения канала, что вынуждает использовать одинаковые терминальные адаптеры (то есть ISDN-модемы) на обоих концах соединения.

Линию ISDN можно приобрести у местной телефонной компании, но они (линии) доступны не в каждом регионе. С другой стороны, ISDN способна работать и на обычных медных телефонных проводах. Телефонной компании остается только протестировать имеющийся кабель и, если он соответствует нормам, установить необходимое оборудование у заказчика и на телефонной станции. Учтите, что оборудование, установленное у заказчика, подпитывает сеть ISDN и выдает сигнальную информацию. Поэтому, при отключении электроэнергии и отсутствии дополнительной телефонной линии Вам не удастся даже просто позвонить.

Frame relay

Frame relay (ретрансляция кадров) — это протокол с коммутацией пакетов, применяемый в глобальных сетях. Он появился как часть ISDN, но был внедрен раньше формального принятия спецификаций ISDN. Технологию Frame relay можно считать облегченной версией X.25, без усиленных механизмов выявления и коррекции ошибок, которые прежде требовались из-за ненадежности аналоговых линий. Это логично, поскольку стандарт X.25 разрабатывался, когда большинство телефонных линий были аналоговыми, а Frame relay появился уже в эру полностью цифровых телефонных линий (за исключением «последней мили»). Обычно Frame relay используется для переноса данных, но, установив дополнительное оборудование, Вы сможете его применять для передачи голоса.

Технология Frame relay активно продвигается операторами телефонной связи и обычно используется в сочетании с линиями T1, T3 или выделенными линиями на 56 кбит/с. Каждому подписчику назначается порт, состоящий из специального оборудования, обеспечивающего интерфейс между подписчиком и транспортом общего пользования. В процессе подписки между различными портами Frame relay устанавливаются постоянные виртуальные каналы (permanent virtual circuit, PVC), или постоянные логические соединения. Кроме того, каждому межпортовому каналу назначается фиксированная минимальная скорость передачи данных, гарантированная оператором связи. Порт может использовать скорости выше гарантированной, но в этом случае оператор связи лишь попытается передавать данные на более высокой скорости.

Протокол Frame relay ориентирован на создание логического соединения. Каждый обслуживаемый вызов получает идентификатор канала данных (data link call identifier, DLCI), выделяемый динамически. Однако при появлении полупостоянных виртуальных каналов выделение DLCI происходит статически в момент их регистрации. Идентификатор DLCI включается во все последующие кадры данных. Учтите, что DLCI имеет смысл только для локального порта. То есть для одного PVC, существующего между двумя станциями, возможны два идентификатора DLCI, причем отличные на разных портах. Это показано на рис. 2-3.

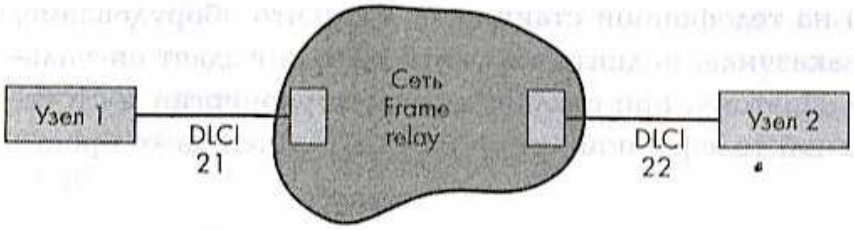


Рис. 2-3. *PVC для узла 1 обозначается DLCI 21, а для узла 2 — DLCI 22*

Стоимость услуг Frame relay зависит не от расстояний, а от гарантированной скорости передачи данных (committed information rate, CIR), обеспечиваемой оператором связи. Чем ниже CIR, тем меньше цена. Некоторые операторы предлагают порты даже с нулевым значением CIR, а другие оценивают услуги Frame relay в зависимости от числа задействованных PVC. По сравнению с технологией асинхронного режима передачи (Asynchronous Transfer Mode, ATM) стоимость Frame relay очень привлекательна.

Преимущество технологии Frame relay в том, что один порт может одновременно использовать несколько PVC для различных целей. Так, например, один PVC можно выделить для обслуживания исходящих

телефонных звонков, что исключит загрузку всей емкости канала одним конкретным видом трафика, например входящим.

SMDS

SMDS (Switched Multimegabit Data Service) — это высокоскоростная служба с коммутацией пакетов, способная переносить большие объемы данных со скоростями от 56 кбит/с до 34 Мбит/с. Она, так же как и Frame relay, широко внедряется операторами телефонии общего пользования. В отличие от ATM и Frame relay, SMDS не устанавливает логического соединения, то есть она не основана на виртуальных каналах. Один порт SMDS связывается с другим, вызывая его предопределенный адрес, а маршрут информации заранее неизвестен. Общественные операторы связи предлагают дополнительные услуги, например возможность фильтрования соединений SMDS. Стоимость услуг SMDS складывается из фиксированной месячной ренты и оплаты за использование. Первая зависит от выделенной пропускной способности, вторая — от объема передаваемых данных, но никак не от географической удаленности портов SMDS.

ATM

Технология сетей с асинхронным режимом передачи (Asynchronous Transfer Mode, ATM) возникла при дополненнии механизмами статистического уплотнения технологии синхронного режима передачи (Synchronous Transfer Mode, STM), широко применяемой в современной индустрии передачи данных. В технологии STM между двумя взаимодействующими узлами сети ходит «поезд» из частиц данных. Все частицы, так называемые «*люльки*» или «*вагоны*» (учтите, в литературе об ATM их часто называют *пакетами* или *ячейками*) принадлежат разным соединениям. За некий промежуток времени поезд может курсировать между узлами несколько раз. На пропускную способность канала между двумя узлами влияют частота, с которой «поезд» повторяет свой маршрут, а также количество и размер «вагонов», что исключает возможность перегрузки сети. Перед началом обмена данными происходит установление связи — соединению назначается конкретный «вагон», который освобождается только после разрыва этого соединения. Если через канал не передаются данные (например, оба телефонных собеседника замолчали), то «вагон» продолжает ходить «порожняком».

Очевидно, что в STM впустую тратятся ресурсы сети, поскольку в качестве идентификатора соединения выступает номер «вагона» (то есть его позиция в «поезде»). В ATM эта схема модифицирована за счет переда-

чи идентификатора соединения внутри самого «вагона» с данными. То есть когда в ATM возникает необходимость передать данные, то они вместе с идентификатором соединения упаковываются в следующий пустой «вагон». Идентификатор соединения называют идентификатором виртуального канала (Virtual Circuit Identifier, VCI). Как я уже упоминал, в ATM добавлено статистическое уплотнение (динамическое перераспределение свободной емкости канала). В STM количество соединений ограничено числом «вагонов», поскольку нельзя динамически увеличивать число обслуживаемых соединений за счет бездействия других. В ATM такого ограничения нет: считается, что не все установленные соединения одновременно передают данные, а если и так, то некоторые данные могут чуточку подождать в буфере до появления свободного «вагона».

Технология ATM гарантирует поступление данных в порядке их отправки, но не гарантирует саму доставку. Сохранение порядка обеспечивается тем, что данные передаются по логическому каналу, то есть следуют по одному маршруту. Технология ATM годится для самых разных приложений, в том числе допускающих утерю данных, но не задержку (например, аудио/видео в реальном времени) — здесь объем потерь должен быть ограничен и не очень велик, — или допускающих задержку данных, но не утерю (например, передача файлов) — здесь задержка должна быть ограничена и не очень большая, — или не допускающих ни утерю, ни задержку данных или, наконец, допускающих как утерю, так и задержку данных.

Технология ATM по своей сути ориентирована на логические соединения, но для совместимости со «старыми» приложениями, работающими без логических соединений, можно реализовать соответствующую службу поверх ATM. Такая служба сама выполняется как ATM-приложение (то есть она не является составной частью ATM), имитируя модель серверов без логических соединений. Эти серверы используют содержащуюся в ячейках информацию о маршруте их следования и направляют данные к месту назначения.

Формально говоря, технология ATM не вписывается в модель ISO/OSI. Она больше всего связана с канальным уровнем (уровень 2), но включает управление потоком данных, виртуальные каналы и другие функции вышележащих уровней. Кроме того, для поддержки протоколов более высоких уровней, например IP поверх ATM, определены уровни адаптации. И, наконец, ATM следует рассматривать не как продукт, а как технологию, имеющую множество применений, в том числе магистраль-

ные услуги для высокоскоростных сетей, эмуляцию локальной сети, «ATM для всех» и др.

Стандарты ATM

Бытует мнение (совершенно беспочвенное): «Чем больше стандартов, тем больше выбор». Международным консультативным комитетом по телеграфии и телефонии (Comité Consultatif International Télégraphique et Téléphonique, CCITT), чьи функции взял на себя Международный союз электросвязи (International Telecommunications Union, ITU) Организации объединенных наций (*http://www.itu.ch*), разработан стандарт широкополосной ISDN на основе ATM. Американский национальный институт стандартов (American National Standards Institute, ANSI, *http://www.ansi.org*) для разработки стандартов ATM создал группу T1S1.5. Несколько коммуникационных компаний, правительственных агентств и исследовательских групп, занимающихся ATM, организовали Форум ATM (ATM Forum), насчитывающий сейчас более 400 участников. Также в создание стандартов ATM вовлечена группа IETF (Internet Engineers Task Force). Все упомянутые организации разрабатывают стандарты, которые иногда дополняют друг друга. Примером тому служат LAN Emulation Standard и RFC 1577 «Classical IP and ARP over ATM», разработанные IETF и описывающие способ передачи IP через ATM. За более полной информацией обратитесь к разделу «Эмуляция LAN» этой главы.

Формат заголовка ATM

Ячейка ATM состоит из 5 байт заголовка и 48 байт данных. Существует два формата заголовка: UNI и NNI, применяемые соответственно для интерфейсов «пользователь-сеть» и «сеть-сеть». В заголовке UNI четыре бита выделены для поля **Управление потоком данных** (Generic Flow Control, GFC), что позволяет предотвратить перегрузки сети за счет ограничения объема входящих в нее данных. Поле **Индикатор виртуального канала** (Virtual Channel Indicator, VCI) идентифицирует ATM-сеанс и назначается либо навсегда, либо временно. Для группирования нескольких VCI используется поле **Идентификатор виртуального пути** (Virtual Path Identifier, VPI). Значения VPI и VCI имеют смысл только для конкретного узла, а при переходе данных из одного канала в другой они меняются. Поле **Индикатор типа полезной нагрузки** (Payload Type Indicator, PTI) отражает, занята ли ячейка данными пользователя или управляющей информацией. Это поле также позволяет выявлять заторы. Поле **Приоритет утери ячейки** (Cell Loss Priority, CLP) используется двояко. Во-первых, в нем можно хранить управляющую информацию о превышении пользователем оговоренной скорости передачи данных.

Во-вторых, в ячейках, несущих пользовательские данные, оно указывает, не допускает ли пользователь утерю этих данных. Так, например, при обращении к базе данных утеря информации недопустима, а в мультимедийных приложениях — разрешается, но в ограниченном объеме. Поле **Контроль ошибок заголовка** (Header Error Check, HEC) — это циклическая контрольная сумма (CRC) заголовка ячейки ATM. В нее не входит CRC остальных данных.

Уровни адаптации ATM

Технология ATM ориентированна на соединения, а данные в ней пересылаются ячейками по 53 байта. Но потребности приложений гораздо разнообразнее. Для их удовлетворения в технологии ATM определены *уровни адаптации* (adaptation layers), обеспечивающие все необходимые виды услуг.

Уровень 1

Уровень 1 (Adaptation Layer 1, AAL1) предоставляет услуги приложениям, для которых характерна постоянная скорость передачи данных. Это приложения с предсказуемым профилем генерируемого трафика. Данные отправляются периодически, с относительно постоянными интервалами.

Уровень 2

Уровень 2 (Adaptation Layer 2, AAL2) предоставляет услуги приложениям, генерирующим трафик непредсказуемо и с переменной скоростью передачи данных.

Уровень 3/4

Уровень 3/4 (Adaptation Layer 3/4, AAL3/4) предоставляет услуги приложениям, которые функционируют без установления логического соединения (то есть без создания канала, или сеанса), а также программам, передающим данные переменного объема. Кроме того, приложения могут надеяться на выявление ошибок передачи. Задержка данных не должна вызывать нестабильность работы. Приложение этого уровня имеет возможность послать сразу несколько блоков данных уровню ATM и не дожидаться завершения передачи предыдущих.

Уровень 5

Уровень 5 (Adaptation Layer 5, AAL5) похож на AAL3/4, однако передает только одну порцию данных в единицу времени.

ATM-адреса

Международным союзом электросвязи (ITU) определен формат адресов для ATM-сетей общего пользования (то есть обслуживаемых коммерческими операторами передачи данных). Этот формат основан на системе нумерации, принятой в телефонных сетях, а такой тип адресов называется E.164. Форум ATM определил три формата адресов для частных ATM-сетей, каждый длиной 20 байт. Первый основан на формате ITU E.164, второй — на формате IEEE 802, а третий — на спецификации ISO.

На территории США регистрацией ATM-адресов занимается ANSI. Подробности — по адресу *http://www.ansi.org/regfact.html*.

Эмуляция LAN

Стандарт «Эмуляция LAN» (LAN Emulation, LANE) разработан для того, чтобы существующие приложения и протоколы для Ethernet и TokenRing без каких-либо изменений могли функционировать поверх ATM. В результате переход на ATM-сети возможен при минимальной модификации. Стандарт LANE позволяет создавать виртуальные локальные сети, то есть объединять рабочие станции и серверы по функциональным признакам (например, для обеспечения безопасности или совместной работы с приложениями), а не по их местонахождению. Архитектуру LANE составляют четыре основные компонента: BUS-сервер (Broadcast and Unknown Server), LANE-клиент, сервер конфигурирования LANE и LANE-сервер.

BUS-сервер

Компонент BUS-сервер (Broadcast and Unknown Server) отвечает за разрешение адресов. В традиционных локальных сетях Ethernet или TokenRing, для передачи данных и разрешения адресов активно используется широковещание, что невозможно в ATM-сетях, которые, как я уже неоднократно упоминал, ориентированы на установление соединений. BUS-сервер обрабатывает широковещательные кадры и направленно пересылает те, для которых не удалось разрешить MAC- или ATM- адрес. Он принимает ARP-запросы (Address Resolution Protocol) и рассылает их всем членам эмулируемой LAN. После этого принимает ARP-ответы и отсылает их станции, запросившей разрешение адреса.

LANE-клиент

Компонент LANE-клиент (LAN Emulation Client, LEC) выполняется на всех узлах ATM-сети, в том числе на мостах, маршрутизаторах, комму-

таторах, рабочих станциях и серверах. Он инициирует установление связи и передачу данных к другим LANE-клиентам. Также он запрашивает у LANE-сервера соответствия между MAC- и ATM-адресами.

Сервер конфигурирования LANE

Сервер конфигурирования LANE (LAN Emulation Configuration Server, LECS) предоставляет служебную информацию (например, ATM-адрес LANE-сервера) и отслеживает, в какой виртуальной сети работает каждый из LANE-клиентов.

LANE-сервер

LANE-сервер (LAN Emulation Server, LES) обеспечивает соответствие между MAC- и ATM-адресами. Обычно LANE-сервер и BUS-сервер работают на одном узле сети.

В качестве иллюстрации того, как взаимодействуют описанные компоненты, представьте себе ситуацию, когда один узел желает передать данные другому. Для этого выполняются следующие операции.

1. Передающий узел отправляет по IP ARP-пакет.
2. LANE/BUS-сервер широковещательно рассылает этот ARP-пакет.
3. Принимающий узел сообщает свой MAC-адрес.
4. Передающий узел отправляет другой ARP-пакет, запрашивая также ATM-адрес принимающего узла, заданного его MAC-адресом.
5. LANE-сервер сообщает ATM-адрес принимающего узла.
6. Теперь передающий узел может создать виртуальный ATM-канал и передать данные.

Отказоустойчивость ATM

Единственное слабое место ATM-сети — серверы LANE (LES) и BUS. Избежать опасных ситуаций позволяет протокол SSRP (Simple Server Redundancy Protocol), разработанный компанией Cisco Systems. Он копирует конфигурационную информацию, которую хранит LECS, и контролирует состояние LES и BUS. Когда LANE-клиент пытается установить соединение (или возобновить его после отказа LES/BUS), LECS направляет его на функционирующий в данный момент LANE-сервер. Компания Olicom разработала для протокола SSRP дополнение — DCR (Dynamic Connection Redundancy), которое позволяет клиенту автоматически обнаружить работающий сервер LES или BUS.

IP поверх ATM

В классических локальных сетях IP-пакеты передаются в поле данных кадров Ethernet. В RFC 1577 определены способы работы протокола IP при использовании ATM вместо технологии Ethernet или какой-то другой. Эта возможность реализована в составе уровня адаптации ATM, работающего над канальным уровнем модели ISO/OSI. Преимущество «IP поверх ATM» (IP over ATM, IPOATM) в его большей эффективности, поскольку не надо упаковывать IP-пакеты в кадры локальной сети. Недостаток IPOATM в отсутствии поддержки широковещательных пакетов уровня MAC.

Протоколы поверх ATM

Спецификация «Протоколы поверх ATM» (Multi-Protocol Over ATM, MPOA) фактически состоит из нескольких документов, цель которых — унифицировать стандарты и обеспечить межсетевое взаимодействие посредством ATM. Разрабатывалась MPOA под патронажем Форума ATM. В состав MPOA входят:

- спецификация IPOATM — разработана IETF, описана в RFC 1577 и 1483;
- спецификация LANE — разработана Форумом ATM;
- протокол NHRP (Next Hop Routing Protocol) — разработан IETF, описан в RFC 2235, 2233 и 2232. Определяет способы разрешения адресов и маршрутизации между разными подсетями;
- спецификация MARS (Multicast Address Resolution Server) — разработана IETF и описана в RFC 2149. Определяет способы *группового вещания* (multicast) и разрешения адресов в среде ATM;
- спецификация IPNNI (Integrated Private Network to Network Interface) — разработана Форумом ATM. Описывает протокол маршрутизации, аналогичный RIP (Routing Information Protocol) и OSPF (Open Shortest Path First).

ATM для всех

Строго говоря, стандарты LANE и IPOATM — это промежуточный шаг при переходе к ATM-сетям. Они оба «маскируют» тот факт, что основная сеть является ATM-сетью. Однако чтобы полностью использовать возможности технологии ATM, в том числе услуги QoS (Quality of Service), необходима модернизация самих приложений. Первым разработку новых API начал Форум ATM. Но вскоре стало ясно, что приложе-

ния, использующие эти API, смогут работать только в ATM-сетях. Поэтому со временем Форум ATM изменил тактику; теперь он описывает лишь то, как приложения устанавливают, используют и разрывают виртуальные соединения.

Также Форум ATM подробно описал информацию (и механизм ее получения), которой обмениваются различные сетевые уровни, и способы запроса приложениями специальных услуг. Синтаксис и семантика этих API будут уточнены в других стандартах, например серий Windows Sockets (Winsock) и X/Open. Выигрыш здесь в том, что приложения, написанные для Winsock API или X/Open API, смогут работать как в обычных, так и в ATM-сетях. А уже существующие приложения, не использующие специальные услуги, не придется модифицировать для работы в ATM-сетях. Кроме того, поскольку Microsoft Winsock API уже ориентирован на логические соединения, то добавить к нему поддержку специальных услуг не составит труда, причем сохранится возможность писать приложения для не-ATM-сетей.

Gigabit Ethernet

Примерно 100 компаний создали объединение Gigabit Ethernet Alliance для разработки стандартов технологии Ethernet нового поколения. До появления Gigabit Ethernet двумя основными направлениями технологии Ethernet были собственно Ethernet (10 Мбит/с) и Fast Ethernet (100 Мбит/с). Хотя технология Gigabit Ethernet еще только разрабатывается и не получила широкого распространения, некоторые поклонники рассматривают ее как реального конкурента ATM. Но предсказание исхода сражения между Gigabit Ethernet и ATM выходит за рамки этой книги. Достаточно лишь отметить, что Gigabit Ethernet обеспечит передачу данных со скоростью 1 Гбит/с (гигабит в секунду) на расстояние от 300 м до 10 км. Очевидно, что сначала технологию Gigabit Ethernet будут внедрять на магистральных участках локальных или глобальных сетей, и вряд ли в ближайшем будущем она дойдет непосредственно до каждого рабочего места. По всей вероятности, в крупных сетях построят трехуровневую модель, состоящую из Ethernet, Fast Ethernet и Gigabit Ethernet.

Под началом IEEE создан комитет 802.3z для формирования стандартов Gigabit Ethernet. Стандарт 802.3z требует реализации Gigabit Ethernet в физической среде одного из двух типов: неэкранированная витая пара (UTP) или оптоволоконный кабель. Сегодня, поскольку сама технология только развивается, применяется только оптоволоконный кабель. Реа-

лизация Gigabit Ethernet на кабеле UTP остается, по крайней мере сейчас, в проекте.

Текущие реализации Gigabit Ethernet используют полнодуплексный оптоволоконный канал, соединяющий две станции. Другими словами, любой сегмент Ethernet состоит лишь из пары станций, каждая из которых имеет собственный канал передачи. Таким образом, исключена возможность возникновения коллизий. Обнаружение коллизий в технологии Gigabit Ethernet потребовало бы непосредственной близости одной станции к другой, поскольку при таких высоких скоростях в физическую среду могут попасть несколько кадров данных до того, как принимающая станция получит первый из них. А так как обе станции способны одновременно начать передачу, то при их удаленности друг от друга вероятна потеря стазу нескольких кадров данных. Но стандартом Ethernet допускает искажение и повторную передачу не более одного кадра.

В стандарте 802.3z для увеличения допустимого расстояния между станциями описаны две технологии: увеличение нагрузки (carrier extension) и «выстреливание пакетов» (packet bursting). В первой короткие кадры искусственно удлиняются до 512 байт (напомним, что кадры Ethernet имеют длину от 64 байт). Это позволяет рабочим станциям обнаруживать коллизии, находясь на большем удалении друг от друга. Очевидно, что при таком подходе пропускная способность сети расходуется нерационально, поскольку передаются лишние данные (которые будут отсеяны при приеме). Во втором варианте проблема решается за счет передачи кадров «очередями», сразу по несколько штук (что отличается от классической сети Ethernet, где за один раз станция может отправить только один кадр). Длина первого кадра «очереди» увеличивается до 512 байт, а последующие передаются без изменений.

Кабельные модемы

Большинство пользователей ПК в США уже имеют доступ к широкополосной системе связи в виде существующих кабельных сетей. Такие сети способны переносить данные на скорости до 30 Мбит/с. Кабельные модемы позволяют подключаться к Интернету через кабельные сети, так же как классические модемы — через телефонную сеть общего пользования. Скорость обычных модемов при работе в кабельной сети составляет около 28,8 кбит/с (иногда до 56 кбит/с). Большая скорость достижима потому, что используемая среда передачи (кабельная сеть) может переносить сигналы в гораздо более высоком диапазоне частот (порядка 6—8 МГц). Таким образом, кабельный модем для передачи и

приема данных использует расширенный спектр частот. Обычный модем подключают к компьютеру через последовательный порт, который не способен обеспечить скорость передачи данных, необходимую кабельному модему. Поэтому кабельные модемы подключаются через трансивер адаптера Ethernet. Модем, соответственно, переводит данные из формата Ethernet в аналоговый вид и наоборот. Поскольку для такой трансляции данных существует несколько популярных стандартов, то взаимодействующие компьютеры должны быть оборудованы кабельными модемами одной марки.

Скорость до 30 Мбит/с — только теоретически возможна, но обычно она существенно ниже. Например, из-за того, что сервер на другом конце кабельного соединения не обеспечивает достаточную скорость подключения к Интернету. Как только сервер получит требуемые данные, он сравнительно быстро отошлет их клиенту, но вот получение этих данных сервером иногда затягивается. Некоторые компании, например @Home, решают эту проблему путем кэширования на сервере часто используемых данных.

Еще одна причина снижения скорости кабельных модемов заложена в самой архитектуре кабельных сетей, где информация проходит только в одном направлении — от источника к потребителям (обычно нескольким сразу). Обратная связь и возможность непосредственного («точка-точка») взаимодействия клиентов не предусмотрены. Тем не менее существует способ отправлять запросы при помощи обычного модема по телефонной линии, однако это приводит к несимметричности сети, то есть ее пропускная способность в разных направлениях существенно различается. Такую сеть стоит применять там, где потоки данных несимметричны (например, просмотр Web, когда короткие запросы вызывают лавину ответной информации), но не в приложениях, требующих большой пропускной способности в обоих направлениях (например, в видеоконференциях).

Используемые в кабельных сетях провода сами являются источником проблем: разветвители (для подключения нескольких телевизоров к одному кабелю) и дешевые провода существенно снижают скорости передачи данных. Учитывайте и то, что кабельный канал от Вашего дома до поставщика услуг обычно используется коллективно. Суммирование всех упомянутых факторов приводит к уменьшению реальной скорости передачи данных примерно до 1,5 Мбит/с, о чем, конечно, не стоит жалеть — достаточно вспомнить «черепашью» скорость 28,8 кбит/с или даже 56 кбит/с.

xDSL

Технология DSL (Digital Subscriber Line) позволяет увеличить скорость на существующих телефонных линиях, проведенных посредством медных кабелей. Строго говоря, сама линия xDSL не является цифровой — это обычный медный провод, на обоих концах которого установлены цифровые модемы. (Префикс «*x*» — переменный, то есть в разных вариантах DSL он различный.) Преимущество xDSL по сравнению с ISDN в том, что даже при отключении электроэнергии в доме или офисе линия xDSL будет работать, так как это обычная телефонная линия.

Медные провода способны переносить сигналы в гораздо более широком диапазоне частот (в мегагерцовой области), а ограничение полосы пропускания телефонных линий в 3,3 кГц* обусловлено лишь наличием фильтров на телефонных станциях, но никак не свойствами самого медного провода. Поскольку в медном проводе сигнал затухает достаточно быстро, то пропускная способность прямо зависит от длины линии. Так, например, провод 24-го калибра позволяет переносить данные при скорости 1,544 Мбит/с на расстояние 18 000 футов, а при скорости 51,840 Мбит/с — лишь на 1 000 футов.

Все описанные ниже технологии схожи в одном: по телефонному проводу передается высокочастотный сигнал, сгенерированный телефонной компанией, а технологии DSL лишь модулируют его.

DS1/T1/E1

Технология DS1/T1/E1, изначально разработанная в Bell Labs, позволяет объединить двадцать каналов по 64 кбит/с в один высокоскоростной поток. Длина кадра в полученном канале составляет 193 бита, сюда входят и управляющие данные для уплотнения/разуплотнения. В Америке общий поток данных переносится со скоростью 1,544 Мбит/с, а сама технология известна как DS1 или — и это более распространенное название — T1. В настоящее время построено достаточно много таких линий, причем примерно через каждые 6 000 футов на них необходимо устанавливать повторители. В Европе эту технологию знают как E1, но в ней общий поток данных переносится на скорости 2,048 Мбит/с.

Линии T1/E1 используются для доступа к локальным и глобальным сетям (от доступа к серверам до частных сетей). В настоящее время они также связывают сотовые ретрансляторы, маршрутизаторы и другие

* По российским стандартам полоса пропускания телефонного канала составляет 3,1 кГц (от 300 Гц до 3 400 Гц). — *Прим. перев.*

подобные устройства. Линии T1/E1 не приспособлены для использования в жилых районах (частными лицами) из-за сильных помех — взаимное наложение сигналов обычно не позволяет иметь больше одной линии T1 в одном 50-парном кабеле. Кроме того, для жилых районов лучше подходят асимметричные системы, позволяющие передавать больше данных в район и меньше в обратном направлении.

DSL/ISDN

Модемы DSL/ISDN способны пересылать данные со скоростью 160 кбит/с на расстояние до 18 000 футов. Как и в технологии ISDN, их пропускная способность разделена между двумя B-каналами, обеспечивающими скорость по 64 кбит/с, и одним D-каналом, обеспечивающим скорость 16 кбит/с. Такие модемы используются для предоставления услуг ISDN BRI.

HDSL

Высокоскоростная технология HDSL (High-Data-Rate Digital Subscriber Line) основана на современных разработках из области модуляции сигналов, что позволяет передавать данные быстрее и с меньшим числом повторителей. Эта технология применяется для соединения сотовых ретрансляционных станций и офисных АТС, для предоставления доступа в Интернет и создания частных сетей передачи данных. Две линии HDSL обеспечивают общую пропускную способность, эквивалентную каналу T1 (1,544 Мбит/с), при рабочих расстояниях до 12 000 футов. Используя три линии HDSL вместо двух, можно обеспечить пропускную способность канала E1. Технология HDSL симметрична, то есть обеспечивает одинаковую скорость передачи данных в обоих направлениях.

ADSL

Асимметричная технология ADSL (Asymmetric Digital Subscriber Line) появилась позднее HDSL и предназначена для тех случаев, когда асимметричная загруженность канала выгодна. Это, например, доступ в Интернет, удаленный доступ к локальной сети, интерактивное видео и тому подобное. Она обеспечивает скорость приема (то есть из сети к потребителю) в диапазоне от 1,5 до 9 Мбит/с. Скорость передачи варьируется от 16 до 640 кбит/с. Большое преимущество ADSL по сравнению с ISDN в том, что питание в медный кабель идет от телефонной сети, — другими словами, пользоваться обычной телефонной связью можно даже при отключении электроэнергии. Технология ADSL стандартизирована институтами ANSI и ETSI (European Telecommunications Standards Institute).

В некоторых реализациях ADSL для модуляции линейного сигнала применяется *амплитудно-фазовая модуляция с подавлением несущей* (carrierless amplitude/phase, CAP). Как следует из названия, при использовании для передачи данных CAP-модуляции сигнал несущей частоты перед попаданием в линию связи подавляется. Хотя CAP-модуляция достаточно проста в реализации и относительно дешева, но она позволяет передавать данные на скорости не выше T1 и весьма чувствительна к помехам. Эта модуляция не признана частью стандарта ADSL ни в ANSI, ни в ETSI.

В технологии ADSL есть другой способ модуляции — *дискретная многочастотная модуляция* (discrete multitone, DMT). При таком способе вся доступная полоса частот делится на 256 каналов, а поскольку высокочастотные сигналы более чувствительны к помехам, полученные каналы имеют разную пропускную способность. Модуляция DMT позволяет достичь более высоких скоростей, чем CAP, но в настоящее время ее реализация обходится дороже. Однако она принята ANSI и ETSI в качестве стандарта для ADSL.

RADSL

Адаптивная технология RADSL (Rate-Adaptive Digital Subscriber Line) — это обычная ADSL, способная перед началом передачи протестировать качество и длину канала и подобрать подходящую скорость. Стандарты RADSL еще разрабатываются, поэтому неясно, будет ли эта подстройка происходить однократно или периодически.

VDSL

Сверхскоростная технология VDSL (Very-High-Data-Rate DSL) является самой быстрой из семейства DSL. Она обеспечивает скорости от 13 до 52 Мбит/с при приеме и от 1,5 до 2,3 Мбит/с при передаче данных по обычным медным проводам на расстояниях от 1 000 до 4 000 футов. Однако это, вероятно, самая далекая от стандартизации технология семейства DSL.

SDSL

Одноканальная технология SDSL (Single-Line DSL) аналогична VDSL, но имеет два главных отличия:

- ограничена расстоянием 10 000 футов;
- использует лишь одну линию.

Стандарты SDSL все еще разрабатываются, и пройдет как минимум год (или больше), прежде чем они получат распространение.

В таблице собраны основные характеристики xDSL, о которых я рассказал ранее.

	Скорость приема (из сети к потребителю)	Скорость передачи (от потребителя в сеть)	Области применения
DSL	160 кбит/с	160 кбит/с	Услуги ISDN, передача данных и голоса
HDSL	1,544 Мбит/с по двум витым парам	1,544 Мбит/с по двум витым парам	T1, WAN, LAN, доступ к серверам
HDSL	2,048 Мбит/с по трем витым парам	2,048 Мбит/с по трем витым парам	E1, WAN, LAN, доступ к серверам
ADSL	1,5 – 9 Мбит/с	16 – 640 кбит/с	Доступ в Интернет, интерактивное видео, LAN, удаленный доступ
VDSL	13 – 52 Мбит/с	1,5 – 2,3 Мбит/с	Доступ в Интернет, интерактивное видео, HDTV, LAN, удаленный доступ

Спутники

Попытки расширить канал доступа в Интернет зашли достаточно далеко, буквально «до небес». Технология спутниковых сетей «шагнула» от лабораторных образцов до коммерческих систем.

Компания Hughes Systems запустила проект под названием DirectPC. (Кстати говоря, DirecTV — это система, передающая кабельные телевизионные каналы через спутниковые сети.) Система DirectPC требует небольшой спутниковой антенны, которая кабелем подключается к адаптеру, установленному в ISA-слот персонального компьютера. Обеспечиваемая скорость приема данных — до 11,7 Мбит/с, но на самом деле это пропускная способность одного транспондера́, совместно используемого несколькими потребителями. Пропускная способность, которую реально получают пользователи, определяется популярностью системы, то есть числом пользователей на один транспондер. Компания Hughes утверждает, что их системы обеспечивают каждому пользователю скорость 400 кбит/с. Для DirectPC, как и для некоторых кабельных систем, необходимо наличие обычного модема, через который посылаются запросы к системе DirectPC. Результат — асимметричная сеть, где скорость передачи в одном направлении существенно отличается от обратной; она подходит для асимметричных приложений, например путеше-

ствий по Web. А программы, для которых необходима высокая пропускная способность в обоих направлениях, например видеоконференции и базовые услуги телефонии, не смогут работать с этой системой. Стоимость услуг DirectPC зависит не от длительности соединения, а от объема передаваемых данных.

Еще одна система, в которой сочетаются спутниковые и радиотехнологии, это LMDS (Local Multipoint Distribution System). Она отличается только тем, что пользователь должен установить не спутниковую «тарелку», а маленькую антенну для связи с местным ретранслятором, который, в свою очередь, через спутник свяжется с центральным узлом. Кабель от системы LMDS подключается к ПК через адаптер Ethernet. В технологии LMDS вся доступная пропускная способность сети делится на несколько каналов. И хотя каждый из них способен передавать данные со скоростью до 50 Мбит/с, реальная скорость, зависит от многих факторов, в том числе от уровня помех и от числа пользователей одного канала.

Беспроводные сети

Ожидается, что с ростом популярности портативных компьютеров, устройств на базе Windows CE и устройств PDA (Personal Digital Assistant), популярность беспроводных локальных сетей (Wireless LAN) также заметно возрастет. Уже сейчас ставится вопрос не «если», а «когда». Не так давно IEEE завершил разработку спецификации IEEE 802.11, которая описывает беспроводные локальные сети и служит стандартом для их совместимости. В общих чертах беспроводная сеть состоит из устройств трех типов: клиент, сервер и точка доступа. Последняя, будучи приемно-передающим устройством, выполняет те же функции (но в сильно уменьшенном масштабе), что и ретранслятор сотовой связи, работая промежуточным звеном между клиентом и сервером. Клиент — это любое мобильное устройство, инициирующее запросы на передачу данных; а сервер — устройство, к которому клиент направляет свои запросы. Технология беспроводных сетей обеспечивает передачу данных на скорости от 1 до 2 Мбит/с при удаленности устройств на расстояния от 200 до 1 000 футов.

Реализация беспроводной сети усложняется тем, что клиент может послать запрос через одну точку доступа, а затем переместиться в зону, обслуживаемую другой точкой доступа. В этом случае первая точка доступа должна передать ответственность за клиента второй точке. Еще одна трудность известна как «проблема скрытой точки». Она возникает, когда одно беспроводное устройство слишком удалено от другого и

не может напрямую связаться с ним, но оба они достаточно близки к одной и той же точке доступа (а последняя способна обслужить либо одного клиента, либо другого, но не двух одновременно). Ситуация осложняется и тем, что беспроводному устройству необходима возможность отключаться (для продления срока службы батарей) и периодически «просыпаться», чтобы точка доступа о нем не забыла. Это еще одна причина того, что точки доступа должны работать как приемно-передающие устройства.

Спецификация IEEE 802.11 затрагивает как физический, так и канальный уровни модели ISO/OSI. В ней описан один стандарт для инфракрасных частот и два — для радиочастот. Последнее обстоятельство обусловлено вынужденным компромиссом между двумя «лагерями» производителей оборудования для беспроводных сетей. К одному относятся компании Lucent Technologies, Digital Equipment, Persoft и другие — приверженцы технологии Direct Sequence, использующей для передачи данных широкий диапазон частот. Похоже, что при небольших объемах передаваемых данных эта технология оптимально использует ресурсы сети и, следовательно, более подходит для конфигураций с относительно маленьким числом устройств. Их конкуренты — компания Xircom с партнерами — продвигают на рынок технологию *переменной частоты* (frequency hopping). Здесь при передаче данных используются переменные значения как частот, так и временных промежутков. Эта технология больше подходит для реальных полей, поскольку лучше, чем Direct Sequence, защищена от случайных помех. Обе технологии обеспечивают скорость передачи данных от 1 до 2 Мбит/с. Инфракрасный стандарт обеспечивает такие же скорости и не требует расположения взаимодействующих устройств на линии прямой видимости, но, с другой стороны, инфракрасный сигнал, в отличие от радиоволн, не проходит сквозь стены, закрытые двери и иные подобные преграды.

Стандарт IEEE 802.11 определяет только уровень MAC, что способствует лучшей совместимости и позволяет в будущем реализовать его в микросхемах, а затем расширить область применения и уменьшить стоимость локальных сетей стандарта IEEE 802.11. Этот уровень MAC использует *Множественный доступ с контролем несущей и предупреждением коллизий* (CSMA/CA). Вспомните, что в технологии Ethernet применяется протокол CSMA/CD (с обнаружением коллизий). В самом стандарте описаны целых два алгоритма предупреждения коллизий.

Первый использует *точечную управляющую функцию* (point coordination function, PCF) и в соответствии со спецификацией IEEE 802.11 не обязателен. По своей природе PCF не является распределенной, в резуль-

тате решения по предупреждению коллизии принимает одно устройство. Оно опрашивает остальные устройства и может повышать приоритет некоторых из них. Время работы PCF ограничено, так как периодически вместо нее используется распределенный интеллектуальный алгоритм доступа к среде, названный *распределенная управляющая функция* (distributed coordination function, DCF). Основной инструмент предупреждения коллизий для спецификации IEEE 802.11 — именно DCF. Спецификация IEEE 802.11 для уровня MAC требует, чтобы принимающая сторона подтверждала прием адресованных лично ей кадров. Столкновения кадров можно избежать еще посредством «рукопожатия», когда отправитель сначала посылает «запрос на передачу» и ожидает приема «подтверждения передачи». Такая схема гарантирует предупреждение коллизий, но в ущерб оптимальности использования сети.

Ссылки

http://www.itu.ch

http://www.standards.ieee.org

http://www.ansi.org

http://www.adsl.com (домашняя страница Форума ADSL)

http://www.gigabit-ethernet.org

http://www.iol.unh.edu/consortiums/ge/index.html (Консорциум Gigabit Ethernet)

RFC 2333, «NHRP Protocol Applicability Statement»

RFC 2332, «NBMA Next Hop Resolution Protocol»

RFC 2331, «ATM Signaling Support for IP over ATM—UNI Signaling 4.0 Update»

RFC 2226, «IP Broadcast over ATM Networks»

RFC 2225, «Classical IP and ARP over ATM»

RFC 2149, «Multicast Server Architectures for MARS-Based ATM Multicasting»

RFC 2022, «Support for Multicast over UNI 3.0/3.1 – Based ATM Networks»

RFC 1932, «IP over ATM: A Framework Document»

RFC 1755, «ATM Signaling Support for IP over ATM»

RFC 1626, «Default IP MTU for Use over ATM AAL5»

RFC 1577, «Classical IP and ARP over ATM»

RFC 1483, «Multiprotocol Encapsulation over ATM Adaptation Layer 5»

ГЛАВА 3

Протоколы сетевого и транспортного уровней

В этой главе рассмотрены главным образом протоколы сетевого и транспортного уровней (уровни 3 и 4) эталонной модели ISO/OSI.

IP

Сеть Интернет состоит из множества хост-компьютеров и маршрутизаторов. Подключенные друг к другу компьютеры образуют сети, а те, в свою очередь, — «острова», соединенные маршрутизаторами. Чтобы хост-компьютер или маршрутизатор мог принимать и отправлять данные, он должен иметь уникальный адрес. Протокол IP (Internet Protocol) определяет формат адресов и механизм (с негарантированной доставкой) передачи данных в виде *датаграмм* (datagram), которые вкладываются в поле данных нижележащего протокола, например Ethernet, Token-Ring, ATM (Asynchronous Transfer Mode) или PPP (Point-to-Point Protocol).

IP-адресация

В протоколе IP версии 4 (IPv4) сетевые адреса имеют длину 32 бита. Распределением IP-адресов занимается организация IANA (Internet Assigned Numbers Authority), чей Web-узел Вы найдете по адресу *http://www.iana.org*. Все множество IP-адресов разделено на несколько категорий, называемых *классами адресов* (табл. 3-1).

Таблица 3-1. *Классы IP-адресов*

Класс	Старшие биты IP-адреса	Маска подсети	Число возможных сетей	Число возможных хостов
Класс A	0	255.0.0.0	126	16 777 214
Класс B	10	255.255.0.0	16 384	65 534
Класс C	110	255.255.255.0	2 097 152	254
Класс D (групповое вещание)	1110	Нет	Нет	Нет

Обычно IP-адреса представлены в *десятичном формате с точками* (dotted decimal), разделяющими байты (со значениями от 0 до 255), например: 255.25.25.25. С каждым IP-адресом связана 32-разрядная *маска подсети*, которая при побитном умножении на IP-адрес делит последний на две части. Первая — уникальный идентификатор сети (адрес сети), вторая — уникальный идентификатор хоста в пределах данной сети (адрес хоста).

Организация, получающая адреса класса B, не обязана размещать все 16 000 компьютеров в одном месте. Можно составить подсети из пары сотен компьютеров в каждой. Для этого надо просто переопределить маску подсети, увеличив в ней количество бит, равных 1. Этот процесс называют *делением на подсети* (subnetting); он позволяет сократить количество адресов, доступных для компьютеров и маршрутизаторов за счет увеличения максимального числа возможных подсетей.

Некоторые IP-адреса имеют специальное назначение и не могут быть выданы компьютерам или маршрутизаторам, например адрес, все биты которого равны 0 или 1, а также некоторые другие. В следующей таблице приведено еще несколько примеров.

IP-адрес	Назначение
255.255.255.255	Широковещание в локальной сети
<сеть>.255.255.255	Широковещание в сети класса A (здесь и далее *<сеть>* — идентификатор сети)
<сеть>.255.255	Широковещание в сети класса B
<сеть>.255	Широковещание в сети класса C
<127>.*любой*	Адрес локальной заглушки

Групповая адресация

Класс D IP-адресов зарезервирован для *группового вещания* (multicast). К нему относятся IP-адреса от 224.0.0.0 до 239.255.255.255. Такие IP-адреса представляют логические группы узлов, причем сами узлы (несколько компьютеров и/или маршрутизаторов) могут территориально находиться далеко друг от друга. Так же как в обычной адресации, некоторые групповые IP-адреса зарезервированы и имеют специальное назначение. Например, 224.0.0.1 обозначает все системы данной подсети, а 224.0.0.2 — все маршрутизаторы данной подсети. Подробнее об этом — в RFC 1918, 1519, 1518 и 1466.

IP-датаграммы

IP-датаграмма состоит из IP-заголовка и данных, которые часто называют *полезной нагрузкой* (payload). Длина IP-заголовка не фиксирована (рис. 3-1).

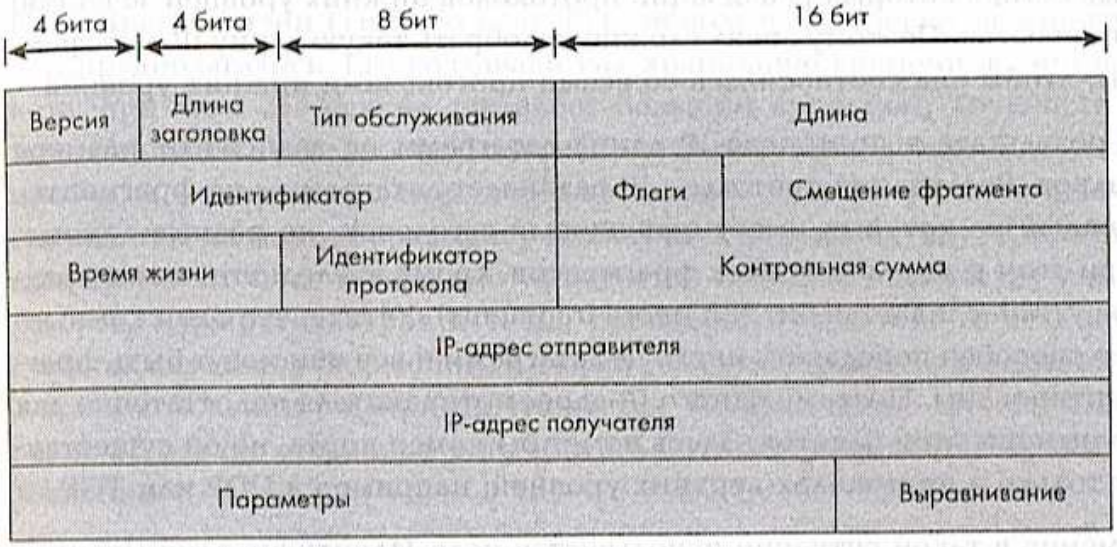


Рис. 3-1. *IP-заголовок*

Поле **Версия** (Version) состоит из четырех бит и обычно равно 4, обозначая четвертую версию протокола IP. Когда протокол IPv6 получит большее распространение, общепринятым значением станет 6. Версия 5 протокола IP была экспериментальной.

Поле **Длина заголовка** (Internet Header Length, IHL) состоит из 4 бит и содержит длину IP-заголовка в 32-разрядных словах. Минимальная длина IP-заголовка составляет 20 байт (пять 32-разрядных слов).

Поле **Тип обслуживания** (Type Of Service, TOS) состоит из 8 бит, в нем указывают необходимость специальной обработки пакета, например минимизации задержки, повышения скорости и т. д. Подробности — в RFC 791. В настоящее время большинство маршрутизаторов Интернета просто игнорируют это поле.

Поле **Длина** (Length) состоит из 16 бит и характеризует общую длину (в байтах) IP-пакета. Из этого следует, что IP-пакеты не могут быть длиннее 65 535 байт, а поскольку IP-заголовок занимает как минимум 20 байт, то полезная нагрузка составит не более 65 515 байт. Все хосты и маршрутизаторы должны поддерживать IP-пакеты длиной не менее 576 байт (20 байт — IP-заголовок, 512 байт — полезная нагрузка, 44 байта — параметры или заголовок протокола нижнего уровня). Это требование

гарантирует, что IP-пакеты такой длины будут передаваться без фрагментации. (Фрагментация описана далее в этом разделе.)

Сами IP-датаграммы вкладываются в кадры протоколов нижнего уровня, например Ethernet или TokenRing, и могут пересылаться через разные сети, в которых реализации протоколов нижних уровней зачастую отличаются. Поэтому очень сложно подобрать такую длину IP-датаграммы, чтобы она соотносилась со всеми протоколами нижних уровней.

В результате в протоколе IP длина датаграмм не зависит от размера кадров. Реализация протокола IP разбивает датаграммы на фрагменты, каждый из которых имеет одинаковый заголовок, но разные данные. При этом в заголовках всех фрагментов, кроме последнего, в поле флагов устанавливается бит MF (more fragments). Учтите, что один компьютер способен передавать много IP-датаграмм и все они могут быть фрагментированы. Поэтому одного IP-адреса отправителя недостаточно для идентификации пакетов. Здесь помог бы номер порта, но он существует только в протоколах верхних уровней, например в UDP или TCP.

Именно в такой ситуации используется поле **Идентификатор** (Identification). Оно имеет одинаковое значение во всех фрагментах, составляющих одну IP-датаграмму, и определяется хостом-отправителем. Поле **Смещение фрагмента** (Fragment Offset) состоит из 13 бит и задает место фрагмента при сборке исходного потока данных. Для первого фрагмента значение смещения равно 0. Не забывайте, что разные фрагменты передаются как отдельные IP-пакеты и, возможно, по разным маршрутам. Следовательно, окончательная сборка выполняется только адресатом, а не промежуточным маршрутизатором. Механизмы сборки фрагментов описаны в RFC 815 и 791.

Поле **Флаги** (Flags) состоит из 3 бит. Старший бит всегда имеет значение 0. Следующий бит называется «Don't Fragment» (Не фрагментировать) и, установленный в 1, говорит о недопустимости фрагментации данной датаграммы. Последний бит — «More Fragments» (Еще фрагменты) — равен 1 во всех фрагментах одной датаграммы, кроме последнего.

Из-за увеличения скоростей передачи данных в сетях, фрагментация порождает проблемы. Рассмотрим следующую ситуацию. Предположим, что передаются IP-пакеты максимальной длины, и они подвергаются фрагментации. Это значит, что до повторного использования значения поля **Идентификатор** будет передано $(2^{16} - 1)$ пакетов, каждый по $(2^{16} - 1)$ байт. Здесь не учтены затраты на протоколы IP и нижних уровней. Таким образом, общее количество бит составит примерно $2^3 x 2^{16} x 2^{16}$, то есть

2^{35}, или около 32 гигабит. При скорости 1 Гбит/с одно и то же значение идентификатора будет использоваться каждые 32 секунды. (Реальное время и того меньше, поскольку описана наиболее благоприятная ситуация.)

Поле **Время жизни** (Time To Live, TTL) используется несколько иначе, чем предполагалось. Его создавали как хранилище времени жизни пакета, при этом 255 секунд считалось большим временем. Трудность в том, что оценить реально необходимое значение очень сложно. Еще одна проблема — маршрутизатор, который «видит» пакет в течение миллисекунд, тем не менее сбрасывает со счетчика целую секунду. В настоящее время это поле используют в качестве счетчика количества узлов (хостов, маршрутизаторов и т. п.), пройденных пакетом, то есть как *счетчик переходов* (hop count). Каждый узел уменьшает его на единицу, а при достижении 0 пакет уничтожается. По умолчанию его значение равно 32, однако отправитель (то есть его IP) может присвоить ему любое значение.

Поле **Идентификатор протокола** (Protocol Identifier) состоит из 8 бит и показывает, какой протокол верхнего уровня вложен в поле данных IP-пакета. Основные его значения приведены в табл. 3-2.

Таблица 3-2. *Значения поля* ***Идентификатор протокола***

Значение	Протокол
1	Internet Control Message Protocol (ICMP)
2	Internet Group Management Protocol (IGMP)
6	Transmission Control Protocol (TCP)
7	Зарезервировано
8	Exterior Gateway Protocol (EGP)

Это поле используется в IP для выбора одного из протоколов верхнего уровня, которому следует передать принятый пакет, например UDP (User Datagram Protocol) или TCP (Transmission Control Protocol). Самые последние и полные сведения об идентификаторах протоколов — в RFC 1700.

Поле **Контрольная сумма** (Checksum) применяется для выявления ошибок передачи пакета. Здесь контрольная сумма — это двоичное дополнение 16-разрядной суммы по содержимому заголовка. Имейте в виду, что поскольку заголовок изменяется (например, в поле **TTL**), то контрольная сумма пересчитывается каждым узлом, в том числе и всеми промежуточными.

Поле **Параметры** (Options) имеет переменную длину и может вообще отсутствовать (длина 0 тоже допустима). Возможны два формата этого поля:

- один байт — тип параметра;
- несколько байт — тип параметра, длина и значение.

Тип параметра состоит из трех подполей.

- Флаг COPIED (1 бит). Значения 1 или 0 свидетельствуют о том, необходимо или нет скопировать параметр во все фрагменты датаграммы.
- Класс параметра (2 бита). Значение 0 — контроль, 2 — отладка. Значения 1 и 3 зарезервированы.
- Номер параметра (5 бит).

Информация о типах параметров IP собрана в табл. 3-2. Помните, что числовое значение типа параметра можно вычислить по предоставленной здесь информации. Подробное описание — в RFC 791.

Таблица 3-3. *Параметры IP*

Класс параметра	Номер параметра	Длина в байтах	Флаг COPIED	Описание
0	0	1	0 или 1	Обозначает конец списка параметров
0	1	1	0 или 1	Используется для выравнивания длины списка параметров
0	2	Переменная	1	Параметры безопасности. В Интернете применяется не он, а средства безопасности IP
0	3	Переменная	1	Позволяет маршрутизатору отказаться от маршрутизации источника и записать новый маршрут датаграммы. При маршрутизации источника отправитель указывает полный маршрут, которого должны придерживаться промежуточные маршрутизаторы
0	7	Переменная	0	Используется для записи маршрута. Каждый маршрутизатор заносит свой IP-адрес в указанное место и обновляет значение указателя, чтобы тот определял место для адреса следующего маршрутизатора

Таблица 3-3. *Параметры IP* (продолжение)

Класс параметра	Номер параметра	Длина в байтах	Флаг COPIED	Описание
0	8	4	1	Идентификатор потока; введен для нужд ныне свернутой сети Atlantic Satellite. Сейчас не используется; применялся только в IP версии 5
0	9	Переменная	1	Запрещает маршрутизаторам отказываться от маршрутизации источника и хранит предполагаемый маршрут датаграммы. «Строгая» маршрутизация отличается от «мягкой» тем, что промежуточные маршрутизаторы поставлены в более жесткие условия
2	4	Переменная	0	Используется для записи времени обработки датаграммы

Поле **Выравнивание** (Padding) — это лишь дополнительно занятое место, добавленное для того, чтобы IP-заголовок заканчивался на 32-ом байте. Например, если поле **Параметры** заканчивается на 30-ом байте, то поле **Выравнивание** состоит из 2 байт, а если поле **Параметры** заканчивается на 50-ом байте, то **Выравнивание** — из 14 байт.

Порты

Приложения должны иметь возможность определять, с кем или с чем они взаимодействуют. Одних IP-адресов для этого явно недостаточно, поскольку зачастую на одном компьютере одновременно выполняются несколько приложений, которые могут взаимодействовать с другими, расположенными на разных или даже на одном удаленном компьютере. Предположим, например, что приложение A, работающее на компьютере 1, желает установить связь с приложением B на компьютере 2. В то же время приложению C, работающему на компьютере 3, необходимо связаться с приложением D на компьютере 2. Можно назначить компьютеру 2 несколько IP-адресов и использовать их для приложений B и D. Но это приведет к перерасходу адресного пространства. Поэтому связь устанавливается через порты. Ниже рассказано, как в протоколах UDP и TCP порты идентифицируют передающие и принимающие приложения. Адреса отправителя и получателя идентифицируют только компьютеры, на которых выполняются приложения. Таким образом, канал (или поток данных) однозначно идентифицируется набором из четырех значений:

- IP-адресом отправителя;
- номером порта отправителя;
- IP-адресом получателя;
- номером порта получателя.

Некоторые номера портов закреплены за важными и наиболее популярными приложениями. Такие предопределенные номера портов используются только серверами. У клиентов порты распределяются динамически. Один порт сервера способен обслуживать несколько потоков данных от разных подключенных к нему клиентов. Это ничему не противоречит, поскольку сочетания четырех описанных выше параметров уникальны. Важность номеров портов тем более увеличивается, поскольку некоторые межсетевые экраны, описанные дальше в этой главе, используют их для задания правил фильтрования трафика.

UDP

Протокол UDP (User Datagram Protocol) — это простейшая надстройка над IP. Он обеспечивает передачу датаграм (негарантированная доставка данных без установления логических соединений), но для непосредственной работы использует IP. Вот какие дополнительные возможности характерны для протокола UDP:

- уплотнение и разуплотнение канала (посредством портов UDP) — теперь сразу несколько приложений клиента, сервера или обоих могут иметь собственные потоки датаграм;
- вычисление контрольной суммы и проверка целостности принимаемых датаграм. Вспомните, что в IP контрольная сумма затрагивает только IP-заголовок, а не все данные. В UDP контрольные суммы вычисляются на основе всего пакета, включая данные.

Протокол UDP не обеспечивает большей надежности передачи данных, чем заложена в нижележащей реализации протокола IP. Приложения, использующие UDP, должны сами повторять передачу утерянных датаграмм и выявлять дублируемые, переупорядочивать датаграммы (иногда они прибывают не в том порядке, в каком были отправлены), управлять потоком данных и т. д.

Пакет протокола UDP состоит из заголовка и данных, и все это вкладывается в данные IP-пакета. Формат UDP-заголовка показан на рис. 3-2.

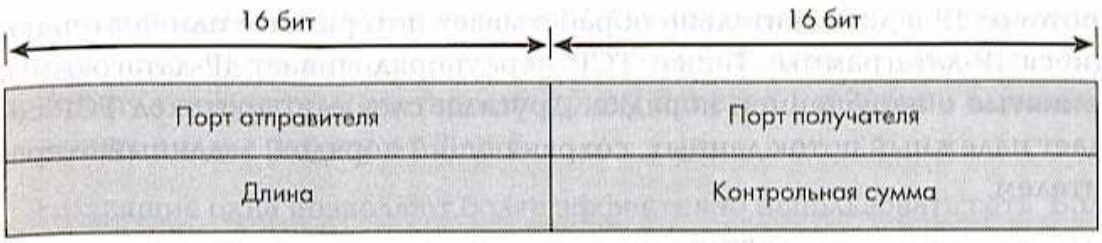


Рис. 3-2. *UDP-заголовок*

Каждое поле UDP-заголовка состоит из 16 бит. Номера портов отправителя и получателя используются при уплотнении и разуплотнении потоков для корректной доставки датаграмм приложениям. В поле **Длина** (Length) указана длина в байтах UDP-пакета, включая заголовок, но без IP-заголовка и IP-параметров. Поле **Контрольная сумма** (Checksum) — это 16-разрядное двоичное дополнение суммы байт псевдозаголовка (содержащего информацию из IP-заголовка), UDP-заголовка и данных. Формат псевдозаголовка показан на рис. 3-3.

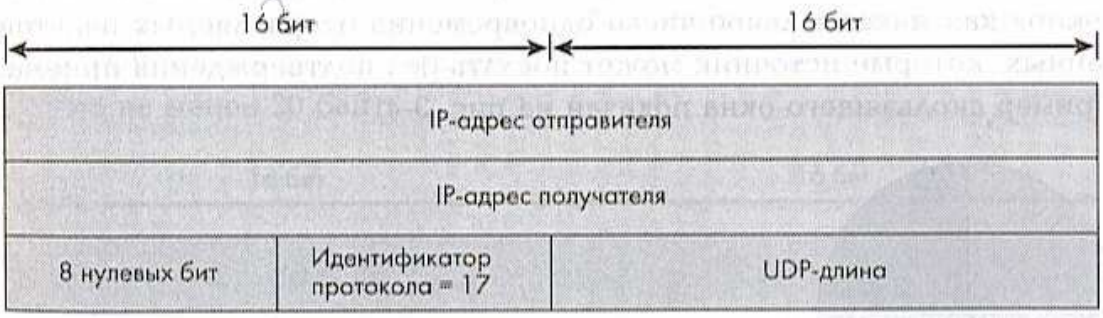


Рис. 3-3. *Псевдозаголовок UDP*

Таким образом, **Контрольная сумма** вычисляется так: в начало UDP-пакета добавляется псевдозаголовок (не физически, но логически), а затем вычисляется контрольная сумма по псевдозаголовку и UDP-пакету. Если контрольная сумма не определена, то в соответствующем поле передается значение 0. Если же реальная контрольная сумма окажется равной 0, то все биты поля будут установлены в 1. Псевдозаголовок UDP подробно описан в RFC 768.

TCP

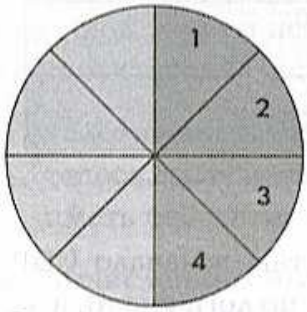
Сам протокол IP не гарантирует надежности передачи данных между двумя компьютерами. О нем иногда говорят «отправил и молись». В свою очередь, протокол TCP (Transmission Control Protocol) обеспечивает надежный, защищенный от ошибок, полнодуплексный канал связи между двумя компьютерами. Протокол TCP использует для передачи данных

протокол IP и дополнительно обрабатывает потерянные или повторяющиеся IP-датаграммы. Также TCP переупорядочивает IP-датаграммы, принятые с нарушением порядка. Другими словами, протокол TCP создает надежный поток данных, сохраняющий порядок, заданный отправителем.

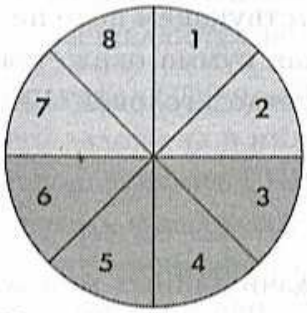
Все эти возможности TCP осуществляются путем разбиения потока данных на пакеты, которые достаточно малы, чтобы поместиться в IP-датаграммы. Эти пакеты называют *сегментами* (segments). При нумерации и отправке IP-датаграмм применяется принцип «подтверждения приема и повторной передачи». Получатель посылает явное или неявное подтверждение каждой принятой IP-датаграммы. Отправитель ожидает некоторое время и, если не получает подтверждения, повторно отправляет IP-датаграмму.

Скользящее окно

Механизм «скользящее окно» (Sliding Window) определяет понятие «окно» как максимальное число одновременно отправляемых пакетов данных, которые источник может послать без подтверждения приема. Пример скользящего окна показан на рис. 3-4.



Скользящее окно отправителя = пакеты 1,2,3,4



Скользящее окно отправителя = пакеты 3,4,5,6

Рис. 3-4. *Скользящее окно*

На этом рисунке размер окна равен 4. Положение окна, изображенного вверху, позволяет передавать первые четыре пакета с номерами 1, 2, 3 и 4. Когда получатель подтвердит прием пакетов 1 и 2, окно сдвинется, и станут видны пакеты от 3 до 6.

Скользящие окна позволяют более эффективно использовать сеть. Большинство сетей являются полнодуплексными, что подразумевает одновременную передачу и прием данных компьютером. Без скользящего окна компьютер передавал бы один пакет, ждал бы его подтверждения, затем передавал бы второй и т. д. При этом в каждый момент времени сеть используется только в одном направлении, называют *старт-стопным* протоколом. В простейшем виде скользящее окно имеет размер 1, то есть сеть работает по старт-стопному протоколу. Обратите внимание, что в протоколе TCP размер скользящего окна может динамически изменяться, чтобы соответствовать состоянию сети.

Заголовок TCP-сегмента

На рис. 3-5 изображен заголовок TCP-сегмента, часто называемый просто «TCP-заголовок». Заголовки TCP-сегментов имеют переменную длину, но не менее 20 байт.

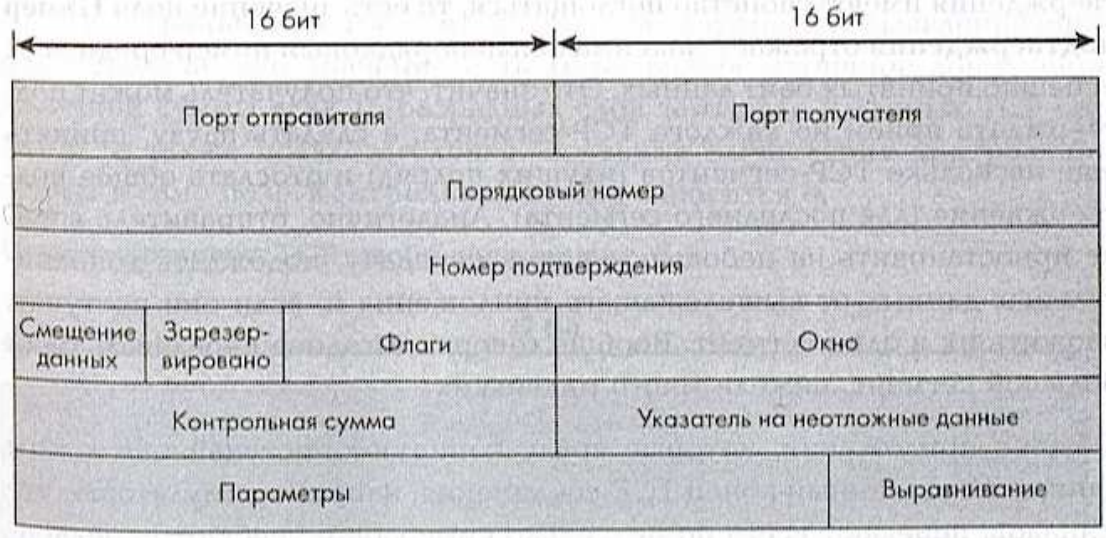


Рис. 3-5. *Заголовок TCP-сегмента*

Прежде чем знакомиться с каждым из полей, обратите внимание на то, что в заголовке нет информации о количестве байт данных в TCP-сегменте. Поскольку TCP-сегмент передается как полезная нагрузка IP-датаграммы, то длина данных протокола TCP вычисляется как разность между размером полезной нагрузки IP и размером TCP-заголовка.

Как говорилось ранее, протокол TCP обеспечивает надежную связь двух процессов. Конечные точки этого потока данных однозначно идентифицируются через:

- поле **IP-адрес отправителя** в заголовке IP-датаграммы;
- поле **IP-адрес получателя** в заголовке IP-датаграммы;
- поле **Порт отправителя** в заголовке TCP-сегмента;
- поле **Порт получателя** в заголовке TCP-сегмента.

Поле **Порядковый номер** (Sequence number) отражает позицию первого байта данных текущего сегмента в общем потоке данных. Если установлен флаг SYN (см. ниже), то это поле хранит начальный порядковый номер (Initial Sequence Number, ISN), а порядковый номер первого байта данных равен ISN + 1.

Если установлен флаг ACK, то поле **Номер подтверждения** (Acknowledgement number) содержит значение, которое отправитель этого сегмента ожидает увидеть в поле **Порядковый номер** следующего сегмента (другими словами, это порядковый номер последнего успешно принятого байта данных, увеличенный на 1, в общем потоке данных). Подтверждения имеют свойство поглощаться, то есть значение поля **Номер подтверждения** отражает максимальный порядковый номер среди всех успешно принятых байт данных. Это значит, что получатель может подтверждать прием не каждого TCP-сегмента, а сделать паузу, принять еще несколько TCP-сегментов (идущих подряд) и отослать общее подтверждение (для последнего сегмента). Аналогично, отправитель вправе приостановить на небольшое время передачу, подождать дополнительных данных от вышележащего приложения и, если они поступят, вложить их в один сегмент. Вообще говоря, выгоднее передавать один большой сегмент, нежели много маленьких.

Обычно приложения, которые сразу копируют поступающие к ним данные на удаленный конец TCP-соединения, например эмуляторы терминалов, передают всего по несколько символов и требуют немедленной их отправки. Поэтому если отправитель буферизует данные, то такие приложения не очень стабильно работают. Для решения этого вопроса в протоколе TCP предусмотрена возможность принудительной передачи. Соответствующий параметр находится в 6-разрядном поле **Флаги** (Flags), значения которого перечислены в табл. 3-4.

Таблица 3-4. *Значения поля* ***Флаги***

Значение	Описание
32	URG: есть срочные данные
16	ACK: подтверждение приема данных
8	PSH: как можно быстрее передать данные приложению
4	RST: переустановить TCP-соединение
2	SYN: синхронизация соединения (см. поле **Порядковый номер** заголовка TCP-сегмента)
1	FIN: конец передачи; этот абонент готов разорвать соединение

Поле **Смещение данных** (Data Offset) состоит из 4 бит и отражает, сколько 32-разрядных слов занимает TCP-заголовок. Оно необходимо, поскольку TCP-заголовок имеет переменную длину. Если требуется, заголовок дополняется нулевыми байтами.

Поле **Зарезервировано** (Reserved) состоит из 6 нулевых бит.

Поле **Окно** (Window) относится к описанному выше механизму «скользящего окна». Его значение показывает количество байт, которое готов принять отправитель (текущий размер окна). Позицию данных, которые ожидает принять получатель, указывает порядковый номер.

Поле **Контрольная сумма** (Checksum) используется для выявления ошибок передачи. Его значение есть 16-разрядное двоичное дополнение двоичной суммы всех 16-разрядных слов заголовка и данных TCP-сегмента, а также псевдозаголовка. Во время вычисления контрольной суммы в само поле **Контрольная сумма** заносится 0.

Псевдозаголовок TCP изображен на рис. 3-6.

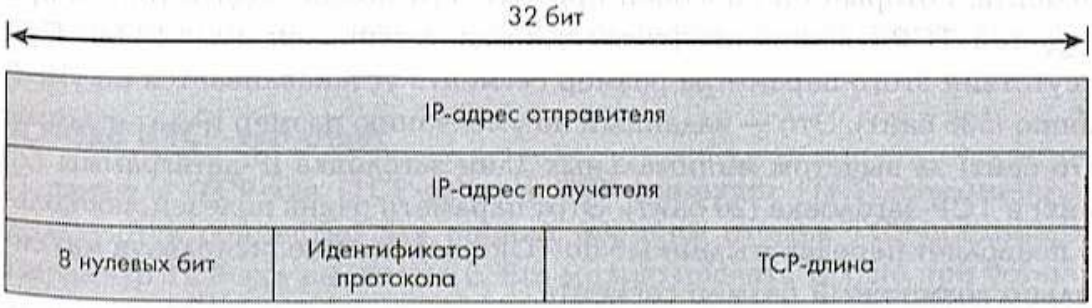


Рис. 3-6. *Псевдозаголовок TCP*

Поле **Указатель на неотложные данные** (Urgent Pointer) заголовка TCP-сегмента состоит из 16 бит. Его сумма с полем **Порядковый номер** дает порядковый номер байта, следующего непосредственно за срочными данными. Таким образом, реализации протокола TCP позволяют инфор-

мировать приложения верхних уровней о наличии срочных данных и об их окончании.

Поле **Параметры** (Options) имеет переменную длину. Существует много различных параметров, и в TCP-заголовке их может быть больше одного. Это поле имеет два разных формата:

- один байт — тип параметра;
- несколько байт — тип параметра, длина и значение.

Ниже описаны некоторые типы параметров. Это не полный список, а более подробную информацию Вы найдете в RFC 1323, 1185, 1072 и в других.

Конец списка параметров

Параметр **Конец списка параметров** (End Of Option List) обозначается типом 0, длиной 1 и не имеет числового значения. Этот параметр всегда указан последним в списке параметров.

Нет операции

Параметр **Нет операции** (No Operation) обозначается типом 1, длиной 1 и не имеет числового значения. Его используют для выравнивания следующего параметра на 32-разрядную границу.

Максимальный размер принимаемого сегмента

Параметр **Максимальный размер принимаемого сегмента** (Maximum Receive Segment Size) обозначается типом 2 и длиной 4, а значение и есть максимальный размер принимаемого сегмента. Этот параметр используется отправителем для указания максимального размера TCP-сегмента, который он способен принять; его можно задать только при создании TCP-соединения (точнее, когда установлен флаг SYN). При отсутствии этого параметра размер сегмента устанавливается по умолчанию (536 байт). Это — заданный по умолчанию размер IP-датаграммы (576 байт) за вычетом минимальных длин заголовка IP-датаграммы (20 байт) и TCP-заголовка (20 байт). Этот параметр очень полезен, поскольку позволяет передавать данные по TCP-соединению, используя максимально возможный размер сегмента.

Масштаб окна

Параметр **Масштаб окна** (Window scale) обозначается типом 3 и длиной 3, а его значение отражает степень изменения масштаба принимающего окна.

Этот параметр очень полезен в глобальных сетях, для которых свойственны большие задержки передачи, но высокая пропускная способность каналов. Обычно максимальный размер окна составляет 65 535 байт (максимальное 16-разрядное число). Но представьте международное соединение, где время отзыва* может достигать 30 миллисекунд. А при скорости 4 Мбит/с для передачи целого окна потребуется лишь 14,5 миллисекунд. Согласитесь, это не самое эффективное использование ресурсов сети, причем с ростом пропускной способности ситуация только ухудшается. Параметр **Масштаб окна** позволяет увеличить размер окна далеко за 65 535 байт. Так, если значение этого параметра равно *N*, то реальный размер окна равен *<размер_окна>*x2^N. Поскольку масштабный множитель является степенью двойки, то такое умножение равносильно побитовому сдвигу влево на N разрядов, поэтому параметр иногда называют **Сдвиг окна** (Window Shift). Естественно, он используется только по договоренности обеих сторон TCP-соединения.

Разрешен SACK

Параметр **Разрешен SACK** (SACK Permitted) обозначается типом 3 и длиной 2. Он свидетельствует, что отправитель допускает выборочное подтверждение (SACK, Selective Acknowledgement). Этот параметр можно переслать только при установлении TCP-соединения. Описание SACK — в следующем разделе.

SACK

Параметр **SACK** (Selective Acknowledgement) обозначается типом 5 и имеет переменную длину. Он показывает, что принято несколько блоков, но некоторые промежуточные утеряны. Это несколько повышает эффективность. Представьте ситуацию, когда передано 1 000 блоков данных, а приняты все, кроме 500-го. Если используется выборочное подтверждение, то только один блок (с номером 500) будет отправлен повторно.

TCP-эхо и TCP-эхо-ответ

Параметры **TCP-эхо** (TCP-echo) и **TCP-эхо-ответ** (TCP-echo-reply) используются совместно для оценки времени отзыва TCP-соединения. Параметр **TCP-эхо** имеет тип 6. Его можно послать только при установлении TCP-соединения. Станция, принявшая пакет с параметром **TCP-эхо**, должна в ответ отправить пакет с параметром **TCP-эхо-ответ** (он имеет тип 7 и длину 6). Знать время отзыва достаточно важно, поскольку

* Минимальное время, за которое данные проходят в одном направлении и обратно. — *Прим. перев.*

ку его занижение приводит к лишним повторам передачи, в то время как завышение — к бесполезному ожиданию подтверждения, которое уже никогда не придет.

Так же как в случае IP-заголовка, поле **Выравнивание** (Padding) — это незначащее занятое пространство, гарантирующее, что TCP-заголовок оканчивается на 32-разрядной границе.

Зондирование пустым окном

В протоколе TCP предусмотрен механизм «зондирования пустым окном» (Zero Window Probe). Отправив полное окно данных, передающий узел ожидает подтверждения приема. В случае утери подтверждения возникает тупиковая ситуация, когда передача данных прекращается вовсе. Чтобы предотвратить это, узел-отправитель посылает пакет с 1 байтом данных — «пустое окно». Принимающий узел обязан на него ответить. Получение ответа свидетельствует о том, что принимающий узел исправен, но действительно не может принять больше данных (вероятно, из-за того, что приложение, находящееся выше протокола TCP, все еще обрабатывает предыдущую порцию). Таким образом, зондирование «пустым окном» проводят при невозможности дальнейшей передачи данных, но чтобы сохранить TCP-соединения. Первое «пустое окно» отправляется по истечении стандартного времени ожидания. Если ответа нет, то интервалы между последующими «пустыми окнами» экспоненциально увеличиваются.

Синдром «глупого окна»

В RFC 1122 описан феномен, получивший название синдром «глупого окна» (Silly Window Syndrome), который приводит к значительному снижению производительности и неэффективному использованию ресурсов сети. Возникает синдром «глупого окна», когда окно принимающего узла слишком мало. Причиной может быть действительно маленький размер окна либо такое стечение обстоятельств, когда необходимо дополнительно отослать небольшой объем данных во время ожидания подтверждения приема уже отправленных.

Представьте ситуацию: размер принимающего окна равен 1 024 байтам, и отправлено два сегмента по 512 байт каждый. Получатель подтверждает прием первого из них. Когда это подтверждение поступает к отправляющему узлу, он устанавливает размер окна равным 512 байтам. Предположим далее, что приложение запросило срочную передачу сегмента из 64 байт и еще одного — из 512 байт. Узел-отправитель посылает эти 64 байта и дополнительно 448 байт (512 – 64). Если приемник подтвердит получение срочных данных, то окно отправителя уменьшится

до 64 байт. В результате — отправляются короткие TCP-сегменты, а значит, ресурсы сети используются неэффективно.

В RFC 1122 подробно описано, как избежать возникновения синдрома «глупого окна». В решении проблемы участвуют и отправляющий и принимающий узлы. Когда размер передаваемого сегмента слишком мал, отправитель задерживает передачу всех данных, за исключением срочных. Длительность задержки зависит от времени отзыва. Со своей стороны, принимающий узел не должен продвигать окно маленькими порциями. Также он обязан выдерживать паузу перед отправкой подтверждений, ожидая, что прибудут еще данные, и тогда он отправит одно общее подтверждение. Эта пауза тоже зависит от времени отзыва, но с учетом того, чтобы передающий узел не успел начать повторную передачу данных.

Эффективность работы TCP зависит от правильности настройки многих параметров и является отдельной задачей даже для сети с известными характеристиками (например, локальной). При изучении других сетей, например глобальных или беспроводных (которые более склоны к утере информации и к разрыву соединений), обеспечение эффективной работы протокола TCP еще более осложняется — об этом можно написать отдельную книгу. Вот некоторые параметры, существенно влияющие на работу протокола TCP:

- размер скользящего окна принимающего узла;
- корректность оценки времени отзыва;
- момент оценки времени отзыва;
- момент отправки подтверждений.

ICMP

Протокол ICMP (Internet Control Message Protocol) используется хостами и маршрутизаторами для обмена управляющей информацией, например, об ошибках или о процессе начальной загрузки. Его считают прикладным протоколом, находящимся выше IP. На рис. 3-7 показан формат ICMP-сообщений и способ их вложения в IP-пакеты.

Все биты, непосредственно относящиеся к ICMP-сообщению, являются полезной нагрузкой IP-пакета. В IP-заголовке поле **Идентификатор протокола** устанавливается равным 1. Учтите, что ICMP хотя и рассматривается как самостоятельный прикладной протокол, но является неотъемлемой частью любой реализации протокола IP, поскольку управляет его работой.

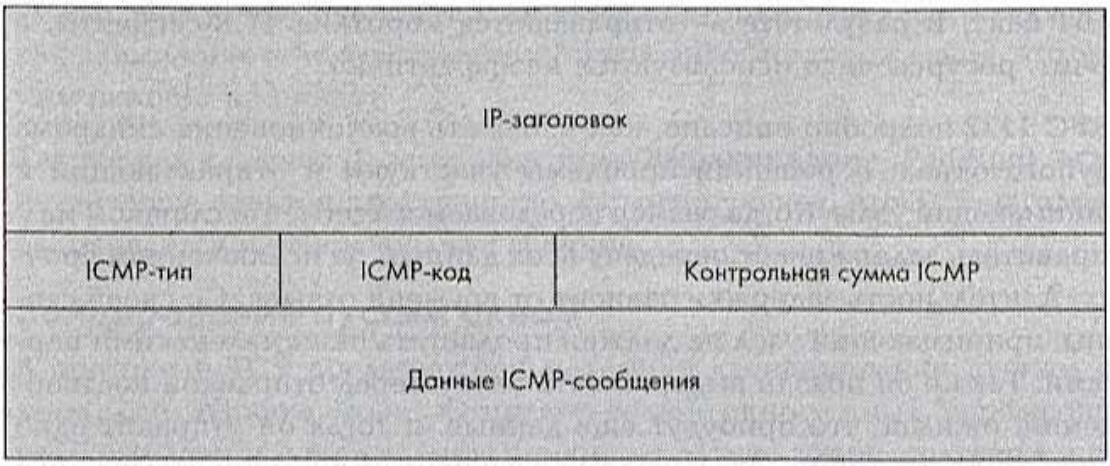


Рис. 3-7. *Формат ICMP-сообщений*

Некоторые ICMP-сообщения генерируются только маршрутизаторами, а остальные — и маршрутизаторами, и хост-компьютерами. Никогда ICMP-сообщения не отправляются в ответ на другие ICMP-сообщения. Так предотвращается рекурсия ICMP-сообщений. Также ICMP-сообщения не генерируются в следующих случаях:

- если принято ICMP-сообщение об ошибке;
- если принята датаграмма с адресом локальной заглушки (loopback) либо групповым адресом;
- если принят не начальный IP-фрагмент.

Существует два типа ICMP-сообщений: ICMP-сообщения об ошибках и ICMP-запросы. Подробности о ICMP-сообщениях и полное описание их формата — в RFC 1256, 1122, 792 и др. Далее в этом разделе описаны только наиболее важные ICMP-сообщения.

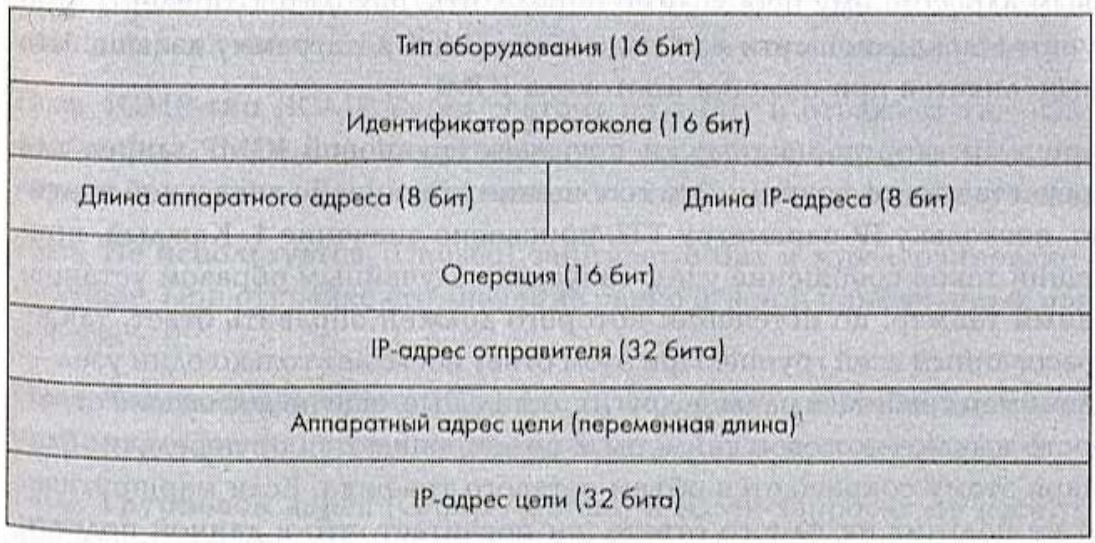
Обычно маршрутизаторы посылают ICMP-сообщения для отчета об ошибках. Если маршрутизатор не обрабатывает датаграмму из-за недостатка ресурсов (например, памяти), то посылает ее отправителю ICMP-сообщение «Подавление источника» (source quench). Если маршрутизатор получает пакет данных для неизвестного адресата, то посылает ICMP-сообщение «Адресат недоступен» (destination unreachable). Чтобы рекомендовать хосту другой, более предпочтительный маршрутизатор для передачи данных к некоторому адресату, маршрутизатор посылает ICMP-сообщение, именуемое «Перенаправление» (redirect). В ICMP-сообщения об ошибках включается заголовок IP-датаграммы, вызвавшей ошибку. Это позволяет ее отправителю выяснить, с какими адресатами возникла ошибка, так как иногда одновременно проводятся несколько раз-

ных сеансов связи. Хосты обычно посылают ICMP-сообщения для проверки качества связи и оценки времени отзыва. Для этого используется ICMP-сообщение «Эхо» (echo). Хост, получивший такое сообщение, должен ответить ICMP-сообщением «Эхо-ответом» (echo reply).

ARP и RARP

В протоколе IPv4 каждому компьютеру, маршрутизатору и т. п. присваиваются логические адреса. С появлением технологии интерактивного видео, каждый телевизор стал потенциальным кандидатом на получение IP-адреса; то же относится к электролампочкам, кондиционерам, обогревателям, тостерам и микроволновым печам. Но поскольку IP-адреса это лишь логические адреса, необходимо установить соответствие между ними и физическими адресами устройств, например адресами сетевых адаптеров Ethernet или TokenRing. Протокол ARP (Address Resolution Protocol) предоставляет средства для поиска физического адреса устройства при известном его IP-адресе. Этот процесс называют *разрешением адреса* (address resolution). Протокол ARP описан в RFC 826.

С переносом протокола IP в среду типа ATM, в RFC 2320 и 1390 был определен ARP-подобный механизм разрешения адресов в сетях, использующих технологию ATM вместо Ethernet. Для обратной процедуры — получения IP-адреса по известному физическому — используется протокол RARP (Reverse Address Resolution Protocol). Но, пожалуй, единственное полезное его применение — при загрузке бездискового компьютера. Протокол RARP описан в RFC 1931 и 903.



[1] Длина аппаратного адреса цели зависит от используемого им протокола второго уровня модели ISO/OSI. Для Ethernet она равна 48 битам.

Рис. 3-8. *Формат ARP/RARP-сообщения*

Формат сообщений протоколов ARP/RARP показан на рис. 3-8.

Поле **Тип оборудования** (Hardware Type) отражает тип используемого оборудования. Например, значение 1 обозначает локальную сеть стандарта Ethernet. Поле **Идентификатор протокола** (Protocol Identifier) определяет протокол, которому принадлежит сообщение. Описаны значения только для ARP и RARP. В случае протокола ARP отправитель указывает только собственные аппаратный и IP-адрес и IP-адрес цели. Сообщения ARP могут отсылаться как широковещательно, так и направленным пакетом к шлюзу, заданному по умолчанию.

Поле **Операция** (Operation) характеризует природу сообщения. Описания значений этого поля приведены ниже.

Значение	Описание
1	ARP-запрос
2	ARP-ответ
3	RARP-запрос
4	RARP-ответ

IGMP

Узел с обычным (не групповым) IP-адресом может самостоятельно входить и выходить из состава *широковещательной группы* (multicast group). Средства для этого предоставляет протокол IGMP (Internet Group Management Protocol). Когда маршрутизатор получает IP-датаграмму с групповым адресом, ему необходимо определить, находятся ли члены этой группы в локальной сети или надо пересылать датаграмму дальше. Что и выполняется при помощи протокола IGMP.

Маршрутизатор периодически посылает групповой IGMP-запрос для переучета членов группы. Это сообщение ограничено локальной подсетью, поскольку IP-параметру TTL присвоено значение 1. Каждый, принявший такое сообщение узел, запускает случайным образом установленный таймер, по истечении которого должен оправить ответ, также адресованный всей группе. При этом ответ посылает только один узел — чей таймер сработает раньше других, остальные, еще не пославшие ответ, просто выключают свои таймеры и воздерживаются от передачи. Благодаря этому сокращается объем сетевого трафика. Если маршрутизатор не получит ни одного ответа, он посчитает, что в данной подсети нет ни одного члена этой группы. Узлы, желающие войти в состав группы, немедленно отсылают этой группе сообщение и обычно повторяют

его раз или два. Полное описание запросов и ответов протокола IGMP приведено в RFC 1112.

Протокол IGMP похож на ICMP: формально, они оба — протоколы верхнего уровня и пользуются услугами IP. На практике, протокол IGMP рассматривается и реализуется как неотъемлемая часть IP, поскольку активно влияет на его работу. Во избежание перегрузки сети, при приеме сообщения с групповым IP-адресом никогда не генерируются IGMP-сообщения об ошибках.

На рис. 3-9 показано IGMP-сообщение. Все данные, относящиеся непосредственно к IGMP, являются полезной нагрузкой IP-пакета.

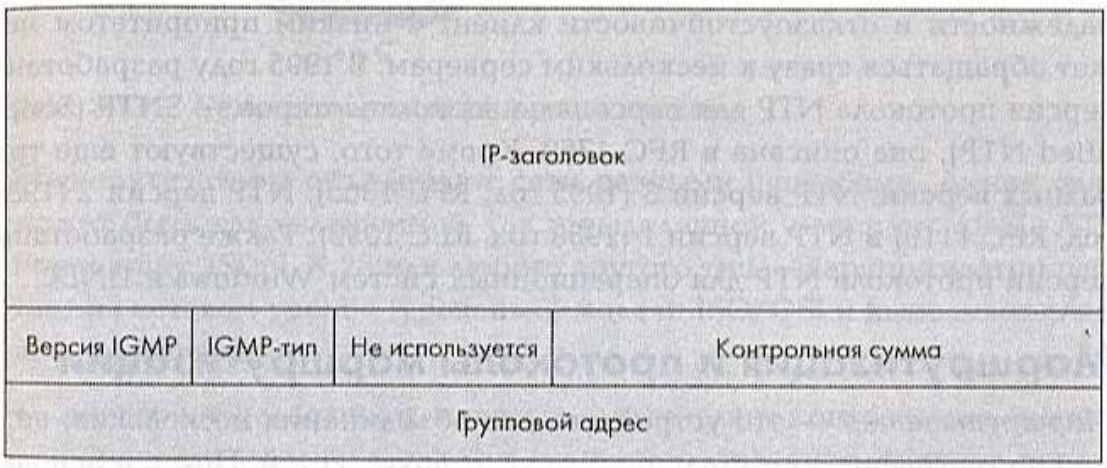


Рис. 3-9. *IGMP-сообщение*

Поле **Версия IGMP** (IGMP Version) состоит из 4 бит и хранит одно число — номер версии протокола IGMP. В RFC 1112 ему присвоено значение 1.

Поле **IGMP-тип** (IGMP Type) состоит из 4 бит и отражает тип IGMP-сообщения. Значение 1 соответствует ответу на запрос о членстве в группе, а значение 2 — сообщению о вхождении в группу нового члена.

Поле **Не используется** (Unused) занимает 8 бит и зарезервировано на будущее. При отправке его значение равно 0, а при приеме оно игнорируется.

Поле **Контрольная сумма** (Checksum) содержит двоичное дополнение суммы сообщения; перед началом вычисления в него заносится 0.

Поле **Групповой адрес** (Group Address) в IGMP-запросах не несет информации. При передаче оно устанавливается равным 0, а при приеме игнорируется. В IGMP-ответах оно хранит IP-адрес широковещательной группы, для которой предназначен этот ответ.

NTP

Протокол NTP (Network Time Protocol) используется для синхронизации времени на компьютерах, объединенных в сеть. Синхронизация выполняется с точностью до миллисекунд. Время передается в формате UTC (Universal Time Coordinate) — это практически то же самое, что GMT (Greenwich Mean Time), а принимающий компьютер сам корректирует его для своей временной зоны, летнего времени и т. д. Протокол NTP присваивает каждому компьютеру один из шестнадцати уровней доступа. Компьютер первого уровня имеет доступ к очень точному сигналу времени. Компьютеры более высоких уровней при обращении к серверу времени имеют меньший приоритет. Для обеспечения большей надежности и отказоустойчивости клиент с низким приоритетом может обращаться сразу к нескольким серверам. В 1995 году разработана версия протокола NTP для персональных компьютеров — SNTP (Simplified NTP), она описана в RFC 1769. Кроме того, существуют еще три разных версии: NTP версии 3 (1995 год, RFC 1305), NTP версии 2 (1989 год, RFC 1119) и NTP версии 1 (1988 год, RFC 1059). Также разработаны версии протокола NTP для операционных систем Windows и UNIX.

Маршрутизация и протоколы маршрутизации

Маршрутизатор — это устройство для объединения нескольких, возможно географически удаленных друг от друга, сетей. Пример использования маршрутизаторов показан на рис. 3-10.

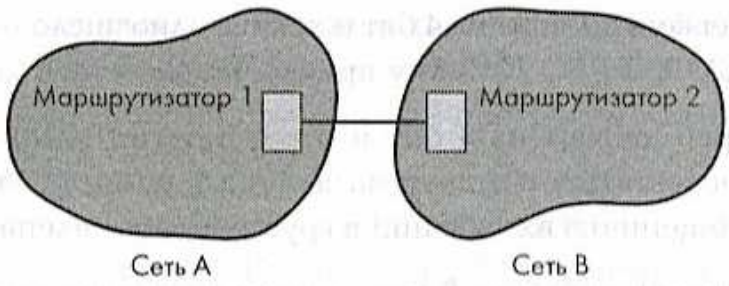


Рис. 3-10. *Пример использования маршрутизаторов*

В этом примере пара маршрутизаторов M1/M2 соединяет две географически удаленные локальные сети. Обратите внимание — маршрутизаторы всегда используются парами. Но это не означает, что в сети обязательно должно быть четное число маршрутизаторов. На рис. 3-11 показана сеть с тремя маршрутизаторами.

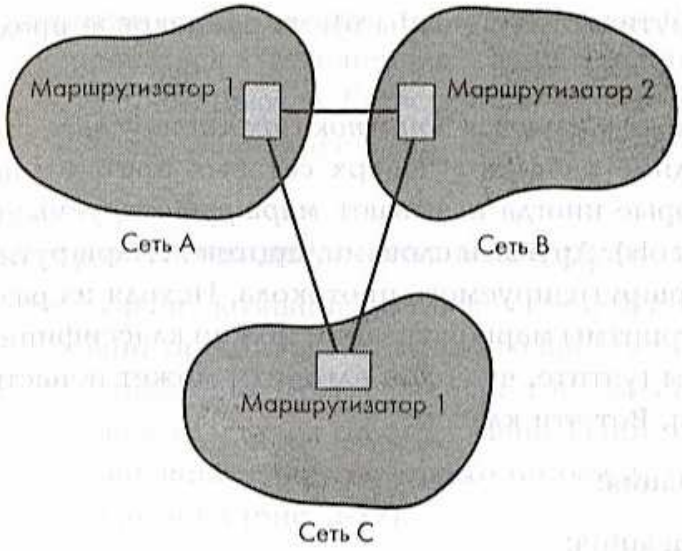


Рис. 3-11. *Сеть с тремя маршрутизаторами*

Маршрутизаторы объединяют сети разными способами. Линия связи может быть как телефонной, так и выделенной, например xDSL, ATM, Frame relay, ISDN, X.25 или любого другого типа. Маршрутизатор работает на сетевом уровне (уровень 3) модели ISO/OSI и выполняет следующие действия:

- проверяет целостность пакетов — вычисляет контрольные суммы и обновляет заголовки пакетов (например, значение счетчика переходов — количества маршрутизаторов, переславших пакет);
- обращается к таблице маршрутизации, чтобы определить дальнейший маршрут пакета. Если адрес получателя уже указан в таблице маршрутизации, то можно быстро определить, в какой порт необходимо переслать пакет;
- ставит пакет в очередь на отправку;
- собственно отправляет пакет;
- обменивается маршрутной информацией с другими маршрутизаторами.

Для обновления таблицы маршрутизации, чтобы она соответствовала текущей топологии сети, необходимы алгоритмы маршрутизации. Алгоритм маршрутизации должен быть простым, быстрым, легко реализуемым, надежным (то есть исключать ошибки при пересылке пакета следующему адресату) и, наконец, гибким к изменениям сети. Естественно, необходимо, чтобы таблицы маршрутизации обновлялись регулярно

и синхронно — все маршрутизаторы должны иметь одинаковое представление о сети.

Алгоритмы маршрутизации реализуются в *протоколах маршрутизации* (routing protocols). Последние работают поверх сетевых протоколов, например IP или IPX, которые иногда называют *маршрутизируемыми протоколами* (routed protocols). Другими словами, протокол маршрутизации работает поверх маршрутизируемого протокола. Исходя из разных критериев оценки, алгоритмы маршрутизации можно классифицировать следующим образом (учтите, что один алгоритм может попасть сразу в несколько классов). Вот эти классы:

- статическая маршрутизация;
- динамическая маршрутизация;
- внутридоменная маршрутизация;
- междоменная маршрутизация;
- одноуровневая маршрутизация;
- иерархическая маршрутизация;
- централизованная маршрутизация;
- распределенная маршрутизация;
- однопутевая маршрутизация;
- многопутевая маршрутизация;
- маршрутизация хостом;
- маршрутизация маршрутизатором;
- канальная маршрутизация (маршрутизация по состоянию канала);
- векторная маршрутизация;
- принудительная маршрутизация;

Все они описаны в последующих разделах.

Статическая и динамическая маршрутизации

При *статической маршрутизации* (static routing) таблица маршрутизации создается при настройке маршрутизатора и не изменяется динамически. Время от времени администратор сети может корректировать ее вручную. Очевидно, статическая маршрутизация не в состоянии оперативно следовать всем изменениям сети, поэтому ее редко используют.

Алгоритмы *динамической маршрутизации* (dynamic routing) способны адаптироваться к изменениям конфигурации сети, поэтому они очень широко применяются. Естественно, разные алгоритмы маршрутизации по-разному анализируют данные и обмениваются маршрутной информацией.

Внутри- и междоменная маршрутизации

Сети больших компаний, а также клиенты поставщиков услуг Интернета обычно организованы в небольшие домены. При этом некоторые маршрутизаторы играют роль внутридоменных и знают только о других маршрутизаторах своего домена. Один маршрутизатор служит «окном во внешний мир» — только он осведомлен о маршрутизации вне данного домена (рис. 3-12).

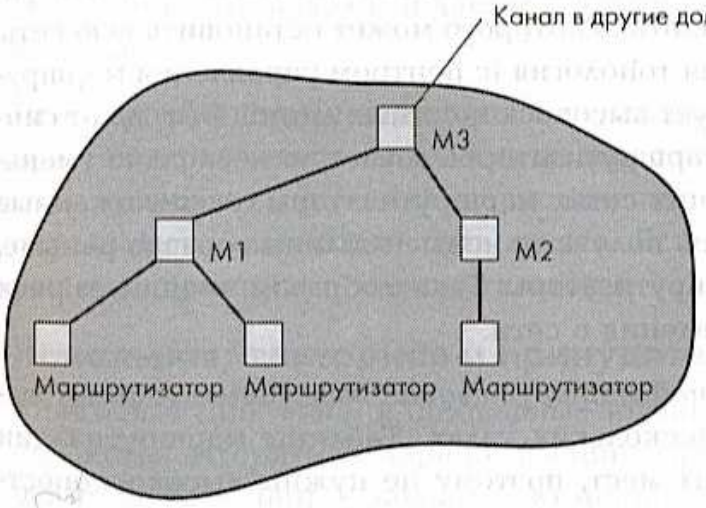


Рис. 3-12. *Внутри- и междоменная маршрутизации*

На этом рисунке маршрутизаторы M1 и M2 — *внутридоменные* (intradomain), а M3 — *междоменный* (interdomain). Для внешнего мира все три поддомена, обслуживаемые M1, M2 и M3, выглядят, как один домен. Все сообщения во внешний мир и обратно передаются через маршрутизатор M3.

Одноуровневая и иерархическая маршрутизации

При *одноуровневой маршрутизации* (flat routing) все маршрутизаторы равноправны. При *иерархической маршрутизации* (hierarchical routing) часть маршрутизаторов функционируют на равных, но доступ к другим узлам осуществляется только через один маршрутизатор более высокого уровня. Фактически, маршрутизаторы верхних уровней иерархии формируют межсетевые магистрали. Таким образом, внутридоменная

маршрутизация является одноуровневой, а междоменная — иерархической, поскольку пакеты, направленные в другие домены, проходят через один магистральный маршрутизатор. Обратимся снова к рис. 3-12, пакет, отправленный из поддомена, обслуживаемого M1 или M2, и покидающий домен через M3, маршрутизируется иерархически.

Централизованная и распределенная маршрутизации

При *централизованной маршрутизации* (centralized routing) все таблицы маршрутизации составляются в одном месте, обычно именуемом *центром управления маршрутизацией* (routing control center), и распределяются среди маршрутизаторов. Эта схема выгодна тем, что все маршрутизаторы имеют общее представление о сети, поскольку их таблицы маршрутизации одинаковы, кроме того, вычислительные ресурсы сосредоточены в одном месте. Слабое место этой схемы — центр управления маршрутизацией, отказ которого может остановить всю сеть. Кроме того, звездообразная топология (с центром управления маршрутизацией в середине) требует высокоскоростных линий. Выгода от синхронизации таблиц всех маршрутизаторов зачастую несколько уменьшается, оттого что в больших сетях маршрутизаторы, расположенные ближе к центру управления, получают новые таблицы гораздо раньше, чем сильно удаленные маршрутизаторы. Таким образом, возникает риск несогласованного представления о сети.

В алгоритмах *распределенной маршрутизации* (distributed routing) вычисления проводятся на нескольких узлах. Таблицы маршрутизации обновляются из нескольких мест, поэтому не нужны высокоскоростные линии, как при централизованной маршрутизации. Кроме того, распределенная маршрутизация более надежна, но требует больше времени на то, чтобы все узлы получили одинаковое представление о сети. Иногда это приводит к ошибкам маршрутизации.

Одно- и многопутевая маршрутизации

При *однопутевой маршрутизации* (single path routing) каждому адресату соответствует только один путь. За счет этого сокращается размер таблицы маршрутизации. В алгоритмах *многопутевой маршрутизации* (multipath routing) для каждого адресата вычисляется несколько путей. Это позволяет оптимально использовать емкость канала связи и повысить общую пропускную способность. Дополнительно обеспечивается некоторая отказоустойчивость сети. Недостаток многопутевой схемы в том, что таблицы маршрутизации занимают больший объем, а сами алгоритмы становятся сложнее.

Маршрутизация хостом или маршрутизатором

При *маршрутизации хостом* (host-intelligent routing) в каждом пакете кроме адреса получателя указан полный список узлов, составляющих маршрут пакета. Они перечислены в порядке прохождения пакета. При этом маршрутизатор служит просто приемно-передающим устройством. Эта схема также известна как *маршрутизация источника* (source routing). Сначала хост генерирует специальный пакет и отправляет его по всем известным ему маршрутам. Каждый маршрутизатор, приняв такой пакет, добавляет в него запись о себе и рассылает по всем маршрутам, о которых знает сам. В такой схеме адресат иногда получает несколько пакетов, при этом все они отсылаются обратно отправителю, который изучает пути их следования и выбирает оптимальный. Это значимое преимущество. Недостаток в том, что механизм определения путей наводняет сеть пакетами данных. Такая маршрутизация используется нечасто.

При *маршрутизации маршрутизатором* (router-intelligent routing) хост просто отправляет пакет данных на адрес получателя. Маршрутизаторы сами проводят вычисления (периодически, но не для каждого пакета) и определяют дальнейший путь пакета. Естественно, в такой схеме все решения принимают маршрутизаторы.

Канальная, векторная и принудительная маршрутизации

Канальные (link state) и *векторные* (distance vector) — два основных семейства алгоритмов маршрутизации. *Принудительная маршрутизация* (policy routing) — подвид векторной.

В первой схеме, которую еще называют *маршрутизацией по состоянию канала*, маршрутизатор широковещательно рассылает список узлов, к которым подключен. Этот список отражает *состояние канала* (link state). При такой схеме маршрутизаторы отправляют маленькие объемы информации множеству узлов. Данные передаются по мере надобности (при изменении состояния канала). Для канальной маршрутизации требуется больше вычислений, нежели для векторной. Но при этом гораздо реже возникают циклические маршруты. К канальной маршрутизации относятся следующие алгоритмы:

- ISO Intermediate System to Intermediate System;
- Netware Link Services Protocol;
- Open Shortest Path First (OSPF).

Маршрутизатор, использующий векторный алгоритм, рассылает полную таблицу маршрутизации, но только своим соседям; то есть много информации ограниченному числу узлов. Суть векторной маршрутизации в минимизации числа узлов, проходимых пакетом. Ее стоит применять в маленьких сетях с небольшим числом надежных каналов. Она проще и требует меньше памяти, чем канальная маршрутизация. При настройке векторного маршрутизатора необходимо указать всех его соседей и «стоимость» каналов к каждому из них (в терминах времени и ресурсов).

Векторные алгоритмы, также называемые алгоритмами Беллмана-Форда (Bellman-Ford) или Форда-Фалкерсона (Ford-Fulkerson), сходятся медленнее, чем протоколы канальной маршрутизации. В данном контексте термин «сходиться» означает приведение таблиц маршрутизации всех узлов к единому согласованному виду, полностью отражающему реальное состояние сети. Случай самой медленной сходимости известен как «*проблема ухода в бесконечность*» (count-to-infinity problem). Рассмотрим ситуацию, проиллюстрированную на рис. 3-13. Пусть стоимость каждого канала равна 1. Узел Y видит, что может достичь X, затратив 1. Узел Z решает, что он сможет достичь X, затратив 2 (1 — до Y и еще 1 — от Y к X). Предположим, что канал между Y и X вышел из строя. Тогда Y решает посылать пакеты, адресованные для X, через Z (поскольку Z сообщил, что имеет канал к X стоимостью 2). Узел Y сообщает, что может достичь X, затратив 3 (1 — до Z, и 2 — от Z к X). Тогда Z сообщает, что достигнет X, затратив 4. Этот процесс продолжается до тех пор, пока оба маршрутизатора не достигнут максимально возможного значения стоимости пути.

Рис. 3-13. *Простая сеть из трех узлов и двух маршрутизаторов*

Одно из возможных решений этой проблемы называется «правилом раздельного информирования» (split horizon rule): стоимость маршрута не должна сообщаться в канал, составляющий его часть. В нашем примере узел Z не должен сообщать узлу Y стоимость пути до узла X (то есть 2). Однако в сетях с достаточно сложной топологией «проблема ухода в бесконечность» иногда возникает даже при использовании этого правила. В качестве примера рассмотрим сеть на рис. 3-14, содержащую не два, а три маршрутизатора.

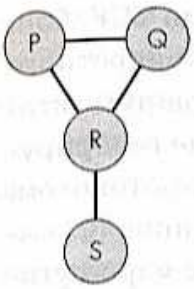


Рис. 3-14. *Сеть с тремя маршрутизаторами*

К векторным протоколам маршрутизации относятся:

- Routing Information Protocol (RIP);
- AppleTalk Routing Table Management Protocol (RTMP);
- Interior Gateway Routing Protocol (IGRP).

Протоколы *принудительной маршрутизации* не что иное, как модификации векторных. Эти алгоритмы при выборе маршрута помимо физических факторов, например числа переходов и стоимости, учитывают и политические, например деловые соглашения с другими компаниями. Такие алгоритмы используются достаточно часто.

Протокол BGP

Протокол маршрутизации BGP (Border Gateway Protocol) является динамическим, междоменным, распределенным, одноуровневым, многопутевым и управляется маршрутизатором. Он разработан с целью устранить недостатки протокола EGP (Exterior Gateway Protocol), считается его приемником и уже скоро полностью вытеснит его с просторов Интернета. В BGP предусмотрено выявление кольцевых маршрутов и использование при принятии решений различных метрик (числа переходов, пропускной способности канала и т. д.).

Изначально маршрутизаторы, работающие по протоколу BGP, обмениваются полными таблицами маршрутизации. Далее, по мере их изменения, рассылаются обновления — последовательные и отражающие только изменения. В таблицах протокола BGP возможно несколько маршрутов к одному месту назначения, но другим маршрутизаторам сообщаются только оптимальные. Естественно, для выбора таких маршрутов необходима какая-то метрика. В BGP она предусмотрена и представляет собой просто число, которое назначено администратором сети. При этом администратор должен сам учитывать такие факторы, как число переходов, скорость канала и его стабильность.

Обновления таблиц BGP рассылаются при помощи протокола TCP. Они содержат информацию о том, какие домены достижимы с конкретного узла. Существует способ выявить сбои в работе узлов и маршрутизаторов посредством сообщений «еще жив» (keep-alive), которые генерируются примерно каждые 30 секунд. Иногда BGP называют протоколом, поддерживающим Бесклассовую междоменную маршрутизацию (Classless Interdomain Routing, CIDR). Это означает, что в таблицах маршрутизации используются 32-разрядные IP-адреса и маски подсетей, значения которых при выборе маршрутов трактуются как «сплошные».

Сейчас реализована новая версия протокола BGP — BGP-4. Помимо других возможностей, в ней можно применять маски подсетей произвольной длины.

Протокол EGP

Протокол маршрутизации EGP (Exterior Gateway Protocol) — динамический, междоменный, распределенный, одноуровневый, однопутевой и управляется маршрутизатором. Он широко используется в Интернете. Информация о достижимости узла передается посредством обновлений (updates), которые содержат строки типа «куда можно попасть из узла X» или «отсюда можно попасть туда-то», то есть какие маршрутизаторы и хосты достижимы от каких маршрутизаторов. Обновления регулярно рассылаются всем соседним узлам, причем каждое содержит информацию о соседях маршрутизатора, пославшего сообщение. Маршрутизаторы собирают эту информацию и из нее составляют свои таблицы.

Протокол IGRP

Протокол маршрутизации IGRP (Interior Gateway Routing Protocol) является динамическим, внутридоменным, распределенным, одноуровневым, многопутевым и векторным. Он разработан компанией Cisco Systems для применения в сложных сетях. При выборе маршрутов учитывается множество факторов, например пропускная способность сети, ее надежность и загруженность, а также размеры сообщений. Администратор сети назначает вес каждого параметра для принятия решения, либо протокол работает с параметрами по умолчанию. Одна из возможностей протокола IGRP — «правило раздельного информирования» (см. выше).

Протокол OSPF

Протокол маршрутизации OSPF (Open Shortest Path First) является динамическим, внутридоменным, распределенным, иерархическим, мно-

гопутевым и канальным. При изменении таблицы состояния канала OSPF-маршрутизаторы обмениваются обновлениями. Для выявления сбоев маршрутизатор рассылает сообщения «еще жив». Протокол OSPF поддерживает запросы QoS (quality of service), когда приложение сообщает о срочности некоторых данных. В этом случае OSPF по своему усмотрению использует имеющиеся каналы, чтобы как можно быстрее отправить данные.

Протокол RIP

Протокол маршрутизации RIP (Routing Information Protocol) является динамическим, внутридоменным, распределенным, одноуровневым, однопутевым и векторным. Разработанный компанией Xerox, он был одобрен другими производителями — 3Com, Apple, Banyan, Novell и Ungermann Bass. Этот протокол идеально подходит для малых и средних сетей, но не для крупных, поскольку максимальное число переходов ограничено числом 16. Протокол RIP не может оперативно учитывать изменения свойств сети (например, время задержки или загруженность каналов). Намечается тенденция к замене RIP более современным протоколом маршрутизации.

Windows Sockets

Стандарт Windows Sockets более известен как *Winsock*. Последняя его версия — вторая — увидела свет в начале 1993 года. Сам Winsock напоминает Berkley Sockets для UNIX и служит основой для написания сетевых приложений.

Основная идея Winsock — позволить разрабатывать сетевые приложения, способные работать с любым стеком TCP/IP. Это актуально, поскольку интерфейсы разных реализаций TCP/IP иногда отличаются. В первой версии Winsock (версия 1.1) поддерживался только транспорт TCP/IP. Winsock версии 2 способен работать и с другими транспортными протоколами, например ATM или IPX/SPX (Internet Packet Exchange/Sequenced Package Exchange), причем разные приложения могут их использовать одновременно. В Winsock версии 2 предусмотрена обратная совместимость с предыдущей версией, то есть приложениям для Winsock 1.1 не нужна модификация, чтобы работать с Winsock версии 2. Архитектура Winsock изображена на рис. 3-15.

Компании Microsoft и Intel поставляют Winsock в виде динамической библиотеки (dynamic-link library, DLL). Реально для разных версий Windows существуют разные версии Winsock DLL, имеющие разные названия. Библиотека Winsock DLL предоставляет сразу два разных API:

«сверху» — Winsock 2.0 API, «снизу» — Winsock Service. Первый предназначен для приложений, а второй — для разработчиков системного ПО, желающих расширить Winsock дополнительными транспортными механизмами.

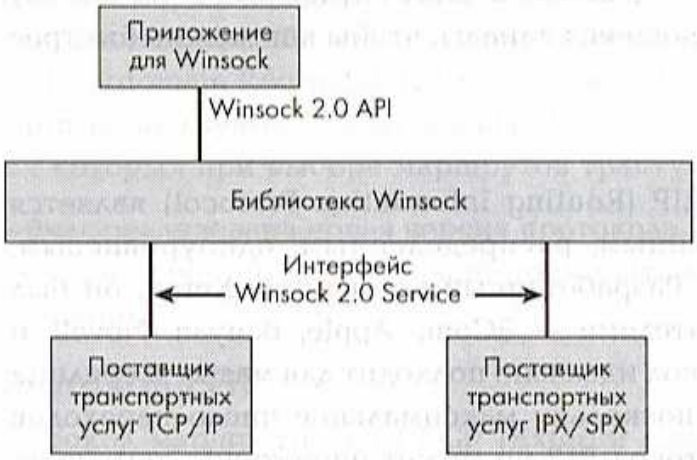


Рис. 3-15. *Архитектура Winsock*

WinInet

WinInet (или Win32 Internet API) — это программный интерфейс и библиотека для операционной системы Microsoft Windows, позволяющие быстро и просто разрабатывать приложения для Интернета, использующие HTTP (Hypertext Transfer Protocol), FTP (File Transfer Protocol) и Gopher. В WinInet используется последовательная модель, аналогичная чтению и записи файлов. Применяя WinInet, можно разрабатывать приложения без знания Winsock, TCP/IP или протоколов верхних уровней, например FTP и HTTP. Также WinInet поддерживает соединения с серверами Интернета, защищенные посредством протоколов SSL (Secure Sockets Layer) или Kerberos. (Подробнее о SSL и Kerberos — в главе 5.)

Протоколы маршрутизации для мостов

Когда протоколы сетевого уровня не используются, то локальные сети обычно соединяются через мосты. Существует два типа протоколов маршрутизации для мостов: *прозрачная* (transparent bridging) и *маршрутизация источника* (source routing bridging). Учтите, что некоторые мосты способны поддерживать оба протокола маршрутизации; такое сочетание называется *прозрачной маршрутизацией источника* (source routing transparent, SRT). Обычно SRT-мосты действуют как прозрачные мосты, но при появлении пакета с маршрутной информацией от источника, мост это обнаружит и соответствующим образом обработает.

Прозрачные мосты

Прозрачный мост получил свое название потому, что он невидим для всех взаимодействующих узлов. Он просматривает все пакеты, прибывающие в оба его порта, и составляет базу данных об адресах узлов, расположенных по разные его стороны. Приняв пакет, мост смотрит адрес получателя и игнорирует его, если получатель находится на той же стороне, откуда пришел этот пакет. Если получатель находится по другую сторону от моста, то пакет пересылается туда. Если мост не знает, с какой стороны находится получатель, то пакет в любом случае будет переправлен на другую сторону.

Этот механизм хорошо работает до тех пор, пока в сети, к которой подключен мост, нет кольцевых маршрутов. Такие маршруты губительны для протоколов канального уровня, поскольку в их пакетах не предусмотрено поле TTL (время жизни), которое есть на сетевом уровне.

Алгоритм *дерева обхода* (spanning tree) позволяет использовать мосты в сетях произвольной топологии (в том числе с кольцевыми маршрутами), не опасаясь, что из-за колец сеть наводнится множественными копиями одного пакета. Алгоритм дерева обхода в основном аналогичен векторной маршрутизации, но для него нет проблемы «ухода в бесконечность». Дерево охватывает все узлы сети. В качестве *корневого* (root bridge) выбирается мост с минимальным идентификатором. Это происходит динамически, путем голосования. Затем каждый мост вычисляет стоимость маршрута до корневого моста. При этом учитывается скорость передачи данных в каждом доступном канале. Этот алгоритм хорош тем, что любой мост заботится только о пути к корню дерева.

В каждом сегменте локальной сети выбирается один мост, который будет рассылать управляющие сообщения. Этот, так называемый *выделенный мост* (designated bridge), должен иметь самую маленькую стоимость пути к корню дерева. Если несколько мостов имеют одинаковую стоимость, то выбирается мост с меньшим идентификатором. Выделенные мосты рассылают управляющие сообщения, которые содержат:

- идентификатор моста, пославшего пакет;
- идентификатор моста, который, по мнению отправителя пакета, является корнем;
- стоимость маршрута к корню.

Управляющие сообщения мостами не пересылаются в другие сегменты сети. На этом этапе каждый мост знает, какой порт использовать для

доступа к корню. Этот порт называют *выделенным* (designated port). Теперь можно построить дерево обхода. В него войдут только каналы, подключенные к выделенным мостам в выделенные порты. Пакеты, принятые из каналов, входящих в дерево обхода, пересылаются дальше, а из других каналов — игнорируются.

Учтите, что здесь опущены многие тонкости алгоритма дерева обхода (таймеры, например). Подробности вы найдете в стандарте IEEE 802.1d — официальном описании алгоритмов дерева обхода.

Мосты с маршрутизацией источника

Суть использования *мостов с маршрутизацией источника* в том, что отправитель указывает в пакете полный путь его следования. Естественно, узел-отправитель сначала определяет маршрут (и сохраняет его для последующего использования) — при помощи так называемого пакета-разведчика. Мосты рассылают этот пакет во все доступные им каналы, оставляя в нем свой «автограф». Затем получатель по своему усмотрению либо выбирает наилучший маршрут (просмотрев в пакете-разведчике список пройденных им узлов) и сообщает его отправителю, либо отсылает все пакеты-разведчики, пришедшие разными маршрутами, отправителю, предоставив ему право выбора лучшего маршрута.

Межсетевые экраны

Все больше и больше компаний подключаются к Интернету и создают собственные Web-узлы. Естественно, при этом возрастают требования к защищенности их сетей. Один из способов обеспечения надежной защиты — *межсетевые экраны* (firewall), позволяющие изолировать внутреннюю сеть и компьютеры от Интернета и ограничить доступ пользователей Интернета к внутренней информации и системам. В основном все межсетевые экраны можно разделить на три категории:

- пакетные фильтры (фильтры без памяти);
- канальные фильтры;
- фильтры прикладного уровня.

Пакетные фильтры

Пакетные фильтры (packet-level filter) блокируют или пропускают пакеты исключительно на основе их свойств (адреса и порты источника и получателя) и не запоминают прошлого состояния. Кроме того, такие фильтры могут анализировать заголовки протоколов сетевого уровня. Этот тип межсетевого экрана самый простой в реализации и обслужи-

вании и почти никак не влияет на пропускную способность сети. Но, к сожалению, уровень такой защиты невысок. Так, например, правила фильтрования можно обойти, используя вложение протоколов. Другими словами, если запрещен протокол HTTP, но разрешен Telnet, то для прохождения HTTP-трафика его можно вложить в сеанс Telnet.

Канальные фильтры

Канальный фильтр (circuit filter) находится где-то между пакетным и фильтром прикладного уровня. Он сохраняет некоторые данные о предыдущих пакетах и решение по маршрутизации очередного пакета принимает, как используя их, так и на основе содержимого пакета. Естественно, этот тип фильтров более сложный, нежели пакетный, поскольку помимо проверки самого пакета необходимо просматривать и обновлять память. В правилах фильтрования можно учитывать как адреса отправителя и получателя, так и тип обслуживания. Обычно канальные фильтры реализуются в качестве шлюзов: клиент связывается с фильтром, который от имени клиента взаимодействует с внешними серверами. Такие фильтры чаще применяют к исходящему трафику, нежели к входящему (например, Telnet).

Фильтры прикладного уровня

Фильтры прикладного уровня (application-level filter) обеспечивают высокую степень защиты, но за счет снижения производительности и увеличения сложности. Такие фильтры реализуются как выделенные межсетевые экраны. Сам сервер приложения находится в частной сети за межсетевым экраном. Внешние клиенты подключаются к последнему, который ведет себя как сервер приложения. Фактически, клиент не может обнаружить, что взаимодействует с межсетевым экраном, который в этом случае называют *прикладным прокси-сервером* (proxy application server). С другой стороны, межсетевой экран «притворяется» клиентом и пересылает принятые клиентские запросы настоящему серверу приложения. Но прежде межсетевой экран на основе заданных правил решает вопрос о допустимости такого запроса и наличии у клиента права на запрашиваемые действия. Из этого следует, что межсетевой экран имеет полное представление об используемых приложениях и протоколах. Единственный недостаток прикладного прокси-сервера — снижение общей производительности.

Протоколы IP Next Generation и IPv6

Текущая версия протокола IP (IPv4) не полностью удовлетворяет растущие потребности. Стремительный рост сети Интернет в последние годы

только обострил эту проблему. Ниже перечислены основные недостатки IPv4.

- Для IP-адресов выделено только 32 бита. В мире, где почти все корпоративные и домашние ПК подключены к Интернету и где каждый тостер, телевизор, микроволновая печь и даже электролампочка являются потенциальными кандидатами на IP-адрес, IP-адресов может просто не хватить на всех.
- Несовершенство системы адресации. Адреса класса A обеспечивают около 16 млн. узлов, класса B — около 64 000 узлов, класса C — до 254 узлов. Узлом может быть любой компьютер, маршрутизатор или сетевое устройство типа принтера. Наиболее популярны адреса класса B, поскольку в действительности мало компаний, имеющих 16 млн. узлов, но подавляющему большинству требуется больше 254 узлов. Однако возможно только 16 384 сетей класса B.
- Таблицы маршрутизации центральных маршрутизаторов Интернета сильно разрастаются. Это значительно ухудшает ситуацию: либо увеличится время обработки каждого пакета, либо для обслуживания таблицы маршрутизации появятся отдельные процессоры, что сильно удорожит сами маршрутизаторы. Оба варианта малопривлекательны.
- Он не годится для приложений, требующих услуг высокого качества (QoS), например для мультимедийных приложений, которые за короткое время передают огромные объемы данных.
- Он не годится для новых приложений, нуждающихся в более высоком уровне безопасности и аутентификации.

Для решения всех перечисленных проблем группа IETF в начале 90-х годов запустила новый проект. Первым конкретным результатом стала публикация в 1995 году RFC 1752. Здесь определены требования к так называемому протоколу IP Next Generation (IPng), а в некоторых дополнительных документах описаны формат заголовка, маршрутизация, адресация и вопросы безопасности нового протокола. Затем появилось еще много документов, например RFC 1883, и протокол был официально переименован в Internet Protocol version 6 (IPv6).

Вот наиболее существенные преимущества IPv6.

- Значительно увеличено адресное пространство. Адреса в IPv6 имеют длину 128 бит, что примерно в 2^{96} раз больше, чем в IPv4.
- Возрасла скорость обработки IPv6-заголовков маршрутизаторами. Несмотря на то, что заголовок стал больше (40 байт в IPv6 против

минимального размера 20 байт в IPv4), он имеет меньше полей (8 против 12). Кроме того, большинство параметров, не обрабатываемых маршрутизаторами, вынесены в дополнительные заголовки.

- Реализованы механизмы аутентификации, проверки подлинности и целостности IP-пакетов.
- Появился способ пометки IP-пакетов, требующих специальной обработки, что обычно используется для мультимедиа и приложений реального времени.

Адресация в IPv6

Адреса в IPv6 имеют длину 128 бит. IP-адреса назначаются сетевым интерфейсам узлов, а не самим узлам. Каждый узел может иметь несколько интерфейсов и, следовательно, несколько IP-адресов. Кроме того, один интерфейс тоже может иметь несколько IP-адресов.

В IPv6 существует три вида IP-адресов: *индивидуальные* (unicast), *групповые* (multicast) и *коллективные* (anycast); все они описаны ниже. Также IPv6 предоставляет несколько механизмов динамического выделения IP-адресов. Первый во многом похож на протокол DHCP (Dynamic Host Configuration Protocol), описанный далее в этой главе. Здесь сервер назначает адреса из заранее выделенного диапазона. Этот способ иногда называют *настройкой с памятью* (configuration with state). Сервер запоминает прошлых владельцев адресов и предоставляет адреса на определенное время. Если адресов достаточно, то клиент получит тот же самый адрес, что и при последнем запросе.

Второй механизм динамического получения IP-адреса — *настройка без памяти* (stateless). Клиент сам создает себе IP-адрес путем конкатенации адреса для локального канала и локального идентификатора (например, аппаратного адреса Ethernet-адаптера). Компьютер, использующий протокол IPv6, не обязан распознавать типы IP-адресов. Это дело маршрутизатора — полностью понимать и соответственно обрабатывать различные типы адресов.

Индивидуальные адреса

На рис. 3-16 показаны разные типы индивидуальных адресов протокола IPv6. Каждый адрес состоит из 128 бит.

Адреса первого типа выделены поставщикам услуг Интернета, которые в свою очередь назначают эти уникальные (в глобальном смысле) адреса своим подписчикам.

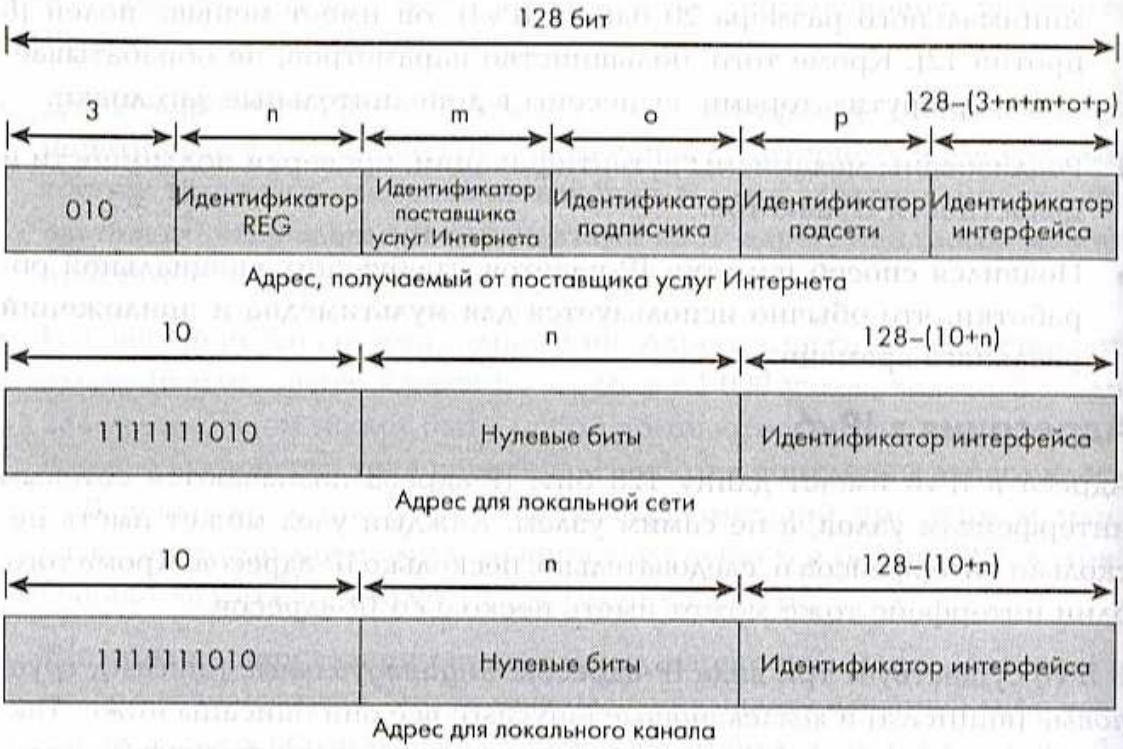


Рис. 3-16. *Индивидуальные адреса в IPv6*

Адреса второго типа предназначены для использования в корпоративных интрасетях, не подключенных к Интернету. Если организация решит подключить свою интрасеть к Интернету, то эти адреса легко обновить до уникальных. Для этого администратору не придется подходить к каждому устройству и вручную менять его IP-адрес, как было бы при использовании IPv4.

Последний тип индивидуальных адресов применяется в основном компьютерами, связанными через один канал, например линию ISDN.

Групповые адреса

В протоколе IPv6 предусмотрено гораздо больше групповых адресов, чем в IPv4. На рис. 3-17 показан групповой IP-адрес длиной 128 бит.

Рис. 3-17. *Групповой адрес в IPv6*

Первые 8 бит группового адреса установлены в 1. Поле **Флаги** (Flags) состоит из 4 бит, из которых три старших зарезервированы и равны 0. Последний бит при обращении к постоянной группе, признанной во всем Интернете, устанавливается равным 0. Это означает группу, которая всем известна и широко распространена.

Поле **Область видимости** (Scope) состоит из 4 бит, характеризующих природу группового адреса. Допустимые значения показаны в табл. 3-5.

Таблица 3-5. *Значения поля* ***Область видимости***

Значение	Описание
0	Зарезервировано
1	Область локального узла
2	Область локального канала
3–4	Не назначено
5	Область локальной сети
6–7	Не назначено
8	Область локальной организации
9–E	Не назначено
F	Зарезервировано

Коллективные адреса

В протоколе IPv6 используется понятие *коллективный адрес* (anycast). Он аналогичен групповому в том смысле, что адресует больше одного сетевого интерфейса (которые физически находятся на разных узлах) и выделяется из диапазона доступных групповых адресов. Отличие лишь в том, что пакет, посланный по коллективному адресу, доходит только до одного из членов группы. Выбор, кому направить пакет, остается за промежуточным пересылающим узлом.

Учтите, использование группового адреса в качестве коллективного должно документироваться либо в файле настроек, либо другим способом.

Заголовок в IPv6

В отличие от IPv4, заголовки в IPv6 имеют фиксированную длину (рис. 3-18).

Поле **Версия** (Version) состоит из 4 бит и имеет значение 6, отражая номер версии протокола IP.

Поле **Приоритет** (Priority) состоит из 4 бит и позволяет отправителю указать приоритет IP-пакета относительно других IP-пакетов того же

источника. Значения от 0 до 7 соответствуют приложениям, которые могут управлять загруженностью канала в зависимости от условий сети. Рекомендуемые значения показаны в табл. 3-6.

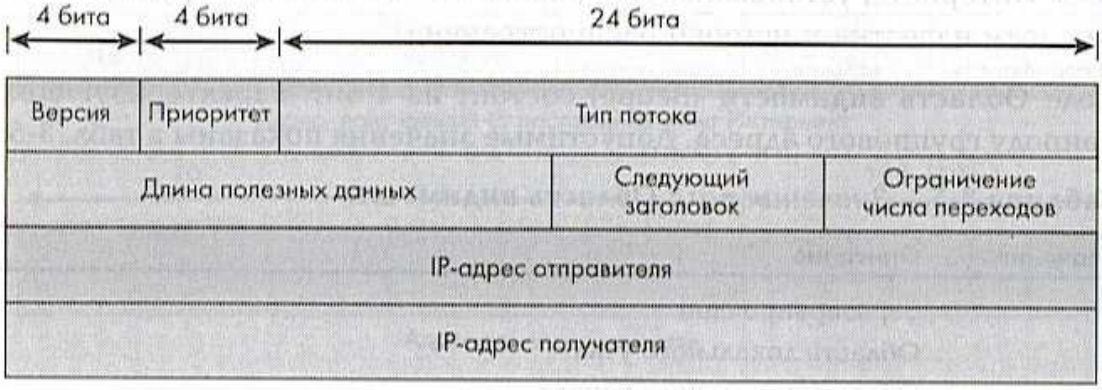


Рис. 3-18. *Заголовок в IPv6*

Таблица 3-6. *Значения поля* ***Приоритет***

Значение	Описание
0	Не описан
1	Время простоя или фоновый трафик (например, сетевые новости)
2	Необслуживаемый трафик (например, электронная почта)
3	Зарезервировано
4	Обслуживаемый трафик (например, HTTP или FTP)
5	Зарезервировано
6	Интерактивный трафик (например, Telnet)
7	Управляющий трафик (например, информация о маршрутизации)

Значения от 8 до 15 используются для трафика, не способного адаптироваться к условиям сети, например, если приложение вообще не может обрабатывать переполнения буфера. Меньшие значения рекомендуются для трафика, утеря которого породит меньше проблем, например, для видео данных следует указывать приоритет 8, а для аудио — 15, полагая, что утеря аудиоданных, как правило, происходит только после утери видео.

Поле **Тип потока** (Flow label) состоит из 24 бит и применяется для запроса специальных функций маршрутизаторов. Название этого поля произошло оттого, что под потоком понимается последовательность пакетов от отправителя к получателю. Предполагается, что это поле будут использовать специализированные приложения, например мультимедиа и реального времени.

Поле **Длина полезных данных** (Payload length) состоит из 16 бит и содержит длину IP-пакета, следующего за заголовком. Здесь учтены также длины всех дополнительных заголовков.

Поле **Следующий заголовок** (Next header) состоит из 8 бит и отражает тип заголовка, следующего непосредственно за IPv6-заголовком. То есть это поле отражает тип протокола верхнего уровня, вложенного в IPv6-датаграмму. Значения те же, что для протокола IPv4 (см. RFC 1700).

Поле **Ограничение числа переходов** (Hop limit) состоит из 8 бит и показывает, сколько еще переходов может сделать IP-пакет. Это поле аналогично полю **TTL** в IPv4-заголовке.

Последние два поля — это 128-разрядные IP-адреса отправителя и получателя.

Дополнительные заголовки в IPv6

В IPv6 можно поместить несколько дополнительных заголовков между IPv6-заголовком и данными (рис. 3-19). Такая схема позволяет маршрутизаторам обрабатывать заголовок пакета достаточно быстро, особенно если нет никаких его дополнений. Кроме того, эта схема удобна для будущих модернизаций.

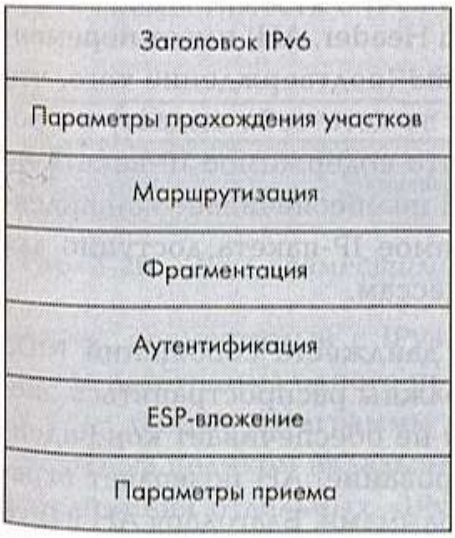


Рис. 3-19. *Необязательные дополнительные заголовки IPv6*

Заголовок **Параметры прохождения участков** (Hop by Hop Options Header) имеет переменную длину и должен просматриваться всеми узлами (в том числе и маршрутизаторами) на пути следования пакета. В настоящее время этот заголовок используется только для передачи па-

кетов, размер полезной нагрузки которых превышает 64 Кб (так называемые Jumbo Payload).

Заголовок **Маршрутизация** (Routing Header) также имеет переменную длину и обеспечивает возможности, аналогичные маршрутизации источника в протоколе IPv4 (см. ранее в этой главе). Здесь содержится список узлов, которые должен пройти пакет на пути к получателю.

В IPv4 фрагментацию и сборку пакетов может выполнять любой промежуточный узел. В IPv6 фрагментирует пакет только отправитель, а собирает — только получатель. В 8-разрядном заголовке **Фрагментация** (Fragment Header) хранится информация, необходимая для обратной сборки пакета.

Рабочая группа IP Security (в составе IETF) определила два дополнительных заголовка для обеспечения безопасности. Их можно использовать как с IPv6, так и с IPv4-заголовками. (Некоторые узлы, использующие IPv4, не поддерживают эти механизмы.) Сочетание двух дополнительных заголовков — **Аутентификация** и **ESP-вложение** — известно как IPSec. Это самостоятельный протокол, имеющий реализацию как для IPv6, так и для IPv4.

Аутентификация

Заголовок **Аутентификация** (Authentication Header, AH) имеет переменную длину и обеспечивает аутентификацию (подтверждение того, что отправитель IP-пакета действительно тот, за кого себя выдает) и проверку целостности, то есть гарантирует, что содержимое IP-пакета не изменилось во время пути. Учтите, что AH не обеспечивает конфиденциальность информации, то есть содержимое IP-пакета доступно для прочтения любым приложениям или процессам.

Для создания AH применяется алгоритм дайджеста сообщений MD5 (подробнее описан в главе 5). На AH не должны распространяться экспортные ограничения США, поскольку он не обеспечивает конфиденциальность данных и не использует шифрование. AH позволяет оградить хост от «атак» непрошеными IP-датаграммами. Благодаря AH в IPv6 появился механизм, посредством которого хост станет принимать данные от клиента только после того, как последний предоставит «удостоверение личности».

ESP-вложение

Заголовок **ESP-вложение** (Encapsulated Security Payload, ESP) в основном предназначен для обеспечения конфиденциальности информации:

данные не сможет прочесть никто, кроме адресата. Кроме того, в зависимости от используемого алгоритма, ESP-вложение способно обеспечить как аутентификацию, так и целостность данных. Вероятно, на ESP-вложения будут распространены экспортные ограничения США. Для создания ESP-вложения можно использовать разные криптографические алгоритмы, но, по соображениям совместимости, предпочтительнее DES в режиме CBC. Шифруются данные и часть ESP-вложения. Существует два режима ESP-вложения: *туннельный* (tunneling) — шифруется вся IP-датаграмма — и *транспортный* (transport) — шифруются данные, начиная с заголовка протокола транспортного уровня, например TCP.

Параметры приема

Заголовок **Параметры приема** (Destination Options) является необязательным. Если он есть, то предназначен только для адресата. Маршрутизаторам, обрабатывающим IPv6-пакет, не вменяется даже его просмотр.

Совместимость IPv6

При разработке IPv6 ставилась задача обеспечить плавный переход с IPv4 на IPv6. На одном хосте или маршрутизаторе можно использовать IPv4 и IPv6 одновременно. Способ адресации в IPv6 позволяет обеспечить совместимость с IPv4 (рис. 3-20).

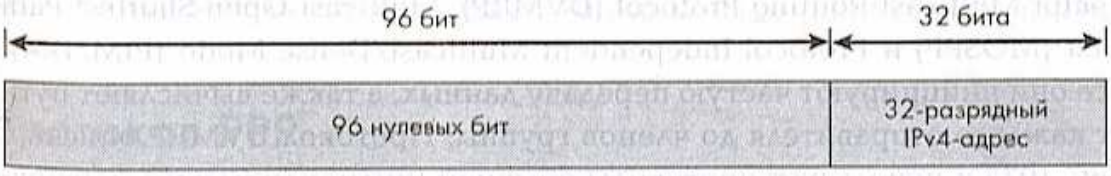


Рис. 3-20. *IPv4-совместимый IPv6-адрес*

Адрес, совместимый с IPv4, используется для *туннелинга* (tunneling), когда датаграмма одного протокола (данные и заголовок) вкладывается в поле данных датаграммы другого протокола. В нашем случае IPv6-датаграмма целиком вкладывается в IPv4-датаграмму. Ожидается, что для соединения отдельных «IPv6-кланов» через каналы, поддерживающие только IPv4, будут применяться специальные маршрутизаторы. Каждому из них назначат уникальный IPv4-совместимый индивидуальный адрес.

Другой тип адреса называется *IPv6-адресом, отображаемым на IPv4* (IPv4-mapped IPv6 address). Адреса такого типа назначаются узлам, поддерживающим только IPv4.

Групповая передача и групповая маршрутизация

Групповая передача (multicast transmission) — это отправка данных определенного рода с одной станции для группы станций, заинтересованных в приеме этих данных. Обратите внимание, что здесь реализуется тип «один многим», который отличается от широковещания типа «один всем». Наиболее популярная реализация групповой передачи — это *групповая магистраль* (multicast backbone, Mbone), описанная в главе 7. Под покровительством IP Multicast Initiative (IPMI) для продвижения IP-технологий группового вещания объединилось множество компаний. Подробнее об IPMI можно узнать по адресу *http://www.ipmulticast.com*.

Ниже описаны два основных варианта групповой маршрутизации — *уплотненный режим* (dense mode) и *разреженный режим* (sparse mode). При изучении материалов помните, что технологии групповой маршрутизации все еще развиваются и проходят стандартизацию.

Групповая маршрутизация в уплотненном режиме

При групповой маршрутизации в уплотненном режиме предполагается компактное расположение в сети членов группы, для которой передаются данные, а также наличие каналов с достаточной пропускной способностью. Примеры протоколов уплотненного режима — Distance Vector Multicast Routing Protocol (DVMRP), Multicast Open Shortest Path First (MOSPF) и Protocol Independent Multicast/Dense Mode (PIM/DM). Все они инициируют частую передачу данных, а также вычисляют пути от каждого отправителя до членов группы. Протокол DVMRP описан в RFC 1075 и использует протоколы обычной (индивидуальной) маршрутизации. MOSPF описан в RFC 1584 и применяет протокол OSPF. Протокол PIM/DM, как следует из названия, не зависит от нижележащего протокола индивидуальной маршрутизации, но при этом передает большие объемы данных, которые иногда даже дублируются.

Групповая маршрутизация в разреженном режиме

При групповой маршрутизации в разреженном режиме предполагается, что члены группы сильно рассредоточены по сети, а емкость каналов ограничена. Напимер, протокол PIM/SM: данные передаются, как в уплотненном режиме, но маршрут не зависит от того, откуда они пришли (тогда как в уплотненном режиме трафик маршрутизируется в зависимости от источника данных). PIM/SM использует нижележащий протокол индивидуальной маршрутизации, причем любой.

Протокол SLIP

Межсетевой протокол для последовательных линий — SLIP (Serial Line Internet Protocol) — широко применялся до недавнего времени, но сейчас его почти заменил PPP. SLIP обеспечивает передачу IP-пакетов через последовательные линии типа «точка-точка». Протокол SLIP можно рассматривать как некий аналог Ethernet или TokenRing. Как в локальной сети IP-датаграммы вкладываются в кадры Ethernet, так на последовательной линии — в кадры SLIP (или PPP). Но в кадры SLIP можно вкладывать только пакеты протокола IP и никаких других, вроде DECNet или IPX (от Novell).

Конец кадра данных в протоколе SLIP обозначается специальным символом — END. Если этот символ встретится в данных пользователя, то он передается как 2 байта — символ ESC, а за ним END. В этом протоколе не предусмотрены механизмы выявления ошибок и повторной передачи. Кроме того, необходимо правильно сконфигурировать клиенты на обоих концах SLIP-соединения, поскольку протокол не предусматривает динамическую настройку.

В SLIP нет стандартного способа динамического выделения клиенту IP-адреса. Некоторые производители создают свои (часто несовместимые друг с другом) дополнения к SLIP, устраняющие этот недостаток. С ростом популярности протокола PPP, имеющего для этого стандартные механизмы, SLIP быстро отходит на второй план (SLIP описан в RFC 1055).

Протокол PPP

Протокол PPP (Point-to-Point Protocol) предоставляет стандартные механизмы передачи данных от протоколов верхнего уровня по последовательным линиям типа «точка-точка». То есть PPP делает то же самое, что SLIP, и его тоже можно рассматривать как аналог Ethernet или TokenRing. В локальной сети IP-датаграммы вкладываются в кадры Ethernet, а на последовательной линии — в кадры PPP.

Отличие PPP от SLIP в большей функциональности. Например, в PPP помимо IP можно вкладывать другие сетевые протоколы, например DECNet, IPX или AppleTalk. Кроме того, протокол PPP более гибок: в нем предусмотрены как сама динамическая настройка канала, так и возможность ее применения обоими участниками соединения. Также он позволяет выявлять ошибки.

Протокол PPP легко расширяется для использования поверх него каких-либо других сетевых протоколов. Он предоставляет механизмы сжатия данных и динамического назначения IP-адреса удаленному узлу при помощи протокола IPCP (IP Control Protocol). В IPCP сервер телефонного доступа (dial-in server) выделяет IP-адрес из заранее определенного диапазона. По завершении сеанса удаленного доступа этот IP-адрес освобождается.

Протокол PPP включает три основных компонента:

- протокол HDLC (High level Data Link Control) — для кодирования данных перед отправкой в линию;
- протокол LCP (Link Control Protocol) — услуги по настройке, созданию и тестированию соединения; кадры LCP пересылаются до начала передачи пользовательских данных;
- протокол NCP (Network Control Protocol) — для настройки различных протоколов, которые способен переносить PPP.

Дополнительная функциональность PPP обеспечивается за счет увеличения нагрузки на сеть. Так, при вложении пакета в кадр SLIP, объем данных, подлежащих передачи в линию, увеличивается всего на 1 байт, тогда как при PPP — на целых 8 байт. Кроме того, перед началом передачи абонентам необходимо обменяться несколькими LCP- и NCP-пакетами. Однако пользователи оправдывают излишнюю нагрузку на сеть расширением возможностей.

В протоколе PPP предусмотрено целых два механизма защиты соединений. Подробно они описаны в RFC 1334. Первый — *протокол аутентификации по паролю* (Password Authentication Protocol, PAP) — сейчас все менее популярен на фоне усиления интереса к вопросам компьютерной безопасности. Основная причина этого — в PAP пароли передаются открытым текстом. Второй механизм — *протокол аутентификации по запросу/ответу* (Challenge Handshake Authentication Protocol, CHAP). Здесь сервер отсылает клиенту запрос (уникальный для каждого PPP-сеанса). На основании секретного значения, известного только клиенту и серверу, клиент вычисляет ответ на этот запрос и отправляет его серверу. Тем временем сервер проводит те же вычисления и сравнивает полученное значение с ответом, принятым от клиента. Протоколы PAP и CHAP не исключают необходимость дополнительной защиты, поскольку позволяют идентифицировать только устройство, но не пользователя.

Протокол PPP описан в RFC 1171, дополнения PAP и CHAP — в RFC 1334.

Протокол PPP позволяет двум компьютерам в некоторой степени безопасно взаимодействовать друг с другом. Он предоставляет средства опознания личности того, с кем или с чем взаимодействует компьютер. Эти средства описаны в следующих разделах.

PAP

В протоколе PAP определен простой двухэтапный метод, используя который один объект может подтвердить свою личность другому. Хост, инициирующий соединение, называют клиентом. Он посылает запрос серверу (аутентификатору), где указывает идентификатор пользователя и пароль. Тот обрабатывает их и отсылает ответ, где сообщает — удостоверена личность клиента или нет. Главный недостаток PAP в том, что идентификатор пользователя и пароль пересылаются незашифрованными. Протокол PAP подробно описан в RFC 1334.

CHAP

Протокол CHAP предоставляет более надежные, чем в PAP, средства аутентификации. Он подразделяется на три этапа и управляется аутентификатором (а не клиентом, как PAP). Клиент получает от аутентификатора запрос и при помощи однонаправленной хеш-функции MD5 вычисляет ответ, который отсылает обратно аутентификатору (о хеш-функциях и MD5 — в главе 5). При вычислении клиент применяет идентификатор пользователя, пароль, а также содержимое самого запроса. Аутентификатор тоже вычисляет предполагаемый ответ и сравнивает его со значением, принятым от клиента. Подробно протокол CHAP описан в RFC 1334.

MS-CHAP

Протокол MS-CHAP (Microsoft CHAP) очень похож на CHAP. Аутентификатор посылает запрос, а клиент по нему вычисляет ответ с помощью результата хеш-функции MD4, примененной к идентификатору и паролю пользователя. (Функция MD4 описана в главе 5.) Основное отличие MS-CHAP от CHAP в том, что аутентификатору не нужно знать идентификатор и пароль пользователя клиента в их открытом виде, что обеспечивает больший уровень безопасности.

Виртуальная частная сеть

Виртуальная частная сеть (virtual private network, VPN) позволяет двум компьютерам безопасно обмениваться данными, даже если они проходят через сеть общего пользования. Название объясняется тем, что обмен данными во многом напоминает взаимодействие в частной сети.

В VPN-сетях применяется технология, называемая *туннелинг* (tunneling) или *вложение* (encapsulation). При этом данные и их «упаковка», сформированные одним протоколом, трактуются как данные для другого протокола: именно он и его «упаковка» обеспечивают маршрутизацию и шифрование данных. Туннельные соединения можно создавать на канальном или сетевом уровнях модели ISO/OSI. Примеры туннельных протоколов: канального уровня — Point-to-Point Tunneling Protocol (PPTP), Layer 2 Forwarding (L2F) и Layer 2 Tunneling Protocol (L2TP); сетевого уровня — IPSec Tunneling mode. (Они подробно описаны в следующих разделах.) Методы, используемые этими протоколами для создания и обслуживания туннельных соединений, несколько отличаются. Но обычно данные передаются посредством датаграмм. Применение методов шифрования (если это возможно) и сжатия данных согласуется при создании туннельного соединения. В некоторых случаях клиенту назначается IP-адрес. Протоколы PPTP, L2F и L2TP для пересылки данных через соединительную сеть задействуют протокол PPP.

PPTP

Спецификация протокола PPTP разработана совместно несколькими компаниями, в том числе Microsoft, 3Com, Ascend Communications и другими. Пакеты протоколов IP, IPX или NetBEUI упаковываются в IP-пакеты. Для обмена информацией по обслуживанию туннельного соединения используется TCP, а для переноса данных — PPP. Перед отправкой данные могут быть сжаты или зашифрованы (используется RSA RC4 — см. главу 5).

Протокол PPTP может применяться, даже если его поддерживают только взаимодействующие клиент и конечный сервер. То есть от промежуточных серверов и маршрутизаторов вовсе не требуется поддержка этого протокола (рис. 3-21).

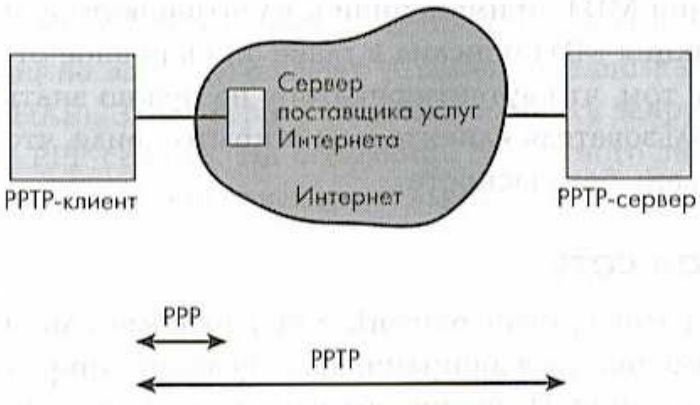


Рис. 3-21. *Ситуация 1: протокол PPTP не поддерживается промежуточным сервером*

Сначала клиент устанавливает PPP-соединение с поставщиком услуг Интернета, а после — PPTP-соединение с удаленным сервером.

Также протокол PPTP можно применять, даже если его не поддерживает сам клиент, но поддерживают сервер поставщика услуг Интернета (с которым установлено телефонное соединение) и сервер, с которым клиент желает взаимодействовать (рис. 3-22).

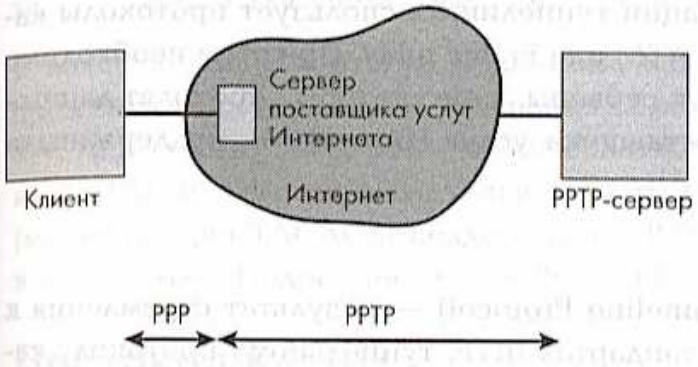


Рис. 3-22. *Ситуация 2: протокол PPTP не поддерживается клиентом*

Здесь клиент устанавливает PPP-соединение с сервером поставщика услуг Интернета, который, в свою очередь, отвечает за создание и обслуживание туннельного соединения между собой и сервером, с которым желает взаимодействовать клиент.

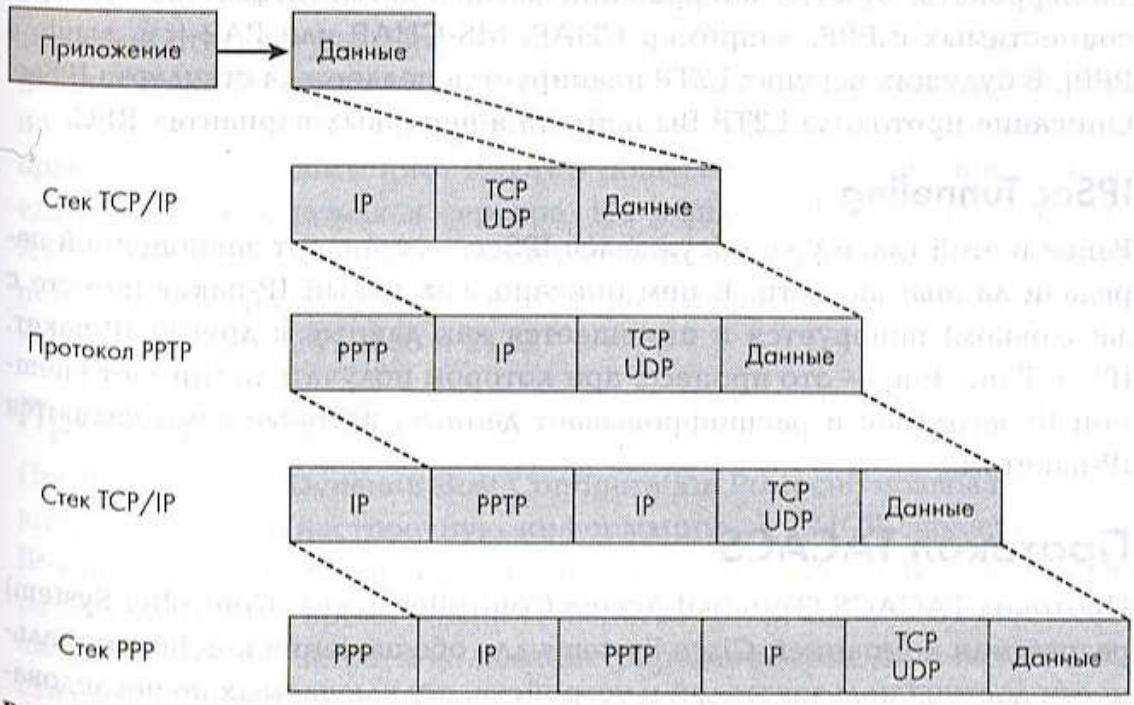


Рис. 3-23. *Структура пакетов при PPTP-соединении через последовательные линии*

На рис. 3-23 представлена упрощенная схема наложения протоколов при передаче данных по последовательной линии. (Протокол PPTP может работать и в локальной сети, но тогда эта схема несколько трансформируется.)

L2F

Туннельный протокол L2F (Layer 2 Forwarding) разработан компанией Cisco Systems и для реализации туннелинга использует протоколы канального уровня, например ATM или Frame relay. При этом необходимо чтобы все маршрутизаторы и серверы, через которые проходят данные (в том числе и серверы поставщика услуг Интернета), поддерживали L2F.

L2TP

Протокол L2TP (Layer 2 Tunneling Protocol) — результат стремления к единому, основанному на стандартах IETF, туннельному протоколу канального уровня. В нем сочетаются лучшие качества PPTP и L2F.

Протокол L2TP способен передавать PPP-кадры вложенными в протоколы IP, X.25, Frame relay или ATM. Для обслуживания туннельного соединения используется протокол UDP, а для передачи данных — сочетание UDP/PPP. Передаваемые по каналу данные можно сжать или зашифровать, причем шифрование выполняется любым из методов, совместимых с PPP, например CHAP, MS-CHAP или PAP (см. выше о PPP). В будущих версиях L2TP планируется поддержка стандарта IPSec. Описание протокола L2TP Вы найдете в черновых вариантах RFC.

IPSec Tunneling

Ранее в этой главе уже обсуждался IPSec — стандарт защищенной передачи данных по сети. В нем описано, как целый IP-пакет (вместе с заголовком) шифруется и помещается как данные в другой IP-пакет. IPSec Tunneling — это процесс, при котором получатель снимает внешний IP-заголовок и расшифровывает данные, которые сами являются IP-пакетом.

Протокол TACACS

Протокол TACACS (Terminal Access Controller Access Controller System) разработан компанией Cisco Systems для обеспечения конфиденциальности доступа пользователей и устройств, подключаемых по последовательной (телефонной) линии. При подключении клиент предъявляет серверу «удостоверение личности» (имя пользователя и пароль). В TACACS

предусмотрен протокол передачи серверу аутентификации «удостоверений» клиентов, подключающихся к серверу удаленного доступа. Первый их проверяет и сообщает последнему о возможности доступа.

Протокол TACACS имеет расширение — XTACACS, позволяющее использовать больше одного сервера аутентификации. Также в XTACACS предусмотрены механизмы слежения за временем подключения клиента, что удобно для учета и ведения статистики.

В качестве транспортного механизма как TACACS, так и XTACACS применяют протокол UDP. Компания Cisco не поддерживает продукты TACACS и XTACACS, разве что бесплатно предоставляет их исходный код. В TACACS пароли передаются открытым текстом, тогда как TACACS+ (модификация TACACS) поддерживает PAP и CHAP (описанные ранее в этой главе). Подробности — в RFC 1492.

Протокол RADIUS

Протокол RADIUS (Remote Authentication Dial-In User Service) используется для обеспечения конфиденциальности доступа пользователей и устройств, подключаемых по последовательной (телефонной) линии. Серверы удаленного доступа принимают звонки от пользователей и пересылают их «удостоверения» RADIUS-серверам. Последние их анализируют (возможно, подключаясь к другому серверу) и сообщают о возможности доступа. При этом проверяется не только подлинность пользователя (существование учетной записи и правильность пароля), но и предоставляется информация о том, к каким службам он имеет право доступа. Кроме того, RADIUS позволяет вести учет времени соединения пользователя и сервера. На рынке RADIUS конкурирует с XTACACS. RADIUS не обеспечен никакой поддержкой, кроме свободного распространения исходного кода. Подробнее об этом продукте можно узнать в RFC 2138 и 2139.

Протокол DHCP

Протокол DHCP (Dynamic Host Configuration Protocol) позволяет компьютерам получать настроечную информацию от DHCP-сервера. Это — IP-адрес, маска подсети, адрес шлюза по умолчанию, адрес DNS-сервера по умолчанию, время жизни IP-пакетов и т. д. Использование DHCP сильно сокращает затраты на администрирование сети. До того как DHCP стал использоваться повсеместно, администраторам сетей приходилось настраивать каждый компьютер вручную. Есть и еще одно преимущество DHCP: мобильным пользователям значительно проще подключить-

ся к корпоративной сети другого офиса (который, возможно, находится в другом городе). Так, например, статически выделенный в Нью-Йорке IP-адрес бывает несовместим с локальной сетью филиала этой же компании в Сиэтле. Кроме того, сервер удаленного доступа (RAS-сервер) может сам использовать DHCP для получения IP-адресов, назначаемых клиентам, подключающимся по телефонной линии.

DHCP позволяет клиентам получать информацию помимо IP-адреса (время жизни IP-пакетов, адрес маршрутизатора по умолчанию и т. д.), не получая при этом нового IP-адреса. Эта возможность называется DHCPInform и используется клиентами, которые уже получили IP-адрес, но требуют дополнительную информацию.

DHCP основан на протоколе BOOTP и предполагает, что маршрутизаторы пропускают BOOTP-сообщения. DHCP-сервер выделяет настроечные параметры «в аренду» и по ее завершении забирает их обратно. Этого нет в BOOTP. Учтите, что DHCP-клиенты не будут работать с BOOTP-сервером (и наоборот) без специальной доработки клиентского и серверного программного обеспечения.

В одной локальной сети или ее сегменте иногда действуют несколько DHCP-серверов. Все их надо правильно настроить, поскольку DHCP это чисто клиент-серверный протокол, не предусматривающий (по крайней мере, пока) взаимодействия между серверами. Клиент отправляет DHCP-серверу (или серверам) широковещательный запрос и может получить сразу несколько ответов с предложением ресурса, например IP-адреса, от разных серверов. Клиент выбирает один из них и широковещательно сообщает об этом. Для остальных DHCP-серверов это означает, что клиент не намерен использовать их ресурсы, поэтому они доступны для дальнейшего использования.

DHCP-сервер имеет несколько *рабочих областей* (scope), причем с каждой можно ассоциировать разные множества ресурсов, например диапазоны IP-адресов. Адреса разрешено резервировать. Обычно это делают для IP-адресов, принадлежащих не-DHCP-клиентам, например маршрутизаторам и DNS-серверам. Это позволяет DHCP-серверу служить единым хранилищем информации об IP-адресах как DHCP-, так и не-DHCP-клиентов.

Вовсе не обязательно, чтобы в каждом сегменте локальной сети существовал свой DHCP-сервер, поскольку маршрутизаторы, при соответствующей настройке, способны пересылать DHCP-запросы между сег-

ментами сети. Эти DHCP-запросы идентичны запросам протокола BOOTP и содержат физический адрес (например, Ethernet-адрес) запрашивающего клиента.

Помимо этого, в DHCP-запросе есть поле **IP-адрес шлюза** (Gateway IP address, Giaddr). Маршрутизатор может быть подключен сразу к нескольким сегментам сети, но всегда способен различить (по соответствию адаптер-сеть), из какого сегмента прибыл конкретный пакет (рис. 3-24). Для описания того, откуда прибыл DHCP-запрос, маршрутизатор заполняет поле **Giaddr**. В DHCP-запросах, не прошедших ни через один маршрутизатор, поле **Giaddr** равно 0. При получении такого запроса DHCP-сервер выделяет требуемый параметр из основной рабочей области (для локального сегмента сети).

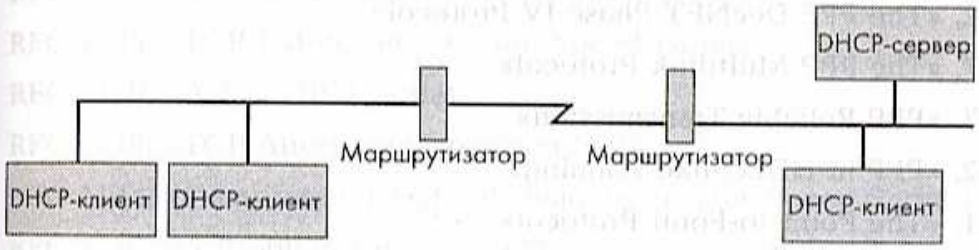


Рис. 3-24. *DHCP-сервер, обслуживающий несколько сегментов сети*

Для каждой рабочей области DHCP-сервера можно задать конкретный набор параметров, что позволяет зарезервировать диапазон IP-адресов для некоторого сегмента сети. Эта схема ограничена, поскольку из-за недостатка IP-адресов некоторым компаниям выделены адреса класса C. При этом, если сеть такой компании разделена на подсети, то для их соединения необходимо использовать несколько DHCP-серверов или маршрутизаторы.

Официальное описание протокола DHCP — в RFC 1541, созданном в IETF рабочей группой Dynamic Host Configuration.

Ссылки

Групповая IP-адресация и маршрутизация

http://www.ipmulticast.com

http://www.stardust.com

RFC 1584, «Multicast Extensions to OSPF»

RFC 1112, «Host Extensions for IP Multicasting»

RFC 1075, «Distance Vector Multicast Routing Protocol»

Протокол PPP

RFC 1764, «The PPP XNS IDP Control Protocol»

RFC 1763, «The PPP Banyan Vines Protocol»

RFC 1762, «The PPP DecNET Phase IV Protocol»

RFC 1717, «The PPP Multilink Protocol»

RFC 1663, «PPP Reliable Transmission»

RFC 1662, «PPP in HDLC-like Framing»

RFC 1661, «The Point-to-Point Protocol»

RFC 1638, «PPP Bridging Control Protocol»

RFC 1619, «PPP Over SONET/SDH»

RFC 1618, «PPP Over ISDN»

RFC 1598, «PPP in X.25»

RFC 1570, «PPP LCP Extensions»

RFC 1552, «The PPP Internetwork Packet Exchange Control Protocol»

RFC 1548, «The Point-to-Point Protocol»

RFC 1547, «Requirements for an Internet Standard Point-to-Point Protocol»

RFC 1378, «The PPP AppleTalk Control Protocol»

RFC 1377, «The PPP OSI Network Layer Control Protocol»

RFC 1334, «PPP Authentication Protocols»

RFC 1333, «PPP Link Quality Monitoring»

RFC 1332, «The PPP Internet Control Protocol»

Протокол TCP

RFC 1739, «A Primer on Internet and TCP/IP Tools»

RFC 1693, «An Extension to TCP: Partial Order Service»

RFC 1644, «T/TCP — TCP Extensions for Transactions Functional Specification»

RFC 1470, «Tools for Monitoring and Debugging TCP/IP Internets and Interconnected Devices»

RFC 1347, «TCP and UDP with Bigger Addresses, A Simple Proposal for Internet Addressing and Routing»

RFC 1337, «TIME-WAIT Assassination Hazards in TCP»

RFC 1323, «TCP Extensions for High Performance»

RFC 1273, «A Measurement Study of Changes in Service-Level Reachability in the Global TCP/IP Internet: Goals, Experimental Design, Implementation, and Policy Considerations»

RFC 1263, «TCP Extensions Considered Harmful»

RFC 1185, «TCP Extension for High-Speed Paths»

RFC 1180, «A TCP/IP Tutorial»

RFC 1146, «TCP Alternate Checksum Options»

RFC 1144, «Compressing TCP/IP Headers for Low-Speed Serial Links»

RFC 1110, «A Problem with the TCP Big Window Option»

RFC 1106, «TCP Big Window and Nak Options»

RFC 1072, «TCP Extensions for Long-Delay Paths»

RFC 1025, «TCP and IP Bake Off»

RFC 962, «TCP-4 Prime»

RFC 896, «Congestion Control in IP/TCP Internetworks»

RFC 879, «The TCP Maximum Segment Size and Related Topics»

RFC 872, «TCP on a LAN»

RFC 813, «Window and Acknowledgement Strategy in TCP»

RFC 793, «Transmission Control Protocol»

Маршрутизация

RFC 1787, «Routing in a Multiprovider Internet»

RFC 1786, «Representation of IP Routing Policies in a Routing Registry»

RFC 1723, «RIP Version 2 Carrying Additional Information»

RFC 1716, «Towards Requirements for Routers»

RFC 1702, «Generic Routing Encapsulation over IPv4 Networks»

RFC 1701, «Generic Routing Encapsulation»

RFC 1520, «Exchanging Routing Information Across Provider Boundaries in the CIDR Environment»

RFC 1519, «Classless Inter-Domain Routing: An Address Assignment and Aggregation Strategy»

RFC 1517, «Applicability Statement for the Implementation of Classless Inter-Domain Routing»

RFC 1479, «Inter-Domain Policy Routing Protocol Specification: Version 1»

RFC 1478, «An Architecture for Inter-Domain Policy Routing»

RFC 1397, «Default Route Advertisement In BGP2 and BGP3 Versions of the Border Gateway Protocol»

RFC 1388, «RIP Version 2 Carrying Additional Information»

RFC 1380, «IESG Deliberations on Routing and Addressing»

RFC 1364, «BGP OSPF Interaction»

RFC 1267, «Border Gateway Protocol Version 3»

RFC 1264, «Internet Routing Protocol Standardization Criteria»

RFC 1195, «Use of OSI IS-IS for Routing in TCP/IP and Dual Environments»

RFC 1136, «Administrative Domains and Routing Domains»

RFC 1104, «Models of Policy Based Routing»

RFC 1075, «Distance Vector Multicast Routing Protocol»

RFC 904, «Exterior Gateway Protocol»

Протокол IPv6

RFC 1972, «A Method for the Transmission of IPv6 Packets over Ethernet Networks»

RFC 1971, «IPv6 Stateless Address Autoconfiguration»

RFC 1970, «Neighbor Discovery for IPv6»

RFC 1933, «Translation Mechanisms for IPv6 Hosts and Routers»

RFC 1897, «IPv6 Testing Address Allocation»

RFC 1887, «An Architecture for IPv6 Unicast Address Allocation»

RFC 1886, «DNS Extensions to Support IPv6»

RFC 1885, «Internet Control Message Protocol for IPv6»

RFC 1884, «IPv6 Addressing Architecture»

RFC 1883, «IPv6 Specification»

RFC 1809, «Using the Flow Label Field in IPv6»

Протокол IPSec

RFC 1829, «The ESP DES-CBC Transform»

RFC 1828, «IP Authentication Using Keyed MD5»

RFC 1827, «IP Encapsulating Security Payload»

RFC 1826, «IP Authentication Header»

RFC 1825, «Security Architecture for the Internet Protocol»

Общие данные

http://www.intel.com/IAL/winsock2

http://www.microsoft.com/intdev/pptp/pptp-f.htm

http://www.microsoft.com/win32dev/netwrk/winsock2/ws295sdk.html

RFC 2139, «RADIUS Accounting»

RFC 2138, «Remote Authentication Dial-In User Service (RADIUS)»

RFC 1753, «IPng Technical Requirements of the Nimrod Routing and Addressing Architecture»

RFC 1752, «The Recommendation for the IPng Protocol»

RFC 1492, «An Access Control Protocol, Sometimes Called TACACS»

RFC 1392, «Internet User's Glossary»

RFC 1390, «Transmission of IP and ARP over FDDI Networks»

RFC 1386, «The U.S. Domain»

RFC 1256, «ICMP Router Discovery Messages»

RFC 1122, «Requirements for Internet Hosts — Communications Layers»

RFC 950, «Internet Standard Subnetting Procedure»

RFC 815, «IP Datagram Reassembly Algorithms»

RFC 792, «Internet Control Message Protocol»

RFC 791, «Internet Protocol»

RFC 768, «User Datagram Protocol»

RFC 719, «Discussion on RCTE»

ГЛАВА 4

Стандарты кодирования

В этой главе рассмотрены некоторые стандарты кодирования символов, файлов и других структур данных, а также отдельные протоколы, принятые в качестве стандартов. Также здесь рассказано о применении MIME (Multipurpose Internet Mail Extensions) для кодирования символов. Использование MIME для кодирования сообщений электронной почты описано в главе 9.

Кодирование символов

Кодировка символов определяет способ их представления в виде данных, пригодных к обработке на компьютере. Она обычно задается в виде таблицы, где каждому символу присвоено имя и числовое значение, которое используют как индекс в таблице и называют *кодом символа* (code point). Пример таблицы кодирования — таблица US-ASCII (табл. 4-1). Обратите внимание, что в кодировке каждому символу сопоставлено число (код символа), но вовсе не обязательно, чтобы эти числа задавали порядок на множестве всех символов. Так, например, для сортировки текстовых строк нельзя использовать коды символов. Для этого существует другая таблица — *сортировки* (sorting table). В следующих разделах описаны некоторые схемы кодирования символов.

Семиразрядный код ASCII/US-ASCII

Строго говоря, в 7-разрядной кодировке ASCII используются все 8 бит, но старший всегда равен 0. Если говорить в терминах объема, то каждый символ занимает 8 бит (1 байт). Эта схема, именуемая US-ASCII, описана в стандарте ISO 646. Кодировку US-ASCII поддерживает подавляющее большинство компьютеров. Другие кодировки можно рассматривать как расширения US-ASCII: хотя они и содержат больше символов, однако первые 128 совпадают с представленными в US-ASCII. При помощи ISO 646 можно использовать разные языки, например английский, суахили и латынь. В табл. 4-1 показана 7-разрядная кодировка US-ASCII.

Таблица 4-1. *Таблица US-ASCII*

Символ	Десятичное значение	Шестнадцатеричное значение
NUL	0	0
SOH	1	1
STX	2	2
ETX	3	3
EOT	4	4
ENQ	5	5
ACK	6	6
BEL	7	7
BS	8	8
HT	9	9
NL	10	A
VT	11	B
NP	12	C
CR	13	D
SO	14	E
SI	15	F
DLE	16	10
DC1	17	11
DC2	18	12
DC3	19	13
DC4	20	14
NAK	21	15
SYN	22	16
ETB	23	17
CAN	24	18
EM	25	19
SUB	26	1A
ESC	27	1B
FS	28	1C
GS	29	1D
RS	30	1E
US	31	1F
SP	32	20
!	33	21
"	34	22
#	35	23
$	36	24
%	37	25

Таблица 4-1. *Таблица US-ASCII* (продолжение)

Символ	Десятичное значение	Шестнадцатеричное значение
&	38	26
'	39	27
(	40	28
)	41	29
*	42	2A
+	43	2B
,	44	2C
-	45	2D
.	46	2E
/	47	2F
0	48	30
1	49	31
2	50	32
3	51	33
4	52	34
5	53	35
6	54	36
7	55	37
8	56	38
9	57	39
:	58	3A
;	59	3B
<	60	3C
=	61	3D
>	62	3E
?	63	3F
@	64	40
A	65	41
B	66	42
C	67	43
D	68	44
E	69	45
F	70	46
G	71	47
H	72	48
I	73	49
J	74	4A
K	75	4B

(см. след. стр.)

Таблица 4-1. *Таблица US-ASCII* (продолжение)

Символ	Десятичное значение	Шестнадцатеричное значение
L	76	4C
M	77	4D
N	78	4E
O	79	4F
P	80	50
Q	81	51
R	82	52
S	83	53
T	84	54
U	85	55
V	86	56
W	87	57
X	88	58
Y	89	59
Z	90	5A
[	91	5B
\	92	5C
]	93	5D
^	94	5E
_	95	5F
`	96	60
a	97	61
b	98	62
c	99	63
d	100	64
e	101	65
f	102	66
g	103	67
h	104	68
i	105	69
j	106	6A
k	107	6B
l	108	6C
m	109	6D
n	110	6E
o	111	6F
p	112	70
q	113	71

Таблица 4-1. *Таблица US-ASCII* (продолжение)

Символ	Десятичное значение	Шестнадцатеричное значение
r	114	72
s	115	73
t	116	74
u	117	75
v	118	76
w	119	77
x	120	78
y	121	79
z	122	7A
{	123	7B
|	124	7C
}	125	7D
~	126	7E
DEL	127	7F

Восьмиразрядная кодировка

Определенная в ISO 646 7-разрядная кодировка не годится для целого ряда языков. В результате ISO определила серию стандартов кодирования, известную как серия 8859. Здесь используются 8 бит вместо 7, то есть старший бит может принимать значения как 0, так и 1. Первые 128 символов в кодировках серии 8859 совпадают с символами кодировки ISO 646. Некоторые наборы символов серии 8859 и соответствующие им языки перечислены в табл. 4-2. Подробности — по адресу *http://www.w3.org/International*.

Таблица 4-2. *Некоторые наборы символов серии 8859*

Набор символов	Языки
8859-1	Африкаанс, албанский, баскский, каталанский, датский, голландский, английский, ирландский, фарерский, финский, французский, галльский (Galego), немецкий, исландский, итальянский
8859-2	Хорватский, чешский, венгерский
8859-3	Эсперанто, мальтийский
8859-5	Болгарский, белорусский, македонский, русский
8859-6	Арабский
8859-7	Греческий
8859-8	Иврит
8859-9	Турецкий
8859-10	Саамский, латышский, литовский

Чаще других используется кодировка ISO 8859-1. Остальные наборы символов применяется гораздо реже, и многие производители, к сожалению, не полностью придерживаются спецификаций. Время, конечно, покажет, но кодировка Unicode, описанная в следующем разделе, считается более перспективной, чем стандарты ISO 8859.

Unicode (16-разрядная кодировка)

Кодировка Unicode становится все более важной. Ее стандарт опубликован организацией Unicode Consortium. Кодировка Unicode — это попытка объединить все стандарты в один, что позволит создавать программное обеспечение, без проблем работающее с разными языками и пользователями. В настоящее время поддержка Unicode встроена в операционные системы Microsoft Windows NT и Novell NetWare версии 4.

В кодировке Unicode каждый символ представлен двумя байтами, то есть 16-разрядным значением. Первые 128 кодовых значений Unicode совпадают с кодами US-ASCII, а первые 256 — с кодами ISO 8859-1.

Вот основные концепции стандарта Unicode.

- Универсальность — охватываются как машинные, так и разговорные языки. Стандарт Unicode содержит кодировки почти всех разговорных языков мира, включая латынь, греческий, индийские языки, древнекорейскую письменность (Hangul), китайский, японский и санскрит.
- Эффективность — спецификация призвана повысить эффективность программного обеспечения. Благодаря фиксированной длине кода символа, ПО не приходится выискивать ESC-последовательности, а эффективность обработки текстов сравнима с обычным письмом.
- Однозначность — гарантия того, что одному 16-разрядному значению всегда соответствует один и тот же символ.

Помимо самого стандарта, Unicode Consortium публикует и другие документы. Например, Технический отчет (Unicode Technical Report), который не является (пока!) частью стандарта. Обычно содержимое отчетов включается в очередную версию стандарта Unicode, поэтому их используют как инструмент для незначительных изменений стандарта. Последняя принятая спецификация названа Unicode 2.1. Вот наиболее существенные отличия Unicode 2.1 от Unicode 2.0:

- добавлен символ — заменитель объекта;
- добавлен символ валюты Евро (Euro);

- исправлены опечатки в разных разделах, в том числе в свойствах символов, идентификаторах, в описании двунаправленного письма и других;
- исправлен пример программы преобразования UTF-7;
- внесены изменения в базу данных Unicode-символов.

Также Unicode Consortium публикует черновики технических отчетов. Их содержимое редактируют и размещают на Web-узел (*http://www.unicode.org*) для широкого обсуждения.

В кодировке Unicode каждому символу соответствует одно кодовое значение. В отличие от US-ASCII, специальные символы, например знаки «минус» и «тире», имеют разный смысл и коды. Однако в Unicode не различаются, скажем, «а» обычная и курсивная. Это свойство шрифта, а не кодировки. Кроме того, в Unicode поддерживаются языки, в которых написание знаков выполняется не слева направо, а также унифицировано представление китайских, японских и корейских (Chinese/Japan/Korean, CJK) идеограмм.

В стандарте Unicode все кодовое пространство разделено на категории по лингвистическим и функциональным признакам (рис. 4-1).

- Область «Основные алфавиты» (General scripts) выделена для относительно маленьких наборов символов. Сюда входят латиница, кириллица, греческий, иврит, индийские и многие другие алфавиты. Естественно, коды первых 128 символов совпадают с кодами из таблицы US-ASCII.
- Область «Символы» (Symbols) предназначена для знаков пунктуации, научных, математических и специальных символов.
- Область «Фонетика и символы CJK» (CJK phonetics and symbols) содержит различные элементы фонетики и пунктуации, а также символы CJK.
- Область «Идеограммы CJK» (CJK ideographs) содержит 20 902 идеограммы.
- Область «Корейские слоги» (Hangul syllables) содержит 11 712 слогов.
- В области «Заменители» (Surrogates) зарезервированы 2 024 кодовых значений для будущих расширений.
- Область «Личные символы» (Private use) содержит 6 400 кодовых значений, определяемых пользователем или производителем.

- Область «Совместимость» (Compatibility) содержит символы, уже имеющие одно представление в Unicode, но требующие второе для обеспечения совместимости.

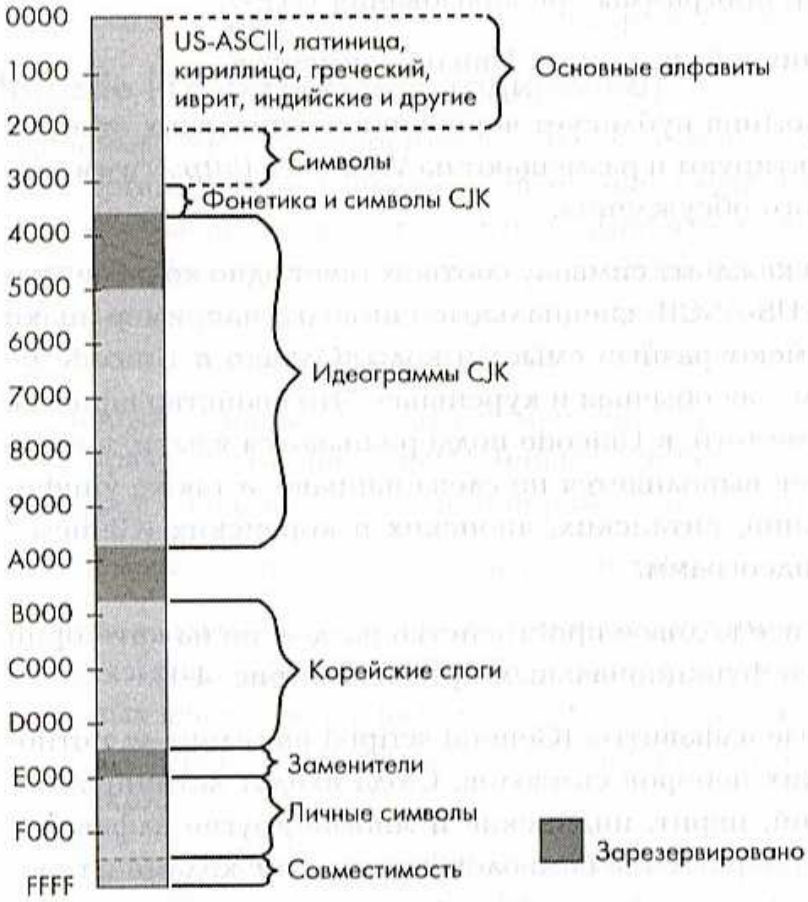


Рис. 4-1. *Распределение кодового пространства в Unicode*

Некоторые кодовые значения зарезервированы, а значения 0xFFFF и 0xFFFE (шестнадцатеричные), имеющие особый смысл, вообще нельзя применять.

Учтите, что кодировка Unicode не задает правил сортировки. Она лишь присваивает числовое значение каждому символу. Для сортировки строк необходимо использовать другую таблицу (ее описание выходит за рамки этой книги), предназначенную для конкретного языка. Также Unicode не имеет отношения к шрифтам.

Unicode и ISO 10646

В начале 90-х годов организации ISO и Unicode Consortium объединили свои усилия для разработки единого стандарта кодирования символов.

Однако каждая из этих организаций до сих пор публикует собственные спецификации (после тщательного согласования с партнером). Таким образом, ISO 10646-1:1993 — это стандарт ISO, соответствующий Unicode, с небольшими изменениями, касающимися корейского языка (и некоторыми дополнительными символами). Стандарт Unicode 2.0 соответствует ISO 10646.

ISO 10646 существует в двух видах:

- 4-байтовая, 31-разрядная кодировка. Все 2^{31} кодовых значения концептуально разделены на 128 подгрупп. Каждая состоит из 256 таблиц по 256 столбцов и 256 колонок. Эта кодировка называется UCS-4 (Universal Character Set, 4-byte form);
- 2-байтовая, 16-разрядная кодировка. Она называется UCS-2 (Universal Character Set, 2-byte form), или Основная многоязыковая таблица (Basic Multilingual Plane).

Стандарт Unicode 1.1 соответствует UCS-2 и является исправленной спецификацией ISO 10646. Подробности о ISO 10646 — по адресу *http://www.iso.ch*.

Формат UTF-7

Аббревиатура UTF-7 обозначает Universal Transformation Format, 7-bit form (7-разрядный универсальный формат перекодирования). В Unicode символы представлены 16-разрядными значениями, но современная сетевая инфраструктура не всегда позволяет передавать 8-разрядные данные и уж тем более 16-разрядные. UTF-7 — это способ перекодирования Unicode-символов в ASCII-символы. Unicode-символы, выходящие за рамки таблицы ASCII, представлены как «Shift-последовательности». Таким образом, UTF-7 позволяет передавать Unicode-символы по сетям, работающим только с 7-разрядными данными. Перекодирование UTF-7 обратимо, то есть обратным преобразованием можно восстановить исходные Unicode-данные.

Подробное описание UTF-7 — в RFC 1642; комментарии — по адресу *http://www.unicode.org*. UTF-7 не является частью спецификации ISO 10646. В книге «Unicode Standard Version 2.0», изданной Unicode Consortium, приведена программа на языке C, выполняющая прямое и обратное преобразования UTF-7.

Формат UTF-8

В UTF-8 (Universal Transformation Format, 8-bit form) 16-разрядные Unicode-символы преобразуются в последовательности длиной от 2 до

5 байт, где начальный байт определяет длину результирующей последовательности. Количество байт зависит от кода конкретного Unicode-символа, причем символы с кодами от 0 до 127 (ASCII-коды) представлены двумя байтами — начальным и собственно кодом. Формат UTF-8 считается достаточно эффективным, но иногда, при выделении символа из середины байтового потока, приходится переводить взгляд на 4 байта назад, чтобы найти начало символа. Перекодирование UTF-8 также обратимо (как UTF-7), исходные Unicode-данные получают обратным преобразованием.

Разработанный организацией X/Open Consortium, формат UTF-8 теперь стал частью спецификации ISO 10646. Полное описание Вы найдете на Web-странице Unicode Consortium по адресу *http://www.unicode.org*. В книге «Unicode Standard Version 2.0» также приведена программа на языке C, выполняющая прямое и обратное преобразования UTF-8.

Методы MIME-кодирования

Прежде всего, MIME — это протокол отправки и доставки сообщений электронной почты, но его часть описывает методы кодирования символов. Их в MIME всего два: Quoted-Printable и Base64. Методы MIME-кодирования применяют для преобразования 8-разрядных данных (использующих все 8 бит) в 7-разрядные ASCII-данные. Это гарантирует успешную передачу данных через Интернет, поскольку многие почтовые серверы обрабатывают данные только в формате ASCII. До появления MIME аналогичное кодирование/декодирование данных выполняли при помощи Uuencode и Uudecode соответственно. (Uuencode и Uudecode описаны далее в этой главе.)

В MIME-заголовке есть поле **Content-Transfer-Encoding** (тип кодировки содержимого). Для него возможны шесть значений: Quoted-Printable, Base64, Binary, 7Bit, 8Bit или X-Token.

Quoted-Printable

Кодировка Quoted-Printable («отображаемая с выделениями») применяется к данным, где большая часть символов относится к ASCII (7-разрядным). При этой кодировке все ASCII-символы остаются нетронутыми, а кодируются только те, у которых старший бит (8-разрядного байта) равен 1. При этом не-ASCII-символы представлены в виде пары символов, где первый — знак равенства (=). В результате большую часть сообщения можно прочитать, даже если оно не декодировано. Кодировка Quoted-Printable описана в RFC 1521.

Base64

Данные в формате Base64 невозможно прочесть без декодирования, а по размеру такое сообщение примерно на одну треть больше оригинала. При кодировании каждые три символа (24 бита) переводятся в четыре ASCII-символа (32 бита) следующим образом: 24 бита делятся на четыре группы по 6 бит, затем каждое 6-разрядное число служит индексом в массиве из 64 отображаемых символов (в алфавите), а символ, на который указывает этот индекс, помещается в результирующую строку. Есть в алфавите и 65-й символ — знак равенства (=), он указывает на необходимость специальной обработки. Эти 65 символов имеют одинаковые кодовые значения в кодировках US-ASCII, EBCDIC (Extended Binary Coded Decimal Interchange Code) и ISO 646 и называются алфавитом Base64 (табл. 4-3).

Таблица 4-3. *Алфавит Base64*

Код	Символ	Код	Символ	Код	Символ	Код	Символ
0	A	17	R	34	l	51	z
1	B	18	S	35	J	52	0
2	C	19	T	36	K	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v	нет	=
14	O	31	f	48	w		
15	P	32	g	49	x		
16	Q	33	h	50	y		

В закодированном потоке символы, не входящие в алфавит Base64, игнорируются. Это позволяет вставлять в поток данных дополнительные символы перевода строки и возврата каретки, чтобы сообщение смогло пройти через промежуточные почтовые шлюзы. При декодировании сообщения такие символы просто игнорируются.

Binary, 7Bit, 8Bit и X-Token

Тип Binary означает отсутствие кодирования, вероятность появления не-ASCII-символов и то, что из-за чрезмерной длины строк протокол SMTP не сможет успешно передать сообщение.

Тип 7Bit означает отсутствие кодировки, использование только ASCII-символов и длину строк, достаточную для успешной передачи сообщения по протоколу SMTP.

Тип 8Bit означает отсутствие кодировки, возможность присутствия не-ASCII-символов и короткие строки.

Тип X-Token означает, что способ кодирования согласован отправляющим и принимающим SMTP-серверами в частном порядке.

Типы Binary, 7Bit и 8Bit определены столь четко с учетом возможных перспектив: хотя сейчас они подразумевают отсутствие кодирования, но в будущем ситуация может измениться, тогда пригодится их формальное различие.

Uuencode и Uudecode

Uuencode позволяет закодировать файлы так, что их можно передавать по сетям, работающим только с 7-разрядными данными. Метод Uuencode преобразует файлы, в которых есть байты со значениями больше 127 (старший бит равен 1), в файлы, где все байты меньше 128 (старший бит равен 0). Такие файлы часто называют ASCII-файлами, так как они состоят только из ASCII-символов. Обратное преобразование выполняет Uudecode. Обычно файлы, закодированные при помощи Uuencode, имеют расширение «*.uu*», например «*filefoo.uu*».

Стандарт ASN.1

ASN.1 (Abstract Syntax Notation One) — это протокол представительного уровня (уровень 6) модели ISO/OSI. Он получил широкое применение в области телекоммуникаций, сотовой телефонии, в авиации, а также в Интернете. В ASN.1 определен язык описания абстрактных спецификаций стандартов и правил кодирования структур данных при передаче данных между двумя системами, использующими общий протокол. Например, при помощи ASN.1 описаны детали протокола LDAP (Lightweight Directory Access Protocol).

Протокол ASN.1 определен в двух черновых стандартах — ISO 8824 и ISO 8825. В этом разделе стандарт ASN.1 описан точки зрения кодирования структур данных.

Кодирование средствами ASN.1 удобно для автоматизации обработки данных, поскольку позволяет использовать компиляторы и подпрограммы. Правила кодирования для ASN.1 описаны в следующих разделах.

Базовая кодировка (BER)

Базовая кодировка (Basic Encoding Rules, BER) основана на нескольких предопределенных базовых типах данных, именуемых *примитивными* (primitive). Используя их, приложение может строить сложные структуры данных, или *построенные типы* (constructed types). В BER каждая единица данных кодируется в виде «ярлык, длина, значение».

Сам ярлык имеет три характеристики — класс, форму и значение:

- классы ярлыков таковы — *универсальный* (Universal), *прикладной* (Application), *личный* (Private) и *контекстно-зависимый* (Context Specific). К универсальному классу относятся ярлыки, применяемые к типам данных, определенным в спецификации ASN.1. К прикладному классу относятся ярлыки, заданные конкретными приложениями, к личному — ярлыки, заданные производителями, а ярлыки последнего класса имеют смысл только по отношению к содержимому структуры данных;
- форма ярлыка — *примитивный* (Primitive) или *построенный* (Constructed);
- возможные значения ярлыков перечислены в таблице.

Тип данных	Значение ярлыка
Boolean	1
Integer	2
Bit String	3
Octet String	4
NULL	5
Object Identifier	6
Object Descriptor	7
EXTERNAL	8
Real	9
Enumerated	10
Sequence	16
Set	17
NumericString	18
PrintableString	19

(см. след. стр.)

(продолжение)

Тип данных	Значение ярлыка
TeletextString	20
VideoTextString	21
IA5String	22
UTCTime	23
GeneralizedTime	24
GraphicString	25
VisibleString	26
GeneralString	27

Если единица кодируемых данных относится к типу «построенный», то ее длина может быть неизвестна. В этом случае значением ярлыка становится другая конструкция «ярлык, длина, значение».

Каноническая кодировка (CER)

Каноническая кодировка (Canonical Encoding Rules, CER) — это разновидность BER, способная оперировать огромными объемами данных, используя меньше памяти. Для применения BER сообщение должно быть полностью доступно, тогда как при CER можно начать кодировку еще неоконченного сообщения.

Особая кодировка (DER)

Особая кодировка (Distinguished Encoding Rules, DER) — разновидность BER с возможностью обеспечения секретности данных. (Подробности см. в главе 5.) С ростом популярности электронной коммерции DER станут применять все чаще.

Пакетная кодировка (PER)

Пакетная кодировка (Packet Encoding Rules, PER) — аналогична BER, но значительно эффективнее, поскольку создает более компактный код при меньших вычислительных затратах. Эта кодировка используется, когда мощность процессора и емкость канала передачи весьма ограничены (например, в авиационной технике).

Ссылки

CCITT Recommendation X-288, «Specification of Abstract Syntax Notation.One»

CCITT Recommendation X-209, «Specification of Basic Encoding Rules for ASN.1»

http://www.iso.ch

http://www.unicode.org

http://www.w3.org/International

RFC 2279, «UTF-8: A Transformation of ISO 1064E»

RFC 2152, «UTF-7: A Mail-Safe Transformation of Unicode»

RFC 2049, «Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples»

RFC 2048, «Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures»

RFC 2047, «MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text»

RFC 2046, «Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types»

RFC 2045, «Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies»

RFC 1698, «Octet Sequences for Upper Layer OSI»

RFC 1642, «UTF-7: A Mail-Safe Transformation Format of Unicode»

RFC 1521, «MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies»

ГЛАВА 5

Основы криптографии и защиты информации

Криптография — это наука о представлении информации в виде, понятном лишь посвященным. Для этого применяется криптографический алгоритм, или *шифр*. Преобразованная информация называется шифртекстом. Исходная информация называется открытым текстом. *Криптоанализ* — искусство «взлома» шифров. *Аутентификация* — это проверка абонентом подлинности пользователя или процесса, с которым он обменивается информацией. Аутентификация может быть односторонней, когда только один абонент проверяет подлинность другого, или двухсторонней, когда это делают оба абонента.

В этой главе описаны некоторые стандарты шифрования и основы защиты информации. На рис. 5-1 схематично представлен материал этой главы.

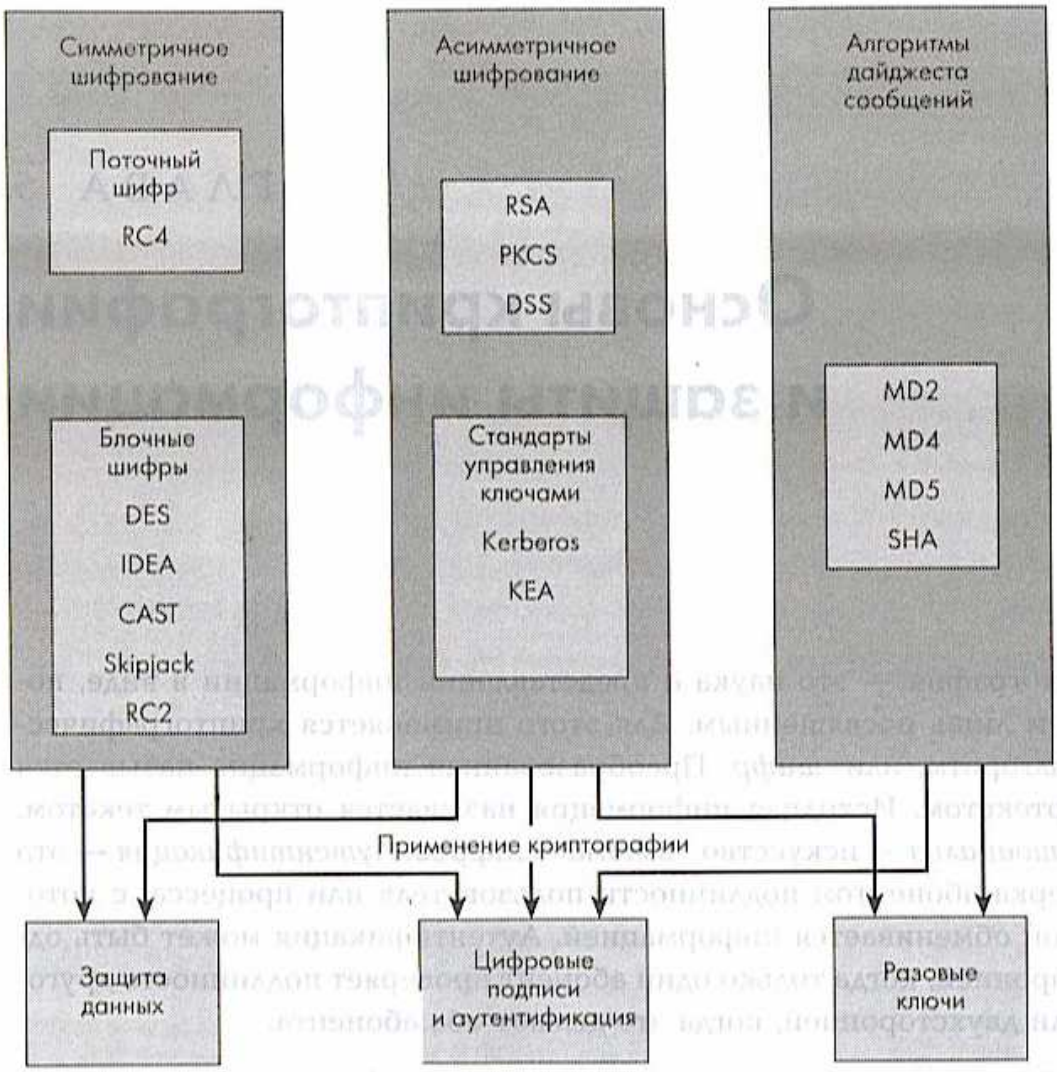


Рис. 5-1. *Криптографические методы и шифры*

Методы шифрования

В общем смысле возможны лишь два типа шифрования: *симметричное* (или классическое) и *асимметричное* (или шифрование с открытым ключом).

Симметричное шифрование

Идея симметричного шифрования описывается математической формулой вида

```
Шифртекст = Функция (Открытый текст, Ключ)
```

с обратной функцией вида

```
Открытый текст = Обратная Функция (Шифртекст, Ключ).
```

На рис. 5-2 показано преобразование информации при симметричном шифровании.

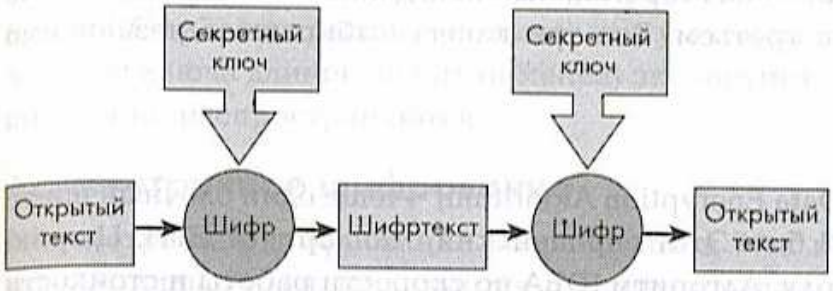


Рис. 5-2. *Симметричное шифрование*

Суть в том, что даже при известных прямой и обратной функциях, восстановление открытого текста без знания ключа невозможно. Симметричное шифрование также называют шифрованием с секретным ключом, поскольку одинаковый секретный ключ должен быть и у отправителя и у получателя. В следующих разделах рассмотрены наиболее важные алгоритмы симметричного шифрования, или шифры. Существует два типа шифров: *блочные* и *поточные*. Первые преобразуют блок входных данных (некоторого объема) в блок шифртекста (вообще говоря, другого объема). Вторые — открытый текст в шифртекст по одному биту за такт.

DES

DES (Data Encryption Standard) — блочный шифр, обрабатывающий блоки по 64 бита, используя ключ длиной 56 бит. Он уже достаточно изучен, протестирован и признан весьма стойким. Существует два основных режима работы DES: ECB (Electronic Code Book) и CBC (Cipher Block Chaining). В первом режиме DES обрабатывает за такт 64 бита данных, используя один и тот же 56-битный ключ. Таким образом, каждый блок шифруется независимо от других. В режиме CBC перед шифрованием очередного 64-битного блока к нему применяется операция «исключающее ИЛИ» с предыдущим блоком. Это гарантирует, что одинаковые 64-битные блоки, находящиеся в разных местах открытого текста, будут зашифрованы по-разному.

DES разработан как очень быстрый алгоритм, пригодный для аппаратной реализации. Экспорт продукции, использующей DES, запрещен правительством США. DES принят ANSI в качестве национального стандарта шифрования США.

В ситуациях, когда надежность алгоритма DES кажется недостаточной, используется его модификация — Triple-DES. Вообще говоря, существует

несколько вариантов Triple-DES. Наиболее простой — перешифрование. Открытый текст шифруется алгоритмом DES на первом ключе, полученный шифртекст — на втором и, наконец, данные, полученные после второго шага, — на третьем. Все три ключа выбираются независимо друг от друга.

IDEA

IDEA (International Data Encryption Algorithm) — еще один блочный шифр с длиной ключа 128 бит. Этот европейский стандарт (от ETH, Цюрих) предложен в 1990 году. Алгоритм IDEA по скорости работы и стойкости к анализу не уступает алгоритму DES.

CAST

CAST — это блочный шифр, использующий 128-битный ключ в США и 40-битный — в экспортном варианте. CAST используется компанией Northern Telecom (NORTEL).

Skipjack/Capstone

Шифр Skipjack, разработанный Агентством национальной безопасности США (National Security Agency, NSA), использует 80-битные ключи. Это часть проекта Capstone, цель которого — разработка общедоступного криптографического стандарта, удовлетворяющего требованиям правительства США. Capstone включает в себя четыре основных компонента:

- шифр Skipjack;
- алгоритм цифровой подписи на базе стандарта DSS (Digital Signature Standard);
- хеш-функцию на базе алгоритма SHA (Secure Hash Algorithm);
- микросхему, реализующую все вышеизложенное (например, Fortezza — PCMCIA-плата, основанная на этой микросхеме).

В реализации Skipjack использована патентованная микросхема Capstone, алгоритм которой засекречен. Ожесточенный спор ведется по поводу того, что в Skipjack будут предусмотрены ключи для «посреднических агентств» (escrow agencies), которые смогут дешифровать сообщения по требованию суда.

RC2 и RC4

Шифры RC2 и RC4 разработаны Роном Райвестом (Ron Rivest), одним из основателей компании RSA Data Security, и запатентованы этой ком-

панией. Они используют ключи разной длины, а в экспортируемых продуктах заменяют DES. Шифр RC2 — блочный, с длиной блока 64 бита; RC4 — поточный. По замыслу разработчиков производительность RC2 и RC4 должна быть не меньше, чем у алгоритма DES. При использовании ключа длиной 40 бит (и менее) экспортные ограничения на эти шифры не распространяются.

Асимметричное шифрование

Асимметричное шифрование разработано в 70-х годах нашего столетия. Основная идея заключается в использовании пары ключей. Первый — *открытый ключ* (public key) — доступен всем и используется теми, кто собирается послать сообщение владельцу ключа. Второй — *личный ключ* (private key) — известен только владельцу. Интересно то, что при асимметричном шифровании два ключа, как правило, взаимозаменяемы. То есть информацию, зашифрованную на личном ключе, расшифровать можно только используя открытый ключ, и наоборот. Это свойство лежит в основе концепции цифровой подписи, обсуждаемой в последующих разделах. Стойкость асимметричного шифрования основана на предположении, что поиск делителей очень большого натурального числа, являющегося произведением двух простых, крайне трудоемкая процедура. Математического доказательства этого предположения не существует, но практикой оно подтверждено. Базовые идеи криптографии с открытым ключом показаны на рис. 5-3.

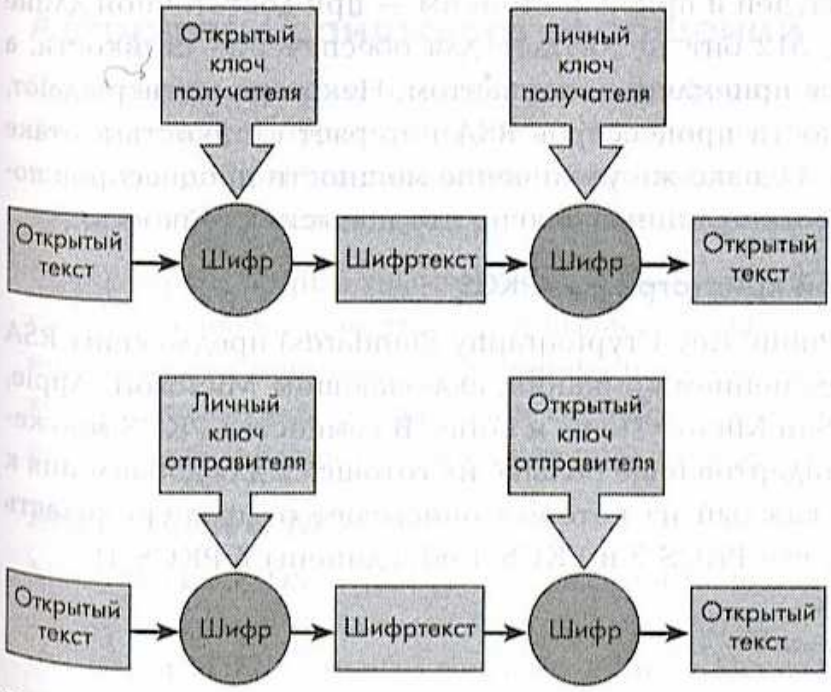


Рис. 5-3. *Асимметричное шифрование*

В верхней части рис. 5-3 показано шифрование текста с использованием открытого ключа получателя. Получатель же, чтобы прочитать сообщение, использует собственный личный ключ. Это гарантирует, что только адресат (который, предположительно, является единственным обладателем личного ключа получателя) может расшифровать сообщение.

В нижней части рисунка показан другой способ асимметричного шифрования — шифрование личным ключом отправителя и расшифровка открытым ключом отправителя. Если открытый текст заранее известен получателю, то на основе этого метода строится цифровая подпись, поскольку сообщение такого вида (с предопределенным открытым текстом) может прийти только от одного отправителя, знающего личный ключ.

RSA

В названии шифра RSA закодированы фамилии его изобретателей: Рона Райвеста (Ron Rivest), Ади Шамира (Adi Shamir) и Леонарда Элдемана (Leonard Aldeman) — основателей компании RSA Data Security. RSA не только самый популярный из асимметричных, но, пожалуй, вообще самый известный шифр. Математическое обоснование RSA таково: поиск делителей очень большого натурального числа, являющегося произведением двух простых, крайне трудоемкая процедура. Также по открытому ключу очень сложно вычислить парный ему личный ключ. Шифр RSA всесторонне изучен и признан стойким — при достаточной длине ключей. Например, 512 бит не хватает для обеспечения стойкости, а 1 024 бита считается приемлемым вариантом. Некоторые утверждают, что с ростом мощности процессоров RSA потеряет стойкость к атаке полным перебором. Однако же увеличение мощности процессоров позволит применить более длинные ключи, что повысит стойкость.

Стандарты открытой криптографии PKCS

Стандарты PKCS (Public Key Cryptography Standards) предложены RSA Laboratories и объединением компаний, включающим Microsoft, Apple, Digital Equipment, Sun Microsystems и Lotus. В семействе PKCS множество различных стандартов (еще больше их готовится для добавления к PKCS в будущем), каждый из которых описывает отдельную область (табл. 5-1). (Учтите, что PKCS 2 и PKCS 4 объединены в PKCS 1).

Таблица 5-1. *Стандарты PKCS*

Название	Описание
PKCS 1	Шифрование при помощи открытых ключей RSA
PKCS 3	Протокол Диффи-Хеллмана (Diffe-Hellman) обмена и согласования ключей
PKCS 5	Шифрование с использованием секретного ключа
PKCS 6	Формат сертификатов, являющихся надмножеством сертификатов X.509
PKCS 7	Синтаксис сообщений, содержащих шифртекст и цифровую подпись
PKCS 8	Формат личных ключей
PKCS 9	Структуры данных, используемые в других PKCS
PKCS 10	Синтаксис запросов на сертификацию
PKCS 11	Описывает API для устройств, использующих криптографические функции, например для смарт-карт

Стандарт DSS

Стандарт DSS (Digital Signature Standard) одобрен правительством США. Длина используемого ключа варьируется в пределах от 512 до 1 024 бит. DSS предназначен для создания цифровой подписи (см. далее раздел о цифровой подписи), но не для закрытия информации. В стандарте DSS найдены некоторые слабые места защиты, вследствие чего он не так широко распространен.

Алгоритмы дайджеста сообщений

Прежде чем говорить о применении шифров, например, для аутентификации или цифровой подписи, полезно рассмотреть алгоритмы вычисления дайджеста сообщений. Они, вместе с алгоритмами асимметричного шифрования, положены в основу систем цифровой подписи.

Сначала рассмотрим *хеш-функцию* (hash function). Она получает на вход произвольное количество бит и выдает на выходе строку фиксированной длины — хеш-код. Если хэш-функцию очень сложно обратить, то хеш-код называют *дайджестом сообщения* (message digest). Алгоритм вычисления дайджеста сообщения предполагает уникальность хеш-кода.

MD2, MD4 и MD5

MD2, MD4 и MD5 — алгоритмы дайджеста сообщений, разработанные Роном Райвестом. Каждый из них вырабатывает 128-битный хеш-код. Алгоритм MD2 — самый медленный, а MD4 — самый быстрый. Алгоритм MD5 можно считать модификацией MD4, где скоростью пожерт-

вовали ради увеличения безопасности. Более подробную информацию Вы найдете в RFC 1321 (MD2), RFC 1320 (MD4) и RFC 1319 (MD5).

SHA

SHA (Secure Hash Algorithm) — это алгоритм вычисления дайджеста сообщений, вырабатывающий 160-битный хеш-код. Алгоритм SHA одобрен правительством США (как часть проекта Capstone). Он несколько надежнее MD4 и MD5, так как вырабатывает более длинный хеш-код, который снижает вероятность того, что разные входные последовательности будут преобразованы в один и тот же хеш-код.

Применение шифров

На основе описанных выше методов разработано много интересных способов применения шифров. Некоторые из них описаны в следующих разделах.

Защита информации и каналов связи

Использование средств защиты гарантирует конфиденциальность информации при передаче по каналу связи. Например, SSL (Secure Socket Layer) версий 2 и 3 компании Netscape и PCT (Private Communication Technology) компании Microsoft позволяют защищать информацию на транспортном уровне. Более того, SSL и PCT известны не только как криптографические приложения. Они также предоставляют API для программистов, использующих интерфейс транспортного уровня. Подробнее SSL и PCT описаны ниже в этой главе.

Цифровая подпись

Цифровая подпись (digital signature) — это способ проверки целостности содержимого сообщения и подлинности его отправителя. Она реализуется при помощи асимметричных шифров и хеш-функций. Цифровая подпись основана на обратимости асимметричных шифров, а также на взаимосвязанности содержимого сообщения, самой подписи и пары ключей: изменение одного из этих элементов сделает невозможным подтверждение подлинности подписи. Работа алгоритма цифровой подписи проиллюстрирована на рис. 5-4.

Отправитель вычисляет дайджест сообщения (в левом верхнем углу рисунка), шифрует его своим личным ключом и отправляет вместе с письмом. Получатель, приняв сообщение, расшифровывает дайджест открытым ключом отправителя. Кроме того, получатель сам вычисляет дайджест принятого сообщения и сравнивает его с расшифрованным.

Если два дайджеста совпадают, то подпись подлинная. В противном случае либо содержание сообщения изменено, либо подпись подделана.

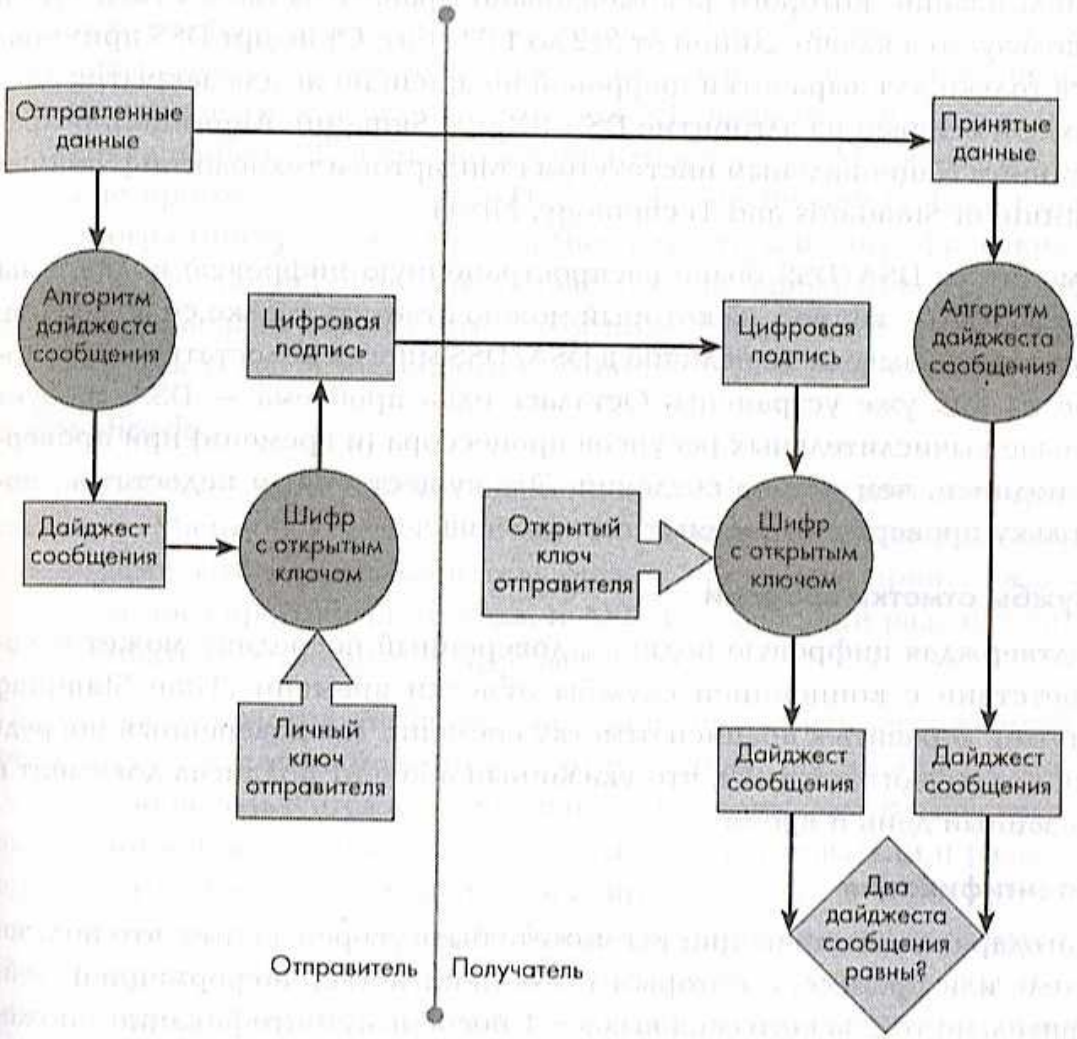


Рис. 5-4. *Модель использования цифровой подписи*

Цифровая подпись на основе RSA

Шифр RSA также используют для выработки и проверки цифровой подписи. Чтобы подписать сообщение, отправитель шифрует его своим личным ключом и отправляет подпись вместе с сообщением. Получатель расшифровывает подпись открытым ключом отправителя и сравнивает результат с принятым сообщением. Если сообщение подлинное, то полученное и расшифрованное сообщение должны совпасть.

Заметим, что при использовании RSA для закрытия информации отправитель шифрует сообщение открытым ключом получателя (в отличие от цифровой подписи, где применяется личный ключ отправителя).

Стандарт цифровой подписи DSS

Стандарт DSS (Digital Signature Standard) — это асимметричный шифр, использование которого рекомендовано правительством США. Здесь используются ключи длиной от 512 до 1 024 бит. Стандарт DSS применяется только для выработки цифровой подписи, но не для закрытия данных. Он основан на алгоритме DSA (Digital Signature Algorithm), разработанном Национальным институтом стандартов и технологий (National Institute of Standards and Technology, NIST).

Заменит ли DSA/DSS более распространенную цифровую подпись на основе RSA — вопрос, на который можно ответить только со временем. Критики указывали на наличие в DSA/DSS многих недостатков, некоторые из них уже устранены. Осталась одна проблема — DSA требует больше вычислительных ресурсов процессора (и времени) при проверке подписи, чем при ее создании. Это существенный недостаток, поскольку проверку выполняют гораздо чаще.

Службы отметки времени

Подтверждая цифровую подпись, доверенный посредник может в, соответствии с концепцией службы отметки времени (Time Stamping Service), добавить к подписи отметку времени. Так доверенный посредник подтвердит тот факт, что указанный абонент подписал документ в указанный день и время.

Аутентификация

Благодаря аутентификации Вы можете быть уверены в том, что пользователь или процесс, с которым Вы обмениваетесь информацией, действительно тот, за кого себя выдает. Способов аутентификации множество. Иногда ее проводят по уникальному свойству, присущему пользователю или процессу, например по отпечаткам пальцев. Или по знанию пользователем или процессом некоторой уникальной информации. В случае цифровой подписи таковой будет личный ключ отправителя. SSL и PCT не только обеспечивают конфиденциальность информации, но и предоставляют услуги аутентификации.

Еще один общий метод аутентификации — протокол Microsoft NTLM Challenge/Response, разработанный на основе первой версии LAN Manager для операционной системы OS/2. Этот способ аутентификации основан на обладании пользователем или процессом некоей информацией, в данном случае секретным паролем. Клиент соединяется с сервером, а тот посылает клиенту запрос. Клиент на основе хеш-кода

пользовательского пароля формирует ответ и отправляет его серверу. Сервер под управлением Microsoft Windows NT, обладающий информацией о паролях пользователей, сумеет независимо вычислить хеш-код и проверить корректность ответа клиента. [Строго говоря, сервер может находиться в другом домене; в этом случае для проверки запроса пользователя применяется свойство Windows NT *проверка на допуск* (pass-through validation).] Прекрасно документированный API облегчает применение протокола Microsoft NTLM Challenge/Response. Таким образом, программисту не надо досконально изучать криптографические алгоритмы. Но, самое главное, пользовательский пароль никогда не передается по каналам связи в открытом виде. Подробности — в разделе «Поставщик услуг безопасности в Windows NT» в этой главе.

Authenticode

Authenticode — это новое детище компании Microsoft, впервые опробованное в Microsoft Internet Explorer 3.0. Фактически, это технология, средствами которой пользователь может проверить происхождение загружаемого программного кода, и API, позволяющий разработчикам подписывать собственные программы.

Применяя технологию Authenticode, можно проверить загружаемый из Интернета код как на подлинность отправителя, так и на целостность. Здесь явно используется асимметричная криптография. В качестве сертификационной службы выступает Verisign. Пользователи и разработчики программного обеспечения могут проверять чужие и получать собственные сертификаты. Браузер Internet Explorer позволяет пользователю выбрать такой уровень безопасности, когда проверяется весь код, загружаемый из сети Интернет. Применив асимметричное шифрование, пользователь проверит подлинность любого полученного программного кода. Посредством программы Windows Trust Verification Service клиент может задать другой уровень безопасности, например считать весь код, поступающий от определенной компании, безопасным или, наоборот, требующим проверки.

Разовые ключи

Так какой же алгоритм шифрования следует использовать — симметричный или асимметричный? На практике оба метода применяются одновременно. Отчасти из-за того, что асимметричное шифрование значительно медленнее симметричного. Поэтому отправитель вырабатывает случайный секретный ключ и его средствами шифрует сообщение (симметричным алгоритмом). Письмо посылается вместе с секретным

ключом, который зашифрован открытым ключом получателя (алгоритмом с открытым ключом). Такой, выработанный случайным образом, ключ называется *разовым* (message key).

Похожая ситуация возникает, когда сообщение необходимо разослать одновременно нескольким получателям. При асимметричной схеме шифрования сообщение придется отдельно зашифровать для каждого абонента, используя его открытый ключ. Это потребует много времени. Вместо этого можно зашифровать сообщение случайно выработанным секретным ключом и затем шифровать только секретный ключ открытым ключом каждого из получателей.

Протокол SET

Протокол SET (Secure Electronic Transaction) позволяет расширить возможности и рамки коммерческой деятельности в Интернете. Спецификация SET разработана совместно компаниями Visa и MasterCard, при участии Microsoft, Netscape, IBM и GTE. Последняя спецификация SET состоит из трех томов: бизнес-описания, описания протокола и руководства программиста. SET — это технология защищенной передачи через Интернет информации о кредитных картах и API для разработки коммерческих приложений в Интернете. В протоколе SET для шифрования информации используется DES, а для шифрования секретного ключа и номера кредитной карты — RSA.

Протокол SSL

Стандарт Winsock описан в главе 3. Протокол SSL (Secure Socket Layer) дополняет его: он позволяет клиенту, обменивающемуся информацией с сервером, применять защиту данных и аутентификацию. При инициализации сеанса связи протокол SSL должен согласовать симметричный сеансовый ключ (session key) и алгоритм шифрования. Симметричный ключ используется для зашифрования и расшифрования информации. В это же время (установления сеанса) можно выполнить аутентификацию клиента и сервера. Когда согласование параметров сеанса завершится, клиент и сервер смогут безопасно обмениваться информацией, используя шифрование. Протокол SSL поддерживает RSA с открытым ключом — во время установления сеанса; RC2, RC4, IDEA, DES и Triple-DES — для шифрования информации; MD5 — для дайджеста сообщений.

Последнюю, третью, версию SSL поддерживают браузеры как Netscape, так и Microsoft.

Протокол PCT

Протокол PCT (Private Communication Technology) обеспечивает защиту для клиент-серверных приложений. Идея — скрыть от посторонних информационный обмен между клиентом и сервером. Сервер считается надежным (достоверным) всегда, а клиенты — не всегда.

Функционально PCT похож на SSL. Так же как в SSL, при установлении сеанса связи PCT согласует с клиентом секретный ключ и симметричный алгоритм шифрования (он используется для шифрования данных). Основное отличие от SSL в том, что PCT отделяет аутентификацию от шифрования и поэтому может иметь надежный механизм аутентификации и соответствовать экспортным ограничениям США. (Напомню, что правительство США запретило экспорт систем шифрования с длиной ключа более 40 бит.) Есть и другие отличия. PCT более эффективен, поскольку использует меньше сообщений и они более короткие по сравнению с SSL. PCT поддерживает алгоритмы RSA, Диффи-Хеллмана и Fortezza для управления ключами; DES, RC2 и RC4 — для шифрования данных; DSA и RSA — для цифровой подписи.

PCT реализован в Microsoft Internet Explorer версии 3 и выше, а также в Microsoft Internet Information Server (IIS) версии 2 и выше. Другие компании, например Spyglass и OpenMarket, тоже заявили о поддержке PCT.

Безопасность транспортного уровня

В IETF создана рабочая группа — Transport Layer Security (Безопасность транспортного уровня), которая пытается создать единый интерфейс защиты транспортного уровня, основанный на SSL, PCT и Secure Shell Remote Login. Компания Microsoft поддерживает это начинание, считая перспективным объединение достоинств SSL версии 3 и PCT версии 2.

Сертификаты

Асимметричное шифрование требует сохранения личного ключа в тайне. Кроме того, необходим надежный метод сопоставления открытого ключа и конкретного человека, процесса или объекта. Как же узнать, действительно ли открытый ключ принадлежит Джону Доу? (Если, например, Джейн Доу опубликует свой открытый ключ как ключ Джона Доу, то сможет прочитать все сообщения для Джона Доу.) В какой-то степени в этом помогает *сертификат* (certificate) — объект, надежно связывающий пользователя и его ключ. Сертификат содержит и другую информацию, например срок действия. Его выдает и подписывает

сертификационная служба (certificate authority, CA), но, заметьте, это не полностью решает проблему.

Как же мы можем проверить подлинность открытого ключа самой CA? Естественно, добавив еще одну CA (и еще, и еще...), пока не дойдем до абсолютно надежной CA. Допустима и такая схема: корпоративная CA, которой доверяют все сотрудники компании, региональная CA более высокого уровня, государственная CA и т. д.

Серверы сертификатов

Сертификаты хранятся как объекты службы каталогов или на специально выделенных для этого серверах. Существует ПО сервера сертификатов, созданное как Microsoft, так и Netscape. Время от времени сертификаты необходимо отзывать. Для этого их заносят в *список отозванных сертификатов* (certificate revocation list, CRL), издаваемый CA. (Сохранение целостности CRL затрагивает ряд вопросов, выходящих за рамки этой книги.)

X.509

X.509 — это стандарт, описывающий формат и синтаксис сертификатов. Строго говоря, стандарт X.509 также описывает службы аутентификации (но не используемый в них криптографический алгоритм), однако X.509 чаще ассоциируется именно с синтаксисом сертификатов. Различные стандарты, касающиеся аутентификации и защиты данных, например SSL, Secure HTTP (S-HTTP, описанный в главе 14), Privacy Enhanced Mail (PEM, описанный в главе 9), используют сертификаты X.509. Ожидается, что широкое распространение сертификаты X.509 получат в среде электронной коммерции. Первая версия X.509 опубликована 1988 году. Текущая версия — третья.

Главная задача сертификата — установить однозначное соответствие между пользователем и его открытым ключом. Вот некоторые поля сертификата стандарта X.509:

- номер версии X.509;
- идентификатор алгоритма службы аутентификации;
- название организации, выдавшей сертификат;
- срок действия сертификата;
- информация об открытом ключе пользователя.

Управление ключами

Как уже упоминалось выше, личные ключи надо хранить в тайне, сертификатами необходимо управлять, а при использовании симметричных шифров абонентам необходимо безопасно согласовывать секретные ключи и обмениваться ими. Kerberos — одно из приложений, выполняющее все эти задачи.

Kerberos

В протоколе Kerberos, разработанном в Массачусетском технологическом институте (Massachusetts Institute of Technology, MIT), реализовано несколько функций. Одна из них — хранение личных ключей в защищенной базе данных. Эти ключи известны только Kerberos и их владельцу. Еще одна функция — доверенный посредник между двумя абонентами, желающими обменяться секретными ключами. Доверенный в том смысле, что обе стороны не сомневаются в его надежности. Kerberos также предоставляет услуги аутентификации и рассылки ключей. В нем используется алгоритм шифрования DES.

Kerberos имеет три уровня защиты. Пользователь сам решает, какой уровень соответствует его потребностям. При установке сетевого соединения Kerberos проводит аутентификацию, и далее клиенты могут считать, что вся информация, поступающая с данного сетевого адреса, достоверна. Некоторые пользователи (или приложения) хотят, чтобы Kerberos проверял достоверность каждого сообщения, но не шифровал их содержимое. Это так называемые *безопасные сообщения* (safe messages). При наивысшем уровне безопасности каждое сообщение проверяется на достоверность и шифруется.

Kerberos работает по принципу службы выделения билетов (ticket). Например, пользователь регистрируется на клиентской системе. Ему предлагают ввести свое имя, а после отправляют запрос серверу аутентификации Kerberos. Тот проверяет существование пользователя с таким именем и в случае положительного ответа вырабатывает случайный ключ. Затем он создает билет, содержащий текущее время, время жизни билета, IP-адрес клиента и только что выработанный сеансовый ключ. Далее шифрует эту информацию ключом, известным только серверу аутентификации. Такой же билет шифруется личным ключом пользователя и отправляется клиенту. Затем клиентская рабочая станция запрашивает пароль пользователя, преобразует его в ключ DES, которым можно расшифровать билет, и сохраняет билет. Он может понадобиться для подтверждения личности пользователя серверу аутентификации.

Алгоритм Диффи-Хеллмана

Алгоритм Диффи-Хеллмана (Diffe-Hellman) широко используется для обмена ключами. Он позволяет двум абонентам независимо вычислить одинаковые ключи, обмениваясь информацией по незащищенным каналам связи. Алгоритм Диффи-Хеллмана считается стойким при правильном его применении (с соответствующей длинной ключа). Этот алгоритм запатентован в США, но срок патента истек в 1997 году.

Алгоритм KEA

Алгоритм KEA (Key Exchange Algorithm) годится только для обмена ключами, но не для защиты информации. Он основан на модифицированной версии алгоритма Диффи-Хеллмана и использует 1 024-битный ключ.

Протокол SKIP

SKIP (Simple Key Management for Internet Protocols) — это протокол управления ключами, разработанный компанией Sun Microsystems. SKIP легко реализуется. В нем описан способ вычисления ключа на основе сертификатов открытых ключей. Однако использование SKIP налагает определенные ограничения на выбор алгоритмов шифрования и хеширования. Протокол SKIP заявлен как необязательный компонент спецификации IPSec (Internet Protocol Security) — см. главу 3.

Протокол ISAKMP

Протокол ISAKMP (Internet Security Association and Key Management Protocol) подлежит обязательной поддержке в IPSec. По сравнению с протоколом SKIP он позволяет более гибко использовать алгоритмы шифрования и хеширования.

Криптоанализ и атаки

Криптоанализ — это наука о преобразовании шифртекста в открытый текст без знания ключевой информации. Любой шифр надежен настолько, насколько надежно его самое слабое место, будь то выработка ключей или их распределение.

Большинство алгоритмов шифрования основаны на предположениях, кажущихся неоспоримыми, но не имеющих строгого доказательства. Как упоминалось выше, одно из таких предположений таково: поиск делителей большого натурального числа, являющегося произведением двух простых, крайне труден.

С ростом мощности вычислительной техники даже атака полным перебором (описанная в следующем разделе) уже не кажется невозможной.

Это заставляет увеличивать длину ключей, что противоречит интересам правительства США, и в частности ФБР и АНБ (Агентство Национальной Безопасности), желающим иметь возможность дешифровать любое сообщение. Законодательно ограничив максимальную длину ключа, они чувствуют себя очень комфортно.

Полный перебор

Атака полным перебором (brute force attack) заключается в опробовании всех возможных ключей. Но проблема в том, что с ростом длины ключа экспоненциально растет и объем вычислений. Некоторые полагают, что увеличение мощности вычислительной техники сделает шифры ненадежными. Но, с другой стороны, это позволит применять более длинные ключи.

Атака на шифртекст

Атака на шифртекст (cipher-text-only attack) предполагает, что атакующий имеет только шифртекст. На практике же он может располагать некоторыми сведениями о содержании сообщения и использовать их.

Атака по открытому тексту

Атака по открытому тексту (chosen-plain-text attack) предполагает наличие у атакующего возможности зашифровать любой выбранный им текст. Таким образом атакующий пытается вычислить ключ. Шифр RSA уязвим к подобным атакам.

Атака по известному открытому тексту

В этом случае предполагается знание атакующим части или всего открытого текста и соответствующего ему шифртекста. На основании чего он и пытается вычислить ключ.

Атака по времени

Это новый вид атаки. Атакующий замеряет время, необходимое для операции модульного потенцирования (возведения в степень по заданному модулю), и пытается использовать эту информацию. Шифры RSA и Диффи-Хеллмана не стойки против этого типа атак.

Атака «посредник»

Атакующий внедряется между двумя абонентами, обменивающимися информацией. Если это происходит в момент обмена ключами, то он получает возможность дешифровать все последующие сообщения. Чтобы не рассекретить себя сразу, он может длительное время маскироваться, правильно шифруя сообщения и отправляя их адресатам. Наи-

более простой способ защититься от подобных атак — использовать цифровую подпись.

Угадывание ключа

Независимо от стойкости алгоритма, пользователь (или разработчик) способен сам его ослабить, применяя ненадежные ключи, поддающиеся полному или частичному угадыванию. Пример — реализация защиты в бета-версии Netscape Navigator. Здесь часть ключа, используемого для шифрования, основана на текущем времени.

Еще один пример — Microsoft Windows for Workgroups 3.11, применявшая файл паролей. Пользователь регистрировался в системе лишь один раз, а каждый раз, когда ему требовался доступ к ресурсам других компьютеров, ему приходилось вводить разные пароли. Но проблема в том, что эти пароли вместе с именами ресурсов сохранялись в файле, зашифрованном якобы случайным ключом. Проблема первой версии Windows for Workgroups — имя пользователя использовалось как часть ключа. Так атакующий получал значительную часть ключа, что позволяло дешифровать файл паролей. Сейчас эта схема модернизирована.

Криптография и API

Этот раздел описывает API, помогающие программистам добавить средства безопасности в разрабатываемые ими приложения. Иногда трудно провести границу между приложением и API. Отличия зависят от точки зрения. Из предыдущей главы Вы знаете, что протокол SSL — это API для тех, кто программирует на транспортном уровне, но для более высоких уровней — это приложение, обеспечивающее защиту данных.

Защита информации и каналов связи

Средства защиты позволяют обеспечить конфиденциальность информации при передаче по каналу связи. Например, SSL версий 2, 3 и PCT защищают информацию на транспортном уровне. И SSL, и PCT не только криптографические приложения, но и API для программистов, пишущих на транспортном уровне.

Поставщик услуг безопасности в Windows NT

NT Security Service Provider Interface (NT SSPI) — это API, впервые реализованный в Windows NT версии 3.5. Он является «прослойкой» между приложениями и ядром безопасности системы. Это позволяет без труда модифицировать модель безопасности и облегчает разработку приложений. Например, приложение иногда хочет установить защищен-

ное соединение и не желает ничего не знать об используемой схеме аутентификации или оно требует подписывать каждое сообщение. Впервые SSPI появился в LAN Manager и в MSN-диалекте NTLM (для сетей Microsoft), но большинство производителей обращаются к другим поставщикам услуг безопасности.

Криптографический API от Microsoft

В то время как SSPI предоставляет базовый набор криптографических операций, криптографический API (CryptoAPI) от Microsoft решает более широкий спектр задач. CryptoAPI позволяет приложениям шифровать и/или подписывать сообщения, используя различные криптографические методы, а также расшифровывать сообщения и проверять подписи. Методы реализуются посредством *поставщиков криптографических услуг* (cryptography service provider, CSP), подключаемых к CryptoAPI. Это удобно, так как для поддержки нового CSP требуется минимальная модификация приложения.

CryptoAPI версии 1 поставляется с базовым CSP, называющимся «RSA». Он поддерживает:

- 40-битные RC2 и RC4;
- 512-битный RSA;
- MD2 и MD5;
- 160-битный SHA.

В CryptoAPI версии 2 добавлена поддержка сертификатов. Она включает поддержку сертификатов X.509 версии 3, а также PKCS 7 и PKCS 10.

Компания Microsoft намерена сделать CryptoAPI межплатформенным и распространить его не только на Windows-системы. Для этого Microsoft лицензировала CryptoAPI компании RSA и позволила поставлять его в составе своих продуктов.

Microsoft Wallet и PFX

Цифровой бумажник Microsoft Wallet предназначен для надежного хранения личной информации, а PFX (Personal Information Exchange) — для безопасной передачи данных из него. Wallet разработан для хранения конфиденциальной информации, например номера социального страхования, номеров кредитных карт и сертификатов. Первая версия Microsoft Wallet увидела свет в 1996 году, а версия 2.1 в данный момент поставляется в составе Internet Explorer 4.0. Бумажник хранит данные на

компьютере пользователя. Очевидно, стоит ограничить доступ посторонних лиц к нему. Для этого предназначен PFX — черновое описание механизма сохранения информации и извлечения ее из бумажника. Одна из функций PFX — сделать цифровой бумажник мобильным: пользователь должен иметь возможность перенести его с рабочего компьютера на домашний и наоборот. Очевидно, что Microsoft Wallet — криптографическое приложение, тогда как PFX является API. Для защиты паролей тоже разработан API, позволяющий приложению не самому просматривать пароль, а проверяет только его правильность. Описание PFX направлено в W3C (World Wide Web Consortium) в качестве основы будущего стандарта.

Authenticode

С одной стороны, технология Authenticode позволяет клиенту проверить личность человека или организации, от которых ему поступил программный код. С другой — представляет собой API, позволяющий разработчикам или организациям подписывать свой программный код. Также средствами Authenticode клиент может проверить целостность (отсутствие искажений) полученного кода.

Технология Authenticode основана на асимметричном шифровании и использует службу Verisign для получения и управления открытыми ключами субъектов и организаций, подписывающих свой код. Authenticode основана на PKCS 10 (сертификационные запросы), X.509 (спецификация сертификата) и алгоритмах SHA и MD5 (хеширование).

Семейство Java API

Это семейство API до сих пор развивается. Вновь появившиеся Java API поддерживают устаревшие приложения и взаимодействие с базами данных. Интерфейс JDBC (Java Database Connectivity) делает доступными стандартные SQL-средства взаимодействия с базами данных. Java IDL предоставляет объектно-ориентированный интерфейс, Java RMI — средства распределенной обработки между Java-объектами. Java Server API — средство доступа к серверам и разработки так называемых сервлетов (servlet) — апплетов, расширяющих возможности сервера. Java Management API позволяет создавать апплеты для управления корпоративными сетями, Java Media API — мультимедийные приложения. Java Security API предоставляет криптографические средства, защиту данных, цифровые подписи и аутентификацию, JavaBeans API реализует взаимодействие с существующими моделями объектов, например COM, CORBA, OpenDOC и OLE.

Ссылки

http://home.netscape.com/assist/security/ssl/index.html (информация о SSL)

http://www.microsoft.com/security

http://www.rsa.com/rsalabs/pubs/PKCS

RFC 2315, «PKCS 7: Cryptographic Message Syntax Version 1—5»

RFC 2314, «PKCS 10: Certification Request Syntax Version 1—5»

RFC 2313, «PKCS 1: RSA Encryption Version 1—5»

RFC 2268, «A Description of the RC2(r) Encryption Algorithm»

RFC 2144, «The CAST-128 Encryption Algorithm»

RFC 1984, «IAB and IESG Statement on Cryptographic Technology and the Internet»

RFC 1848, «Mime Object Security Services»

RFC 1704, «On Internet Authentication»

RFC 1511, «Common Authentication Technology Overview»

RFC 1411, «Telnet Authentication: Kerberos Version 4»

RFC 1321, «The MD5 Message-Digest Algorithm»

RFC 1320, «The MD4 Message-Digest Algorithm»

RFC 1319, «The MD2 Message-Digest Algorithm»

Часть 2

Поиск информации

ГЛАВА 6

Службы каталогов

Службы каталогов (directory service) похожи на телефонные книги и справочники «желтые страницы». По телефонной книге каждый может отыскать информацию, например, о ресторанах индийской кухни в городе или адрес кафе по его названию. Используя службы каталогов, компьютерные программы ищут объекты по их атрибутам, таким, как уровень доступа, членство в доменах и группах, способы аутентификации, сетевые адреса и адреса портов. Для управления огромными объемами информации службу каталогов обычно реализуют в виде базы данных, хранящейся на отдельном сервере.

Рис. 6-1 иллюстрирует стандартную реализацию службы каталогов. Клиент подключается к службе каталогов для просмотра или обновления ее базы данных. Некоторые службы каталогов способны обмениваться информацией с аналогичными службами.

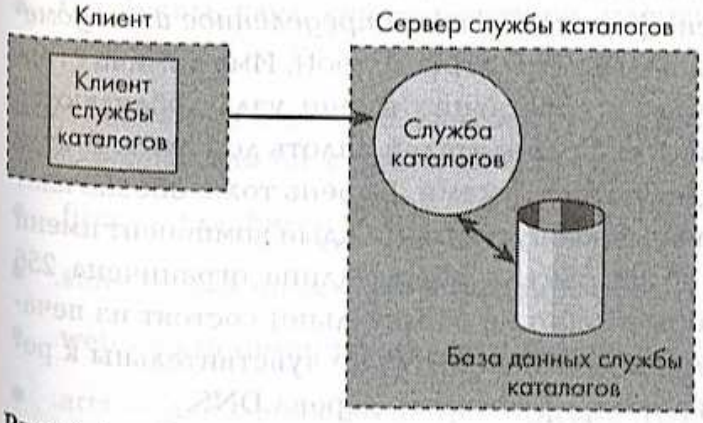


Рис. 6-1. *Стандартная реализация службы каталогов*

Службы каталогов, как правило, выполняют преобразование мнемонических имен (легко запоминаемых) в IP-адреса (трудные для запоминания человеком) и обратно. Это называется *служба имен* (name service) — одна из множества услуг, предоставляемых службой каталогов. Самая популярная служба имен — Система именования доменов (Domain Name System, DNS), описанная в следующем разделе.

Система именования доменов (DNS)

DNS (Domain Name System) — это иерархическая служба имен, созданная для более эффективного использования огромного *пространства имен* (name space) — множества всех возможных имен, доступных посредством службы имен. Пространство имен декларирует соглашения и синтаксис именования объектов.

Основное назначение DNS — трансляция имен и IP-адресов. Пока это наиболее популярная функция, хотя DNS обеспечивает и такие услуги, как хранение и предоставление информации о пользователях и списках рассылки и упрощение обмена почтой.

Основные понятия, определяемые DNS, — пространство имен, распределенная база данных и протокол обмена информацией. Все они описаны далее. Важно понимать, что, хотя термин *DNS* обычно обозначает сервер, сам механизм трансляции применяет модель «клиент-сервер». Допустим, клиент желает установить соединение с другим компьютером для обмена электронной почтой или загрузки файла. Клиенту может быть известно только имя этого компьютера, но не его IP-адрес. Трансляцию осуществляет ПО клиента, запрашивающее у DNS-сервера необходимую информацию. Отдельный компьютер способен одновременно выступать в качестве DNS-сервера и DNS-клиента.

Службу каталогов, или имен, традиционно представляют в виде дерева, что также верно и для DNS. Каждый элемент дерева DNS — *узел* (node) — имеет уникальное *имя домена* (или *полностью определенное имя домена*). Узел на вершине дерева называется *корнем* (root). Имя домена строится по следующему правилу: к собственно имени узла добавляются имена родительского узла и всех прародителей вплоть до корня. В имени домена имена узлов разделяются точками. Корень тоже обозначают точкой, но и в имени домена обычно опускают. Каждый компонент имени домена может составлять до 63 байт, а общая длина ограничена 256 байтами. Все компоненты обычно (но не обязательно) состоят из печатаемых ASCII-символов. Имена доменов в DNS не чувствительны к регистру символов. Рис. 6-2 иллюстрирует часть дерева DNS.

На этом рисунке *Redmond.Warehouse.BigCompany.com* — имя домена. Заметьте, что такие имена синтаксически отличаются от полного пути файла, например *\\Server\Share\Directory1\File1*. Здесь элементы, наиболее удаленные от корня, расположены в конце строки, тогда как в имени домена — в начале.

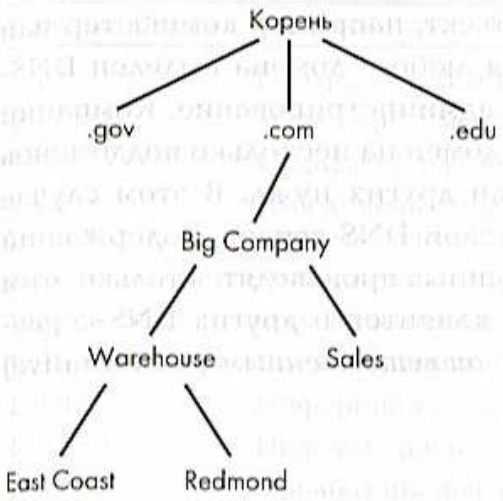


Рис. 6-2. *Фрагмент дерева DNS*

Некоторые DNS-имена (или домены) второго уровня зарезервированы и обозначают:

- .mil — вооруженные силы США;
- .gov — правительственные учреждения;
- .com — коммерческие организации;
- .net — поставщиков сетевых услуг;
- .edu — образовательных учреждений;
- две буквы, следующие за точкой, идентифицируют страну, например: .ru, .su — Россия, .us — США, .ca — Канада.

Когда писалась эта книга, еще семь имен доменов второго уровня были предложены для внесения в DNS:

- .firm — для фирм;
- .store — для продавцов товаров и сферы услуг;
- .web — для организаций, работающих в WWW;
- .arts — для сферы культуры и искусства;
- .rec — для сферы развлечений и отдыха;
- .info — для поставщиков информационных услуг;
- .nom — для индивидуальных пользователей.

Как указывалось ранее, пространство имен DNS — это просто набор или дерево имен доменов и правила создания этих имен. Каждый узел

в имени домена представляет некий объект, например компьютер или псевдоним для электронной почты. Для любого домена выделен DNS-сервер, с которого осуществляется его администрирование. Компании или крупные сети вправе подразделять домен на несколько поддоменов для упрощения администрирования или других нужд. В этом случае каждый из поддоменов должен иметь свой DNS-сервер, содержащий его базу данных. Все изменения в базе данных производятся только этим сервером, и он обрабатывает запросы клиентов и других DNS-серверов. Сервер, хранящий базу, называют *ответственным* (authoritative) сервером этого домена или поддомена.

Описатели ресурсов

Как уже упоминалось ранее, помимо прочего DNS — распределенная база данных. Она размещается на серверах, управляющих различными доменами и поддоменами. Элемент этой базы данных — *описатель ресурса* (resource record) — содержит IP-адрес, имя почтового сервера домена или другую информацию. Каждый узел в DNS задается одним или несколькими описателями ресурса. Формат описателя ресурса приведен в RFC 1183 и 1035.

Поля описателя ресурса показаны на рис. 6-3.

Рис. 6-3. *Поля описателя ресурсов*

Поле **Имя** (Name) содержит имя домена, которому принадлежит этот описатель ресурса. Поле **Тип** (Type) — 16-разрядное значение, определяющее тип ресурса. Например, описатель ресурса типа A содержит IP-адрес, типа MX — имя почтового сервера и т. д. В таблице перечислены некоторые значения поля **Тип**.

Поле Тип (Type)	Описание
A	IP-адрес
NS	Ответственный сервер имен
MD	Приемник почты (устарело, заменено на MX)

Поле Тип (Type)	Описание
MF	Почтовый шлюз (устарело, заменено на MX)
CNAME	Настоящее имя псевдонима
SOA	Индикатор начала полномочий в зоне
MB	Имя домена, владеющего почтовым ящиком (экспериментальный)
MG	Член почтовой группы (экспериментальный)
WKS	Описание общеизвестной службы
PTR	Указатель на имя домена
HINFO	Информация о хосте
MINFO	Информация о почтовом ящике или списке рассылки
MX	Обработчик почты
TXT	Текстовая строка

16-разрядное поле **Класс** (Class) содержит описание семейства протоколов, соответствующего данному типу описателя ресурса. Например, IN — для Интернета, CH — для Chaos и т. д. На практике используется только IN. Поле **TTL** (Time To Live — «время жизни») содержит 32-разрядное число со знаком, определяющее время в секундах, в течение которого описатель ресурса должен храниться в кэше. Поле **Длина данных** (RDLength) определяет длину в байтах поля **Данные** (RData). Поле **Данные** имеет произвольную длину в зависимости от типа описателя ресурса. Например, для описателя ресурса типа A поле **Данные** содержит 32-разрядный IP-адрес.

Для повышения эффективности DNS-серверы обычно кэшируют описатели ресурсов, полученные от других DNS-серверов. Поле **TTL** определяет время в секундах, по истечении которого DNS-сервер обновляет копию описателя ресурса. Значение 0 свидетельствует, что описатель ресурса будет использован единожды и не попадет в кэш.

Запросы и анализаторы

Клиент обращается к DNS-серверу посредством *анализатора* (resolver). Обычно это программный модуль, выполняющийся на DNS-клиенте и обслуживающий DNS-запросы. Анализатор посылает DNS-серверу запросы (от имени клиента) и кэширует ответы. Каждый запрос отправляется посредством датаграмм протокола UDP (User Datagram Protocol), описанного в RFC 768. Клиент может выдавать два типа запросов: *рекурсивные* (recursive) и *итеративные* (iterative). В первом случае DNS-сервер, если не в состоянии самостоятельно удовлетворить запрос клиента, обращается к другому DNS-серверу, если и второй не справляется, он обращается к третьему и так далее, пока запрос не будет удовлет-

ворен. То есть используется рекурсия. Но в большинстве случаев клиенты применяют итеративные запросы: если необходимая информация на запрашиваемом сервере отсутствует, то он отправляет ссылку на другие серверы, возможно ее имеющие. Итеративные запросы меньше нагружают DNS-серверы. Анализаторы обычно кэшируют ответы DNS-серверов — это повышает производительность и снижает нагрузку на DNS-серверы и сеть. Напомню, что описатель ресурса, возвращаемый DNS-сервером, содержит поле **TTL**, указывающее время, в течение которого кэшированная запись считается истинной.

Анализатор должен осторожно обрабатывать простые ошибки, возникающие из-за проблем в сети. Агрессивный механизм повторов только увеличивает трафик в сети, уже имеющей проблемы. С другой стороны, если повторная попытка установить связь с сервером увенчается успехом, анализатор вправе прекратить сообщать об ошибке.

DNS-запрос содержит три поля:

- имя домена, о котором необходимо получить информацию;
- тип ожидаемого описателя ресурса — это один из предопределенных типов, либо звездочка (*), означающая, что могут быть возвращены описатели любого типа, либо MAILB, означающее, что должен быть возвращен описатель ресурса для почтового ящика;
- класс ожидаемого описателя ресурса — один из предопределенных классов или звездочка (*), означающая любой класс.

Ответ на запрос обычно имеет одну из трех форм:

- ошибка — запрашиваемое имя не существует;
- простая ошибка — информация временно недоступна, но, вероятно, станет доступной позднее;
- один или несколько описателей ресурсов, отвечающих критерию запроса.

Зоны DNS

Для удобства пространство имен DNS разделено на *зоны*, которые создаются путем логического уничтожения связи между узлом и его родителем. Полученное изолированное поддерево и называют зоной. Сервер несет полную ответственность за свою зону. Когда сервер в зоне получает запрос, он либо возвращает требуемую информацию, либо ссылается на сервер, ответственный за поддомен, к которому относится

запрос. На рис. 4-6 изображено гипотетическое пространство имен, разделенное на зоны.

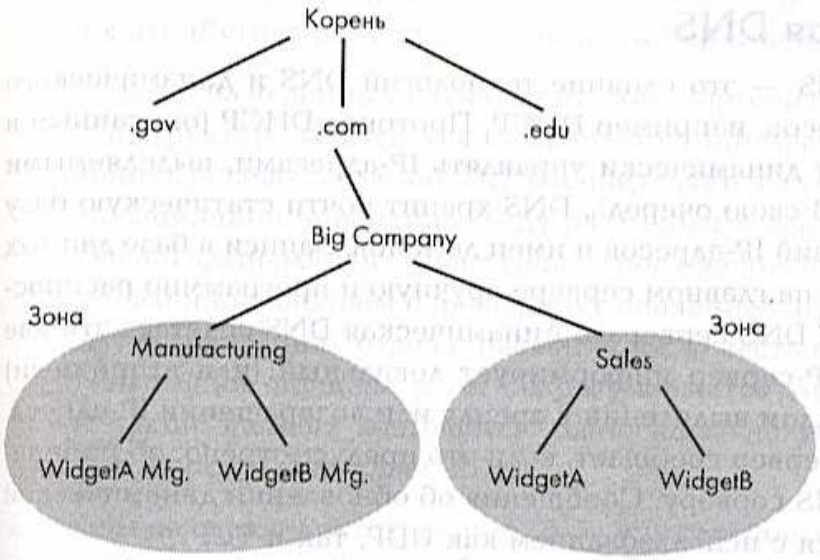


Рис. 6-4. *Гипотетическое пространство имен DNS, разделенное на зоны*

Тиражирование DNS

Каждая зона DNS имеет свой сервер имен. Для надежности сервер тиражируется как минимум дважды. Один сервер назначается *главным* (master), остальные — *подчиненными* (slave). Все операции обновления проводятся только на главном сервере, а подчиненные периодически опрашивают главный сервер для получения обновленной информации. Такое обновление называется *пересылкой зоны* (zone transfer). Взаимоотношение «главный-подчиненный» ограничивается пересылкой зоны, а по отношению к запросам (от анализаторов) все серверы одной зоны равноправны.

Транспортные протоколы DNS

Реализация DNS обязана уметь обрабатывать запросы как по протоколу UDP, так и по TCP. Нельзя удалять поддержку UDP даже в том случае, если поддерживается протокол TCP. Для выполнения запросов предпочтительнее использовать TCP, особенно в тех случаях, когда возвращаются большие объемы информации. С другой стороны, преимущество UDP — уменьшение затрат. Для запросов как по UDP, так и по TCP используется порт 53. На практике пересылка зон производятся по протоколу TCP, тогда как анализаторы применяют UDP. Некоторые клиенты для обнаружения сервера пытаются использовать широкове-

щание или групповое вещание по протоколу IP. DNS-серверы могут (но не обязаны) поддерживать и такие запросы.

Динамическая DNS

Динамическая DNS — это слияние технологий DNS и динамического выделения IP-адресов, например DHCP. Протокол DHCP (описанный в главе 3) позволяет динамически управлять IP-адресами, выделяемыми DHCP-клиентам. В свою очередь, DNS хранит почти статическую базу данных соответствий IP-адресов и имен доменов. Записи в базе данных DNS обновляются на главном сервере вручную и программно распространяются другим DNS-серверам. Динамическая DNS сочетает эти две технологии: DHCP-сервер информирует локальный (или вторичный) DNS-сервер о каждом выделении в аренду или возвращении IP-адреса. Вторичный DNS-сервер сообщает, если это предусмотрено, об изменениях главному DNS-серверу. Сообщения об обновлении динамической DNS пересылаются с использованием как UDP, так и TCP.

Еще одно отличие динамической DNS — направление информационных потоков. В DNS информация движется от главного DNS-сервера к подчиненным. В динамической DNS направление потока обратное — от подчиненного к главному. Динамическая DNS описана в RFC 2136.

Спецификации X.500

В 1988 году OSI определила набор спецификаций X.500. В 1993 году они были дополнены. OSI ставила перед собой задачу, ни много ни мало, создать глобальную службу каталогов. Но рыночные реалии оказались суровы, в результате X.500 не получила широкого распространения. Среди причин этой относительной неудачи — сложность протоколов, высокие требования к ресурсам, необходимым для реализации и работы протоколов, недостаточная проработка бизнес-плана внедрения и неудовлетворение потенциальных пользователей от того, что защищенная информация может стать общедоступной.

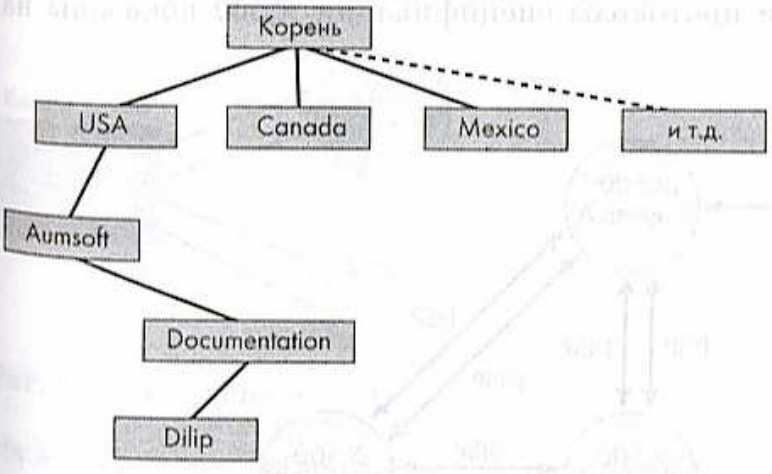
Спецификация X.500 описывает модель хранения информации в каталоге. Согласно X.500, элементы каталога хранятся в базе данных службы каталогов — DIB (Directory Information Base). Любой элемент описывается набором атрибутов, каждый из которых имеет тип и значение. Тип атрибута определяет его синтаксис — вид информации, служащей его значением. Атрибут может иметь множественное значение. Элемент характеризуется произвольным числом атрибутов, причем некоторые обязательные, а другие — нет. Для каждого элемента задан атрибут

objectClass (класс объекта), значение которого и определяет, какие атрибуты обязательны, а какие нет. Примеры значений атрибута objectClass: Личность (Person), Штат (State), Страна (Country) и т. д. Примеры других атрибутов: фамилия, название, номер отдела и т. д.

Элементы в базе данных службы каталогов организованы иерархически. Эта структура называется DIT (Directory Information Tree — информационное дерево каталога). DIT распределяется в географической или организационной иерархии между различными серверами службы каталогов. Каждый элемент имеет *уникальное имя* (distinguished name, DN), представленное набором пар «атрибут-значение», идентифицирующих страну, организацию, отдел и название. Если часть информации в уникальном имени опущена (т. е. подразумеваются значения по умолчанию), такой элемент называется *относительно уникальным именем* (relative distinguished name, RDN).

В примере, показанном на рис. 6-5, уникальное имя выглядит так: /c=USA, /o=Aumsoft, /ou=Documentation, /cn=Dilip.

X.500 требует, чтобы клиент связался со службой каталогов, прежде чем он сможет что-либо сделать. Эту операцию называют *связывание* (bind), она необходима, чтобы клиент предъявил серверу «удостоверение личности». Допустимая степень защиты варьируется от паролей, передаваемых открытым текстом, до применения криптографии с открытым ключом. Для завершения сеанса клиент выполняет «отвязывание» (unbind).



Информационное дерево каталога

Рис. 6-5. *Пример уникального имени — /c=USA, /o=Aumsoft, /ou=Documentation, /cn=Dilip*

Когда клиент установит сеанс, он получит доступ к операциям поиска (search), чтения (read), модификации (modify), добавления (add), удаления (delete) и отмены (abandon). Поиск возвращает элементы, отвечающие заданному критерию — *фильтру поиска*. Результатом этой операции может быть элемент, набор элементов или целое поддерево. Чтение просто возвращает атрибуты указанного элемента, модификация — изменяет существующие элементы. Для модификации имени элемента в спецификацию X.500 включена операция modifyRDN. В X.500 определены операции добавления и удаления элементов, а также процедура отмены еще незавершенной операции.

Каждую операцию и ее результат можно подписать с использованием открытого ключа клиента или сервера.

Приложения получают доступ к информации службы каталогов, как и DNS-клиент, посредством клиентского интерфейса, именуемого *Пользовательским агентом службы каталогов* (Directory User Agent, DUA). Он обращается к серверу службы каталогов по DAP (Directory Access Protocol) — протоколу прикладного уровня, описывающему использование транспортного стека OSI. Реализация транспортного стека OSI требует значительных затрат, причем не только на разработку ПО, но и на оборудование для работы стека протоколов. Различные серверы службы каталогов способны обмениваться информацией посредством протокола DSP (Directory System Protocol), описанного далее. Спецификация X.500 1993 года определяет протокол DISP (Directory Information Shadowing Protocol) — метод тиражирования данных между серверами X.500. Различные компоненты и протоколы спецификации X.500 показаны на рис. 6-6.

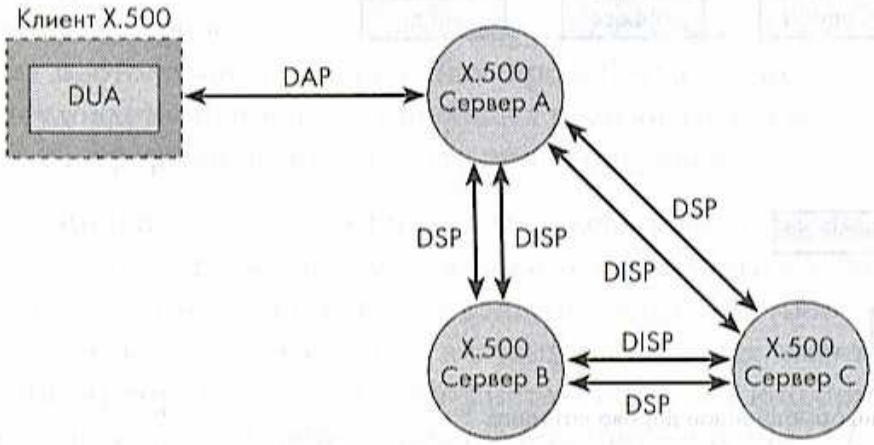


Рис. 6-6. *Компоненты и протоколы спецификации X.500*

Тиражирование позволяет равномерно распределить нагрузку между несколькими серверами, что повышает степень надежности и эффективность работы. Кроме того, тиражирование используют и для перемещения информации на более доступный сервер. В X.500 сказано, что клиент должен видеть всю информацию, независимо от того, к какому серверу каталогов он обращается. Сервер, не имеющий нужной информации, реализует описанное требование посредством *цепочек* (chaining), когда он сам связывается с другим сервером, или *ссылок* (referral), когда просит клиента сделать это самостоятельно. Цепочки реализуются средствами протокола DSP. Хотя последний используется только для взаимодействия между серверами X.500, само взаимодействие — результат запроса от DUA. Рис. 6-7 иллюстрирует концепцию цепочек.

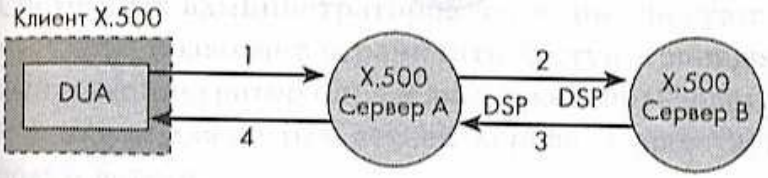


Рис. 6-7. *Концепция цепочек*

На этом рисунке (и на двух последующих) номера стрелок обозначают последовательность обмена информацией. Сначала клиент X.500 посылает запрос серверу A. Сервер A не имеет запрашиваемой информации и обращается к серверу B. Последний передает информацию серверу A, а тот — клиенту.

Рис. 6-8 иллюстрирует концепцию ссылок.

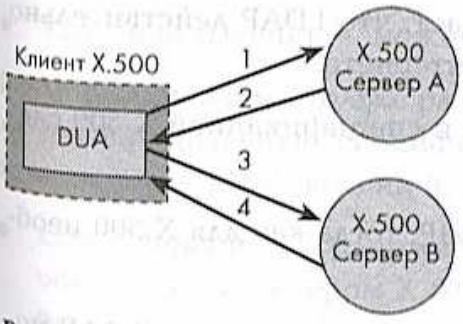


Рис. 6-8. *Концепция ссылок*

На этом рисунке клиент X.500 запрашивает сервер A. Тот не имеет запрашиваемой информации, о чем сообщает клиенту. Ответ, по сути, содержит следующую информацию: «Я (сервер A) не обладаю запрашиваемой информацией. Попробуй обратиться к серверу B». Клиент направляет запрос к серверу B и получает необходимые сведения.

На рис. 6-9 показана концепция *группового вещания* в X.500 (multicasting), очень похожая на цепочки: одновременно информация запрашивается более чем у одного сервера. Каждый сервер возвращает либо ответ, либо сообщение об ошибке, если он не в состоянии удовлетворить запрос. Клиент обрабатывает единственный ответ.

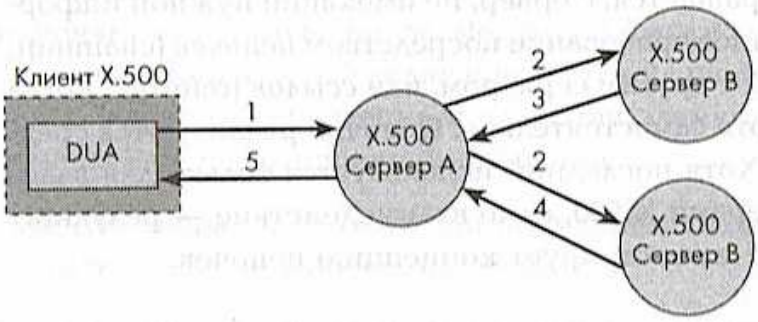


Рис. 6-9. *Групповое вещание*

Протокол LDAP

Протокол LDAP (Lightweight Directory Access Protocol) в настоящее время является открытым стандартом Интернета и описывает доступ к службам каталогов. Основа LDAP — протокол DAP спецификации X.500. Впервые LDAP описан в RFC 1487. Текущий стандарт — LDAP версии 2 — описан в RFC 1777, заменившим RFC 1487. Протокол LDAP призван упростить реализации некоторых элементов спецификации X.500. Наиболее известная реализация LDAP создана в Мичиганском университете (University of Michigan). В последнее время многие компании, включая Microsoft и Netscape, выразили намерение использовать LDAP в своих коммерческих продуктах. Это подтверждает, что LDAP действительно стал стандартом для служб каталогов в Интернете.

Основные отличия между LDAP версии 2 и спецификацией X.500 следующие:

- LDAP работает поверх протокола TCP/IP, тогда как для X.500 необходим стек протоколов OSI;
- в LDAP упрощена процедура связывания (bind). Клиенты LDAP могут устанавливать соединения анонимно (т. е. без ввода пароля) или отправлять пароль открытым текстом. X.500 использует более стойкие криптографические механизмы;
- LDAP не имеет команд чтения (read) и просмотра (list). Клиент добьется того же, соответствующим образом используя команду поиска (search);

- клиент LDAP не может устанавливать соединения с несколькими серверами одновременно. Сервер LDAP не вправе отослать клиента к другому серверу;
- LDAP использует более простое, по сравнению с X.500, кодирование данных.

В Мичиганском университете разработана библиотека на языке C, реализующая клиент LDAP. Она описана в информационном RFC 1823. Эта библиотека позволяет, используя лишь простые вызовы функций с текстовыми параметрами, написать программу на языке C для отправки LDAP-запросов на поиск или чтение (подробнее о командах — в разделе «Спецификации X.500»).

Системных администраторов часто интересуют вопросы безопасности. LDAP позволяет ограничить доступ к каждому элементу: системный администратор определяет права пользователя как для отдельных DN, так и для целых ветвей дерева. Существуют следующие права пользователя:

- Нет (None) — нет доступа к информации;
- Сравнение (Compare) — можно сравнивать значения DN;
- Поиск (Search) — можно осуществлять поиск DN и получать информацию о его существовании;
- Чтение (Read) — можно считывать значения атрибутов DN;
- Запись (Write) — можно изменять значения атрибутов DN;
- Удаление (Delete) — можно удалить DN.

LDAP версии 3 пока только разрабатывается. IETF еще не зафиксировала его официальную спецификацию. Ниже перечислены ожидаемые возможности LDAP версии 3.

- LDAP версии 3 поддерживает модель, в которой сервер LDAP не обязан быть сервером X.500. В версии 2 предполагалось, что клиент посылает запрос с использованием протокола LDAP, после чего он преобразуется в DAP-запрос и только затем передается серверу. Новая модель (версии 3) не требует преобразования между LDAP и DAP. Обе модели изображены на рис. 6-10.
- Теперь сервер LDAP может отсылать клиента к другому серверу.

- Спецификация LDAP версии 2 описывала классы объектов и атрибуты как составную часть протокола, поэтому службы каталогов не масштабировались. В LDAP версии 3 клиент может послать запрос серверу, чтобы узнать набор допустимых классов объектов и атрибутов. Это означает, что по мере необходимости можно вводить новые атрибуты, и LDAP-клиенты сумеют определить их существование.
- LDAP версии 3 допускает представление символов в кодировке Unicode, что увеличивает гибкость и обеспечивает поддержку разных языков (интернациональность).
- LDAP версии 3 позволяет использовать сертификаты формата X.509 для усиления безопасности. X.509 — это спецификация, описывающая форматы сертификатов. Также в версии 3 предусмотрен механизм использования SSL.
- В LDAP версии 3 определен протокол Connectionless LDAP (CLDAP, LDAP без установки соединений). Это вариант LDAP ориентирован на приложения, которым необходимо отправлять простые запросы и получать быстрые ответы. Для передачи запросов и ответов CLDAP использует транспортный механизм UDP.

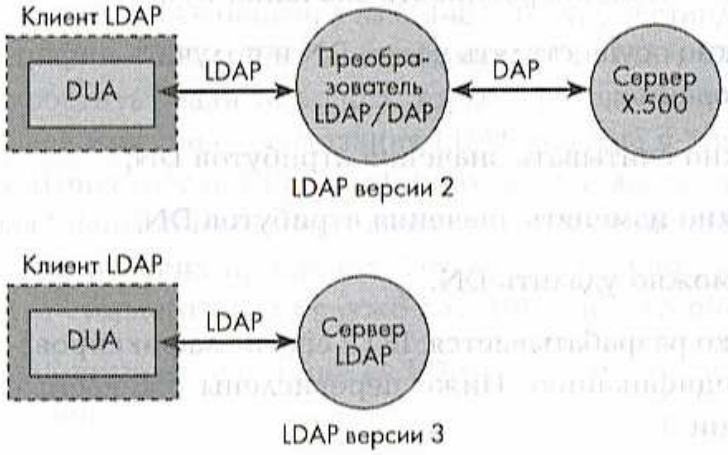


Рис. 6-10. *Модели LDAP*

Ссылки

DNS

RFC 2136, «Dynamic Updates in the Domain Name System»

RFC 1591, «Domain Name System and Delegation»

RFC 1480, «The U.S. Domain»

RFC 1183, «New DNS Resource Record Definitions»

RFC 1101, «DNS Encoding of Network Names and Other Types»

RFC 1035, «Domain Names—Implementations and Specification»

RFC 1034, «Domain Names—Concepts and Facilities»

RFC 1032, «Domain Administrator's Guide»

RFC 974, «Mail Routing and the Domain System»

RFC 920, «Domain Requirements»

X.500

RFC 1309, «Technical Overview of Directory Services Using the X.500 Protocol»

RFC 1308, «Executive Introduction to Directory Services Using the X.500 Protocol»

RFC 1279, «X.500 and Domains»

LDAP

RFC 2256, «A Summary of the X.500(96) User Schema for Use with LDAPv3»

RFC 2255, «The LDAP URL Format»

RFC 2254, «The String Representation of LDAP Search Filters»

RFC 2253, «Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names»

RFC 2252, «Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions»

RFC 2251, «Lightweight Directory Access Protocol (v3) »

RFC 1960, «A String Representation of LDAP Search Filters»

RFC 1959, «An LDAP URL Format»

RFC 1823, «The LDAP Application Program Interface»

RFC 1798, «Connectionless Lightweight X.500 Directory Access Protocol»

RFC 1777, «Lightweight Directory Access Protocol»

RFC 1487, «X.500 Lightweight Directory Access Protocol»

ГЛАВА 7

Настройка и управление

В этой главе описаны некоторые популярные инструментальные средства настройки и управления сетями — но только те, что по статусу близки протоколам. Так, например, говоря о сетях, нельзя не упомянуть о средствах отображения таблиц маршрутизации, но здесь мы их не обсуждаем, поскольку с ними не связаны никакие протоколы. (Протоколы маршрутизации описаны в главе 2.) Более полный список инструментов Вы найдете в RFC 1470, а также в RFC 1574.

Эхо

Протокол передачи *эха* (echo) используется для выявления утери пакетов в сетевом оборудовании и серверах. Этот диагностический протокол поддерживается целым рядом сетевых устройств (хостами, маршрутизаторами, серверами и т. д.). Передача эха состоит из двух этапов: отправки запроса от локальной системы и получение ответа от удаленной. Пакет *эхо-запрос* (ERQ, код типа равен 30) позволяет выявить неисправности удаленной системы: о ее нормальной работе свидетельствует *эхо-ответ* (ERP, код типа равен 31), в котором содержится исходный эхо-запрос для возможной дальнейшей обработки. Подробнее протокол передачи эха описан в RFC 862 и 1575.

Ping

Ping — это способ определения доступности удаленного сетевого ресурса и скорости связи с ним, а также самое простое средство выявления неполадок сети. Иногда слово «Ping» расшифровывают как «Packet Internet Groper».

Этот механизм реализуется двумя способами: посредством эхо-запросов протокола ICMP или через протокол передачи эха (упомянутый выше) поверх TCP или UDP. Если эхо-запрос и эхо-ответ проходят нормально, то Ping считается успешным.

Обычно Ping-утилита, направляет серверу несколько пакетов, а затем оценивает время задержки ответов и процентное соотношение прошедших и потерянных пакетов. Еще одно полезное свойство такой утилиты — способность отображать буфер маршрутизации прибывающих пакетов, что позволяет выявлять проблемы задействованных сетевых ресурсов. Также Ping-утилиты позволяют явно указывать размер пакетов, их количество и длительность паузы перед отправкой следующего пакета.

Подробно Ping описан в RFC 1574.

Трассировка маршрута

Другим средством отладки сети является *трассировка маршрута* (trace route). Она похожа на Ping, но отражает более полную информацию о маршруте следования пакетов.

Прежде чем попасть к адресату, пакет проходит через ряд маршрутизаторов и магистралей, причем обратный маршрут иногда не совпадает с прямым. Утилита трассировки отображает все узлы, через которые проследовал пакет на пути от отправителя до получателя. Это позволяет выяснить, как Ваше подключение обслуживается на участке от локального поставщика услуг Интернета, через магистрали и до системы адресата.

Одна из реализаций трассировки маршрута использует последовательности пакетов с возрастающими значениями в поле **TTL** («время жизни»). Узел, на котором значение **TTL** окажется равным 0 (а при прохождении каждого узла это значение уменьшается на 1), обязан отослать пакет обратно отправителю. Далее значение **TTL** будет увеличено и попытка повторена и так до тех пор, пока пакеты не дойдут до адресата. Этот метод достаточно прост и популярен, хотя при его применении генерируется много пакетов и нельзя отследить их обратный маршрут. Другой способ реализации трассировки маршрутов использует соответствующую функцию протокола IP, которую можно сочетать с эхо-запросами протокола ICMP.

Обычно утилиты трассировки маршрута позволяют явно задавать время ожидания ответов, количество посылаемых пакетов и максимальное число переходов, по достижении которого следует прекратить операцию.

Применяя трассировку маршрутов, имейте в виду, что разные протоколы предъявляют различные требования к пропускной способности ка-

налов. Например, NNTP (Network News Transfer Protocol) допускает небольшие задержки, а протоколы магистралей группового вещания, например MBone (multicast backbone), — нет, поэтому увеличение числа промежуточных узлов или долгое ожидание может вызвать скрытые проблемы в сети и сделать невозможным использование таких протоколов.

Протокол идентификации

Протокол идентификации (Identification Protocol, Ident), описанный в RFC 1413, предоставляет для протоколов, основанных на TCP, базовые средства идентификации пользователей. Прежде его называли протоколом сервера аутентификации (Authentication Server Protocol), но затем переименовали, дабы название точнее отражало его предназначение — идентификацию пользователей.

Часто протокол Ident используется серверами для усовершенствования журналов регистрации или механизмов идентификации. При наличии другого TCP-протокола, например Telnet, SMTP (Simple Mail Transfer Protocol), FTP (File Transfer Protocol), IRC (Internet Relay Chat), NNTP или HTTP (Hyper Text Transfer Protocol), сервер может применять Ident для идентификации пользователей. Это помогает предотвратить некоторые неприятности, связанные с несанкционированным использованием учетных записей.

Протокол Ident ориентирован на установление логического соединения и задействует 113-й порт TCP. Сервер протокола Ident (в UNIX-системах его часто называют identd) запрашивает строку, идентифицирующую пользователя данного клиента. Эту информацию можно внести в журнал регистрации активности пользователей или на ее основе отказать в подключении тем, кто нарушает правила регистрации (например, задает поддельный адрес электронной почты). Наличие идентификационной информации не позволяет отказаться от применения протоколов аутентификации гарантированной стойкости, поскольку эту информацию могут несанкционированно применять достаточно осведомленные пользователи.

Две наиболее частые причины отказов протокола Ident — неверные номера портов TCP и неопознанные пользователи (процессы, не имеющие опознаваемых пользователей, или пользователи, скрывающие информацию о себе).

Whois и Whois++

Whois — это протокол поиска информации о сетях, доменах и организациях. Служба Whois выполняется на многих серверах Интернета. Клиенты Whois посылают серверам запросы. В ответ они получают информацию об объекте, например имя сервера, краткие сведения об организации, информацию об администраторах и способе связи с ними, а также IP-адрес. Протокол Whois напоминает Finger, отличие лишь в том, что последний поддерживают практически все сервера, а первый — только некоторые. Считается, что Whois применяют лишь для простых запросов, но не для расширенного поиска.

В середине 1992 года протокол Whois был доработан и получил название Whois++. Новый протокол не совместим со старым. Основные отличия в том, что Whois++ структурирует принятую от сервера информацию и допускает взаимодействие между серверами.

На серверах Whois++ информация хранится в виде записей базы данных. Каждая запись состоит из нескольких полей, представляющих собой пары «атрибут/значение». Пример такой пары — Email:hacker@atlarge.com. Начало и конец каждой записи выделяются специальными ключевыми словами — #FULL и #END. После #FULL обычно следует описатель шаблона записи. Например, ключевое слово USER свидетельствуют, что запись содержит информацию о пользователе.

Взаимодействие серверов Whois++ напоминает репликацию базы данных. Данные, которыми обмениваются между собой серверы, называются *ссылками*. Обмен ссылками хорош тем, что при обращении клиента к одному серверу тот сможет вернуть ссылку (указатель) на сервер, владеющий запрашиваемой информацией. Взаимодействие между серверами выполняется по протоколу CIP (Common Indexing Protocol), который определен в IETF рабочей группой Find. (Подробнее об этом см. *http://www.ietf.cnri.reston.va.us/html.charters/find-charter.html*.) В IETF этой тематикой занимаются еще две рабочих группы — Access, Searching, and Indexing of Directories (ASID, *http://www.ietf.cnri.reston.va.us/html.charters/asid-charter.html*) и Integrated Directory Service (IDS, *http://www.ietf.cnri.reston.va.us/html.charters/ids-charter.html*).

Протокол CIP позволяет серверам обмениваться порциями сжатой информации, или *центроидами* (centroids). Центроиды содержат всю имеющуюся на сервере информацию, но в них отсутствуют некоторые связи, например ссылки на конкретные записи с данными. В результате на запрос иногда приходит «ложный положительный ответ» — сервер воз-

вратит ссылку, но когда Вы проследуете по ней, то не обнаружите необходимых данных. Но сервер никогда не сообщит об отсутствии информации, если она на самом деле имеется (невозможность «ложного отрицательного ответа»).

Ссылки

RFC 1914, «How to Interact with a Whois++ Mesh»

RFC 1913, «Architecture of the Whois++ Index Service»

RFC 1835, «Architecture of the Whois++ service»

RFC 1834, «Whois and Network Information Lookup Service; Whois++»

RFC 1575, «Echo Function for CLNP»

RFC 1574, «Essential Tools for the OSI Internet»

RFC 1470, «FYI on a Network Management Tool Catalog: Tools for Monitoring and Debugging TCP/IP Internets and Interconnected Devices»

RFC 1413, «Identification Protocol»

RFC 1393, «Traceroute Using an IP Option»

RFC 954, «NICNAME/WHOIS»

RFC 862, «Echo Protocol»

ГЛАВА 8

Поиск

Одна из важнейших проблем нашего времени — необходимость «прочесывать» все существующие информационные ресурсы, чтобы добраться до нужных. С появлением Всемирной Паутины (World Wide Web, WWW) доступ к интерактивным данным стал проще, но одновременно их объем несказанно увеличился. Ориентироваться во всем этом информационном изобилии по-прежнему нелегко.

Поначалу путешествие по Web было возможно только через *страницы ссылок* (link pages) — списки гиперссылок на другие страницы. Позже появились узлы — предметные указатели, например Yahoo!, содержащие огромное количество гиперссылок, упорядоченных по темам. Иногда предметные указатели представляют собой печатные издания. Тем не менее они охватывают не весь объем данных, а лишь ту его часть, которая известна издателям на момент публикации.

Положение спасают *поисковые машины* (search engines). В основном это серверы, хранящие сведения об огромном количестве информационных ресурсов (часто эти сведения представляют собой полный текст исходного документа). Для обнаружения определенных сведений Вам достаточно ввести ключевое слово.

Самые первые методы поиска и передачи файлов

Раньше, до широкого распространения в Интернете Web-протоколов, для обнаружения и передачи документов использовались самые разные методы. В этом разделе мы их и обсудим.

Archie

До появления Web доступ к файлам осуществлялся в основном через FTP-серверы (File Transfer Protocol). Для этого надо было знать имена FTP-серверов (которых очень много) и характеристики информации

(типы файлов), хранящейся на каждом сервере. Пользователям было трудно работать с таким количеством серверов.

Информационная система Archie предназначена для организации сведений о документах, хранящихся на FTP-серверах. Сначала Archie-сервер создает перечень информации об известных ему файлах и ресурсах. Это может быть, например, мета-информация о файле (имя, папка, размер, дата изменения и т. д.). База данных одного Archie-сервера может содержать сведения о файлах нескольких FTP-серверов. Она обновляется через определенные промежутки времени, пополняя списки новыми файлами и данными об измененных файлах.

Как только база данных Archie сформирована, ее предоставляют пользователям. Взаимодействие с Archie-сервером происходит в пакетном режиме через электронную почту или в интерактивном — через Telnet, Web-протоколы или другие. При использовании Archie нет нужды помнить имена всех FTP-серверов. Файл можно найти, зная лишь часть его имени или название папки.

Gopher

Протокол Gopher (от английского *gopher* — суслик) предназначен для путешествий по лабиринтам Интернета. Он представляет собой распределенную клиент-серверную систему доставки документов. Gopher-клиент делает доступной для пользователя иерархию ресурсов (серверы, папки, файлы), содержащую необходимый документ. Пользователь может выбрать папку или файл из списка (с клавиатуры или мышью, в зависимости от интерфейса клиентской части Gopher) или провести текстовый поиск. Это напоминает просмотр библиотечного каталога.

Чаще всего пользователи начинают поиск с корневого каталога сервера Gopher, затем обращаются к иерархической структуре папок или к результатам поисковых запросов, получая, наконец, имена необходимых файлов.

Протокол Gopher — это простой клиент-серверный протокол без памяти (stateless), осуществляющий обмен данными через 70-й порт TCP. Клиент посылает серверу текстовую строку и получает ответ в виде текстового блока, причем в качестве разделителей полей оба используют обычные символы. Ответ сервера содержит как действительные, так и наглядные имена ресурсов. Большинство Web-браузеров помимо того, что они — HTTP-клиенты, также являются Gopher-клиентами.

С появлением Web популярность протокола Gopher пошла на убыль, поскольку Web-протоколы (совместно с поисковыми машинами) обес-

печивают сходную функциональность. Протокол Gopher описан в RFC 1436.

Veronica

Veronica — от «Very Easy Rodent-Oriented Netwide Index to Computerized Archives» («очень простой всесетевой каталог компьютеризированных архивов для грызунов») — это система обнаружения информационных ресурсов на большинстве общедоступных серверов Gopher. Veronica для Gopher — то же, что Archie для FTP. Так, протокол Archie расширяет возможности FTP: один запрос на поиск позволяет обнаружить множество документов из разных папок, хранящихся на нескольких FTP-серверах. Подобным образом Veronica повышает эффективность Gopher: один поиск по ключевому слову извлекает множество документов из нескольких папок разных серверов Gopher (из «Gopher-пространства»).

Серверная часть Veronica обладает новым программным обеспечением сбора информации от нескольких серверов Gopher. Получив данные, Veronica сохраняет их на сервере Gopher, доступном для Gopher-клиента через протокол Gopher.

Jughead

Jughead, как и Veronica, предназначен для систем Gopher. Он позволяет производить поиск по ключевым словам на серверах Gopher.

Стандарт Z39.50

Стандарт Z39.50 — «Описание службы и протокола доступа к информации для библиотечных приложений» («Information Retrieval Service Definition and Protocol Specification for Library Applications») — создан для решения проблем, характерных для поиска в нескольких базах данных разного типа. Эта задача требует специальных знаний — как системных, так и в области баз данных.

Z39.50 определяет правила и процедуры взаимодействия двух разнородных систем для поиска и выборки информации, хранящейся в базе данных. Протокол Z39.50 обладает памятью (stateful protocol), а значит, клиент может предавать информацию серверу по частям. Однако допустима и передача запроса целиком.

Z39.50 находится под контролем Национальной организации информационных стандартов США (National Information Standards Organization, NISO), также известной как «Комитет Z39» Американского национального института стандартов (American National Standards Institute, ANSI). NISO развивает стандарты поиска и выборки данных для информаци-

онной индустрии, включая библиотечное и издательское дело. Последняя версия Z39.50, одобренная NISO в 1995 году, призвана заменить собой стандарт «Поиск и выборка» (Search and Retrieve, SR), принятый ISO в 1991 году.

Для работы с Z39.50 существует два вида URL, описанные в RFC 2056: сеансовый (session) — префикс «Z39.50s», интерактивный режим с многими состояниями, и выборочный (retrieval) — префикс «Z39.50r», режим с одним состоянием, для выборки заранее определенной информации. Подробнее о Z39.50 — в RFC 2056, 1729 и 1625.

Служба WAIS

Служба WAIS (Wide Area Information Server) — это еще одна система поиска и отбора информации. Она состоит из серверного и клиентского ПО, а также сетевого протокола. Средствами этой службы пользователи могут запрашивать информацию из нескольких баз данных, составляя запросы на языке, близком к разговорочному английскому. Протокол WAIS использует одну из версий протокола Z39.50.

Как и Gopher, WAIS постепенно сдает свои позиции, уступая место Web-протоколам (и поисковым машинам на их основе), предоставляющим те же возможности.

Harvest

Система поиска и индексирования информации Harvest — это распределенная индексная служба для поиска, извлечения, накопления, организации, кэширования и тиражирования информации. Сервер Harvest содержит много подсистем, в том числе *накопители* (gatherers), *распределители* (brokers), *кэши объектов* (object caches) и *администраторы тиражирования* (replication managers). Конфигурируя эти компоненты, можно задавать формат данных для поиска и тип представления выходных данных. Все данные сохраняются в специальном структурированном формате системы Harvest — Summary Object Interchange Format (SOIF), который позволяет создавать достаточно изощренные запросы.

Harvest не имеет специального клиента или клиент-серверного протокола. Пользователи обращаются к Harvest посредством Web-браузера, который соединяется с сервером Harvest через HTTP-сервер.

С появлением Web к службе Harvest обращаются все меньше и меньше, несмотря на ее неоспоримые достоинства. В то же время популярность поисковых машин различных производителей, использующих для обмена данными Web-протоколы, продолжает расти. Тем не менее неко-

торые поисковые машины на базе Web применяют для сбора информации все тот же Harvest.

Каталоги и поисковые машины для Web

В связи со стремительным ростом Web возникла потребность в инструментальных средствах для поиска документов. Такие средства предоставляет клиенту HTML-форма. Клиент ее заполняет, определяет ключевые слова или предметную тему, а затем отсылает форму серверу по протоколу HTTP. На сервере форма обрабатывается при помощи CGI-сценариев (Common Gateway Interface) и средств ISAPI (Internet Server Application Programming Interface). (Подробнее об HTML, HTTP, CGI и ISAPI — в главах 12 и 13.) Обработка ведется с использованием базы данных (индекса), находящейся на сервере.

Эту базу данных генерируют (и в дальнейшем обращаются к ней через Интернет) с помощью тематического каталога или поисковой машины. Оба варианта имеют свои преимущества и недостатки. Тематический каталог обычно возвращает меньше документов, причем не самые свежие, но более точно освещающие тему. Однако это справедливо только для обычных областей поиска. Поисковая машина, вероятно, выберет существенно больше данных и документов, к тому же новейших, однако часть из них может не соответствовать заданной теме. Учитывая размеры Web, нельзя рассчитывать, что база данных какого-либо одного предметного указателя или одной поисковой машины будет содержать все существующие документы. Некоторая область Web иногда полностью отражена в одной базе данных и вовсе не представлена в другой.

Тематические каталоги

Как было сказано ранее, один из первых методов поиска в Web базировался на создании тематического каталога, содержащего упорядоченный по темам список URL документов. Тематический каталог был задуман как подобие библиотечного. При составлении каталога для поиска URL документов можно использовать программные средства. Однако чтобы определить степень соответствия документа некоторой тематике, требуется вмешательство человека. Один из наиболее известных тематических каталогов — Yahoo! (*http://www.yahoo.com*).

Поисковые машины для Web

Все поисковые машины состоят из двух основных компонентов. Первый предназначен для выборки информации из баз данных, второй — для создания базы данных, соответствующей критериям поиска. Послед-

няя строится с помощью программ, которые часто называют *роботами*, «*гусеницами*», «*пауками*» или «*червями*». Примеры поисковых машин для Web — Lycos, AltaVista, Infoseek, Inktomi и Excite.

Поисковые машины отличаются друг от друга:

- областью Web, охваченной роботом поисковой машины;
- частотой, с которой робот посещает Web-узлы;
- логикой, применяемой для определения соответствия документа поисковому запросу;
- степенью допустимой детализации и уточнения запроса;
- типами документов, о которых робот собирает информацию (HTML, не-HTML, мультимедиа и т. д.).

Поисковые машины охватывают разные объемы данных и имеют как общие черты, так и свои особенности. Например, о AltaVista и Lycos известно, что они охватывают десятки миллионов URL. Робот AltaVista действует так, как указано в файле *robots.txt* (см. следующий раздел) и может быстро и эффективно обрабатывать большие объемы данных. Робот Lycos тоже подчиняется *robots.txt*, но в первую очередь просматривает наиболее популярные Web-станицы. Кроме того, Lycos составляет заключение о мультимедийном наполнении страницы. Робот HotBot тоже следует указаниям *robots.txt* и старается проводить поиск, не чрезмерно нагружая Web-узел. Для этого он копирует часть содержимого Web-узла, обрабатывает его и возвращается за следующей порцией данных.

Поисковые машины бывают общего и специального назначения и могут выбирать документы, относящиеся к определенной теме, например страницы, содержащие медицинскую информацию, или статьи из некоторой публикации.

Существуют также машины, осуществляющие *мета-поиск*. Они не создают собственной базы данных, а последовательно обращаются к нескольким поисковым машинам, а затем уточняют и комбинируют результаты.

Web-роботы

Основа Web, в основном, — серверы, хранящие информацию, и конечные пользователи, запрашивающие ее. Однако пользователи со своими Web-браузерами — не единственные клиенты. Особую разновидность

клиентов составляют программы, которые называют роботами, «червями» или «пауками» (есть и другие названия).

Программа-робот, часто полностью автоматизированная, просматривает Web-узел и извлекает информацию обо всех доступных документах. Основная цель этих действий — составить базу данных для поисковой машины. Любая эффективная поисковая машина являет собой результат скрытой кропотливой работы Web-робота. У роботов есть и другие задачи, в том числе проверка или копирование HTML-документов и отслеживание гиперссылок в этих документах.

Протокол отключения роботов

Web-роботы, как правило, являются HTTP-клиентами и «ползают» по HTML-документам. Для обработки данных другого типа они могут воспользоваться иными протоколами, например NNTP (Network News Transfer Protocol), FTP или почтовыми. В местах хранения больших объемов данных обычно присутствует какая-нибудь программа-робот. Прикинувшись пользователем, она непрерывно исследует информацию, преследуя некую цель.

Иногда Web-администраторы по каким-то причинам запрещают роботам просматривать Web-узлы или их отдельные участки. В этом случае стоит использовать Протокол отключения роботов (Robot Exclusion Protocol) — инструкцию для роботов, описывающую их поведение на данном узле. Протокол применяют в основном администраторы HTTP-серверов, а иногда и авторы HTML-документов.*

Инструкция, описывающая поведение роботов, помещается в специальный файл — *robots.txt*, который находится в корневом каталоге HTTP-сервера. Когда робот обращается к узлу, он прежде всего проверяет содержимое этого файла. Если файл *robots.txt* не существует, значит, у администратора нет для роботов никаких инструкций. Файл *robots.txt* — это текстовый ASCII-файл. Он состоит из строк «User-agent...» (Пользовательский агент), описывающих значение параметра HTTP_USER_AGENT клиентской программы (тип пользовательского агента) и строк «Allow...» (Разрешить) и «Disallow...» (Запретить), где указаны доступные и запрещенные каталоги. Для конкретного типа пользовательского

---

* Этот протокол применим только в отношении «этичных» роботов, т. е. таких, в которых разработчиками заложен механизм распознавания и выполнения инструкций. Такие роботы соответствуют Стандарту отключения роботов (Standard for Robot Exclusion, SRE). — *Прим. перев.*

агента задают одну или несколько строк «Allow...» или «Disallow...». Например, *robots.txt* может состоять из трех директив:

```
# robots.txt для http://www.example.org/, webmaster@example.org
# Плохой робот (BadRobot) имеет доступ только к файлу
# для плохих роботов
User-agent: BadRobot
Disallow: /
Allow: /bad-robot-read-this.html

# Хороший Робот (GoodRobot) имеет доступ всюду
User-agent: GoodRobot
Disallow:
# Ни один робот не допускается в подкаталоги mirrors и archives
User-agent: *
Disallow: /mirrors
Disallow: /archives
```

Тег <META> для роботов

Инструкцию для роботов может содержать не только файл *robots.txt* в корневом каталоге Web-узла, но и конкретный HTML-документ. Это особенно полезно для HTML-страниц большого Web-узла, у авторов которых нет доступа к корневому каталогу сервера (файл *robots.txt* действителен только в корневом каталоге). Этот метод применим, например, для персональной страницы, расположенной на общедоступном сервере поставщика услуг Интернета.

При помощи тега <META> в HTML-документе создаются специальные заголовки. Этот тэг имеет следующий синтаксис:

```
<META NAME="ROBOTS" CONTENT="NOINDEX, NOFOLLOW">
```

Для него действительны следующие директивы:

- INDEX — проиндексировать документ;
- NOINDEX — не индексировать документ;
- FOLLOW — пройти по гиперссылкам документа;
- NOFOLLOW — игнорировать гиперссылки.

Разделенные запятыми директивы можно задавать в одном теге <META>.

Протоколы и стандарты будущего

Описанные в этом разделе протоколы еще только развиваются, однако знакомство с ними станет логичным завершением рассказа о протоколах поиска в Интернете.

Проект «Управление роботами»

Проект «Управление роботами» (Robot Guidance Project) — это попытка усовершенствовать Протокол отключения роботов. Текущий стандарт в основном ограничивает роботам доступ и не допускает никаких положительных инструкций. Новый стандарт затрагивает такие вопросы, как более тонкая детализация входных данных для разработчиков содержания, работа с протоколами и документами разных стандартов — не только HTTP и HTML, а также активное управление роботами вместо исключающих и запрещающих мер.

Протокол CIP

Протокол CIP (Common Indexing Protocol) — новый проект рабочей группы Find в составе IETF — это общий протокол для обмена индексной информацией. Он основан на концепциях предшествующих протоколов, включая Whois++, X.500 (LDAP) и CCSSO. В нем также использована одна из версий формата SOIF (из системы Harvest) для структурирования индексных файлов. Этот стандарт еще только разрабатывается, однако он весьма перспективен.

Ссылки

ftp://boombox.micro.umn.edu/pub/gopher

gopher://veronica.scs.unr.edu:70/11/veronica

http://lcweb.loc.gov/z3950/agency

http://www.wais.com

http://harvest.transarc.com

http://info.webcrawler.com/mak/projects/robots/robots.html

http://www.botspot.com/robotguidance

http://www.ietf.org/html.charters/find-charter.html

RFC 2056, «Uniform Resource Locators for Z39.50»

RFC 1729, «Using the Z39.50 Information Retrieval Protocol»

RFC 1625, «WAIS Over Z39.50»

RFC 1436, «The Internet Gopher Protocol (a Distributed Document Search and Retrieval Protocol)»

Часть 3

Обмен информацией

ГЛАВА 9

Электронная почта

Электронная почта (electronic mail, e-mail), пожалуй, — самое популярное применение сети Интернет. В этой главе подробно описаны основные протоколы электронной почты, используемые в Интернете.

В самом начале электронная почта существовала в виде самодельных почтовых систем. Коммерческие версии, например PROFS (Professional Office System) компании IBM, поначалу разрабатывались только для мэйнфреймов и мини-ЭВМ.

Однако появление локальных сетей привело к развитию коммерческих продуктов электронной почты. Известный недостаток этих реализаций — неполная поддержка стандартных протоколов. В качестве почтового сервера выступал файловый сервер, который иногда называли «хранилищем сообщений». Персональные компьютеры одной локальной сети обменивались электронной почтой, записывая информацию в файлы, хранящиеся на файловом сервере. Последний, в свою очередь, обеспечивал совместное использование файлов разными операционными системами, например Microsoft LAN Manager и Novell NetWare. Маршрутизация сообщений от одного хранилища кдругому выполнялась посредством частных протоколов или версий, не полностью соответствующих стандартам. Два хранилища сообщений могли находиться как в одной локальной сети, так и связываться через глобальную сеть или при помощи телефонных соединений. К системам электронной почты для локальных сетей относятся Microsoft Mail и Lotus cc:Mail. Электронная почта для локальных сетей очень хорошо работает на уровне отделов. К сожалению, поскольку такие системы слабо масштабируемы, иногда возникают значительные административные трудности при соединении нескольких почтовых серверов (по одному или более на отдел). Такая ситуация показана на рис. 9-1.

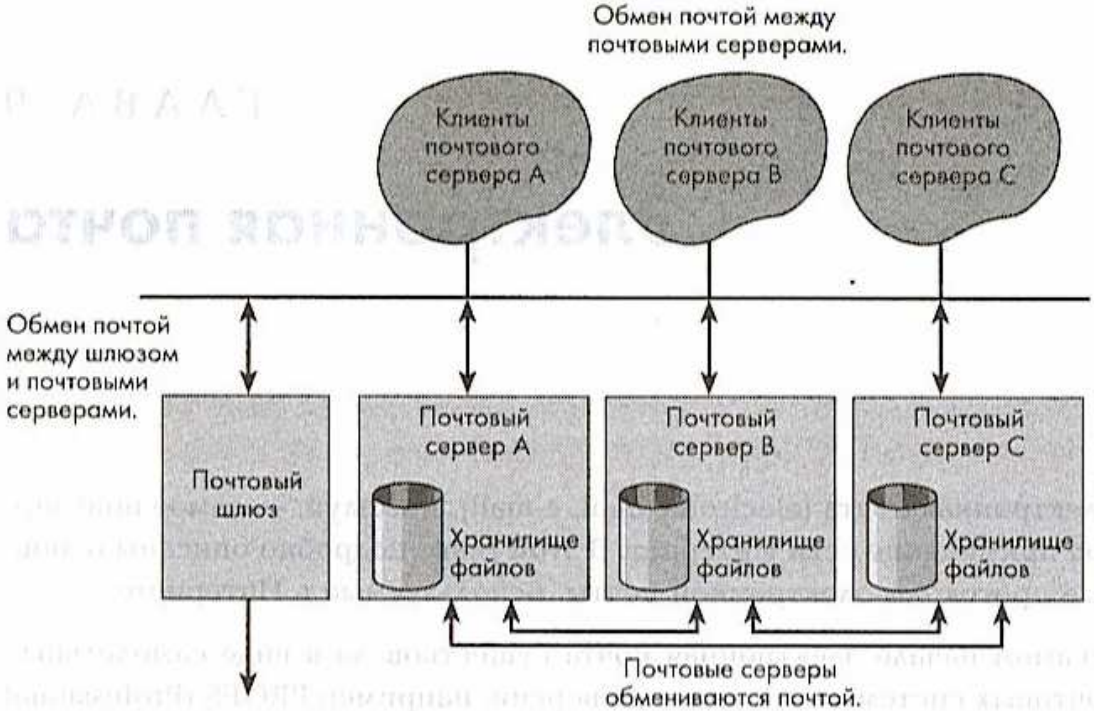


Рис. 9-1. *Архитектура почтовой системы, работающей на уровне отдела*

В системах электронной почты следующего поколения клиенты взаимодействовали с хранилищем сообщений посредством механизма *удаленного вызова процедур* (Remote Procedure Call, RPC). Теперь почтовый сервер состоит из многих работающих параллельно процессов, и его нельзя считать просто файловым сервером. Преимущество систем электронной почты такого типа в том, что они безопаснее и позволяют серверу самому обрабатывать информацию. А значит, поддерживают создание других приложений, например для автоматизации делопроизводства средствами электронной почты. Сильно упрощая, можно сказать, что приложение для автоматизации позволяет серверу обрабатывать сильно структурированную электронную почту на основе содержимого полей сообщений. Возьмем, к примеру, страховую компанию. Телефонист принимает заявку и заносит в сообщение, а сервер — автоматически направляет ее конкретным людям для обработки. Электронное сообщение проходит путь от отдела приема заявок к исполнителям, а потом — в бухгалтерию, причем каждый получатель вправе обновлять содержимое некоторых полей сообщения.

Ниже определены термины, применяемые повсеместно, независимо от типа использующейся системы электронной почты. Электронное сообщение создает человек или компьютерная программа. Человек это де-

лает посредством пользовательского интерфейса, работая на персональном компьютере, мэйнфрейме или мини-ЭВМ. Пользовательский интерфейс обычно взаимодействует с другим программным обеспечением (в виде динамической или другой библиотеки), обеспечивающим взаимодействие с почтовым сервером. Саму библиотеку или ее сочетание с пользовательским интерфейсом называют *пользовательским агентом* (user agent). Обычно пользовательский агент подключается к почтовому серверу через локальную сеть или по коммутируемой (телефонной) линии. Почтовый сервер, с которым связывается пользователь, называют *домашним сервером* (home server) — через него пользователь отправляет свои электронные сообщения. Также домашний сервер принимает и временно хранит все сообщения, адресованные пользователю. Почтовый сервер пересылает электронную почту другим серверам посредством программного модуля — *агента пересылки почты* (mail transfer agent, MTA). Прежде чем попасть на домашний сервер адресата, сообщение иногда проходит множество MTA. На рис. 9-2 демонстрируется такая ситуация, а также указаны протоколы, применяемые на разных этапах.

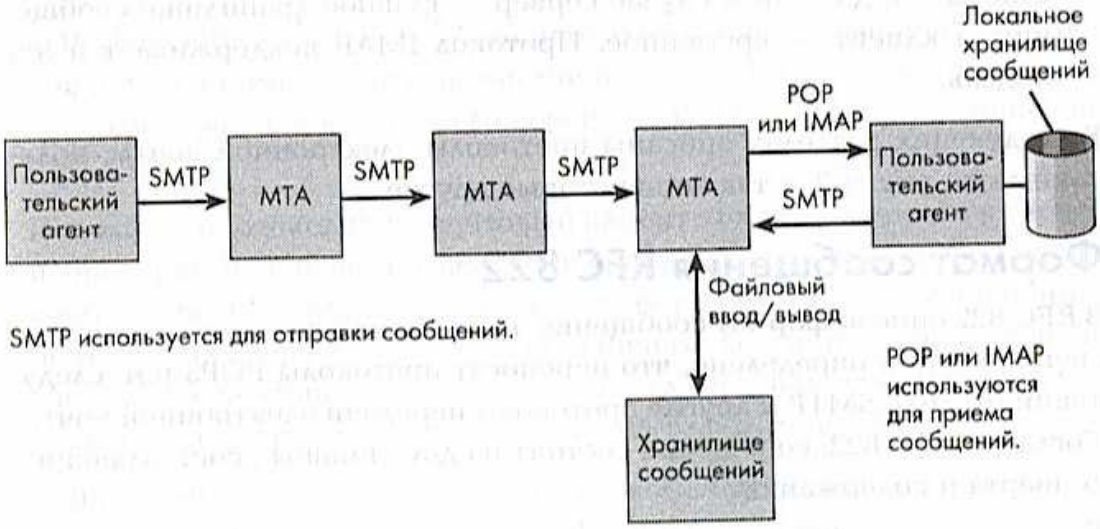


Рис. 9-2. *Маршрут электронной почты через несколько MTA до домашнего сервера адресата (на разных этапах применяются разные протоколы)*

Для описания современной клиент-серверной электронной почты используют термины *автономная* (offline), *интерактивная* (online) и *отключенная* (disconnected) модель.

- Автономная модель — клиент периодически подключается к серверу и принимает накопившуюся почту. После того как сообщения

переданы пользователю, сервер их удаляет. Вся дальнейшая обработка выполняется клиентом, и сервер об этом не заботится. Такая модель реализована в протоколе POP (Post Office Protocol), описанном далее в этой главе. Протокол IMAP (Internet Message Access Protocol) — о нем тоже рассказано далее — также поддерживает эту модель, но его лучше использовать в интерактивной и отключенной моделях.

- Интерактивная модель — клиент устанавливает сеанс связи с сервером, и вся электронная почта обрабатывается во время сеанса на сервере, а клиент управляет этим процессом. Вот некоторые протоколы, использующие эту модель, — IMAP, NFS (Network File System) и CIFS (Common Internet File System).
- Отключенная модель — нечто среднее между автономной и интерактивной моделью. Клиент подключается к серверу, принимает избранные сообщения и обрабатывает их в автономном режиме. Позднее клиент вновь устанавливает связь с сервером и передает изменения (правку сообщения или адресной книги, удаление сообщений, ответы и т. д.) В этом случае сервер — главное хранилище сообщений, а клиент — временное. Протокол IMAP поддерживает и эту модель.

В следующих разделах описаны протоколы электронной почты, показанные на рис. 9-2, а также некоторые другие.

Формат сообщения RFC 822

В RFC 822 описан формат сообщения электронной почты; то есть в этой спецификации определено, что переносят протоколы POP3 (см. следующий раздел), SMTP и другие протоколы передачи электронной почты. Согласно RFC 822, сообщение состоит из двух главных составляющих: конверта и содержания.

В конверте хранятся данные о пересылке и доставке сообщения. Также здесь имеется информация, необходимая для ответа на сообщение. Обычно она представлена в виде текстовых строк, состоящих из ключевого слова, значения и символа возврата каретки и/или перевода строки. Примеры ключевых слов: «From» (От), «Reply to» (Ответ для) и т. д. То есть в конверте указаны как минимум адреса электронной почты отправителя и получателя.

Содержание сообщения состоит, в свою очередь, из заголовка и тела. Заголовок содержит информацию, созданную автоматически пользова-

тельским агентом отправителя с обновлениями от каждого MTA, обработавшего это сообщение. В заголовке есть идентификатор сообщения (message ID), а также даты и время, показывающие, когда каждый MTA обрабатывал сообщение. Пользовательский агент получателя может сжать, переформатировать или вообще скрыть заголовок сообщения. Тем не менее заголовок всегда присутствует. Тело содержит само сообщение, точнее текст, созданный отправителем, поскольку в этой роли может выступать человек или компьютерная программа. Тело сообщения отделено от заголовка пустой строкой — в ней есть только символы возврата каретки (carriage return, CR) и перевода строки (linefeed, LF).

В некоторых реализациях RFC 822, применяемых в Интернете, предполагается, что длина строк не превышает 1 000 байт, а размер сообщения — 64 кбайт. Поскольку отправитель, вообще говоря, не знает о маршруте сообщения и о том, проходит ли оно через промежуточные узлы, использующие описанную реализацию, то перед отправкой сообщение необходимо закодировать, а при приеме — декодировать.

Протокол POP

POP (Post Office Protocol) — это протокол клиент-серверной электронной почты, применимый в автономной модели. Его пересматривали уже несколько раз и последняя его версия — POP3 (версия 3) — описана в RFC 1939.

Используя протокол POP, почтовый клиент при подключении к серверу и проверке новой почты получает сразу все накопившиеся для него сообщения. POP-клиент не может выборочно принять сообщения с почтового сервера — либо все, либо ничего. После того как сообщения загружены, почтовый клиент вправе их удалять или изменять, причем с сервером электронной почты он уже не взаимодействует.

POP3-клиент отсылает POP3-серверу команды и ожидает ответы. Команды протокола POP3 — это ASCII-строки, начинающиеся с ключевого слова (допустимых ключевых слов не так много). Ответы состоят из одной либо из нескольких строк. Первая строка ответа содержит ASCII-текст «+OK» или «–ERR», показывая соответственно успех или ошибку выполнения команды. Если ответ состоит из нескольких строк, то последняя содержит только точку и символы CR/LF. Чтобы при этом исключить разногласия, сервер осуществляет *дополнение точками* (dot stuffing) — любая строка, начинающаяся с точки, но не являющаяся последней, дополняется еще одной точкой, которую удалит клиент.

Протокол POP3 описывается в терминах *автомата* (state machine) — воображаемой машины, которая имеет конечное число состояний. Переход из одного состояния в другое происходит, только когда поступают входные данные и выполняется некоторое условие. Для протокола POP3 существуют три возможных состояния: *авторизация* (authorization), *работа* (transaction) и *обновление* (update). Когда клиент устанавливает связь с сервером электронной почты, последний находится в состоянии авторизации. Когда клиент успешно подтвердит свою личность, сервер переходит в рабочее состояние. Если клиент пошлет команду QUIT, то сервер перейдет в состояние обновления. По завершении обработки в состоянии обновления автомат возвращается в состояние авторизации. В каждом состоянии доступны только определенные команды. В табл. 9-1 приведены команды протокола POP3, их параметры, необходимые состояния и даны пояснения. Параметры в квадратных скобках — необязательные.

Протокол POP3 требует от клиента «удостоверение личности» (имя пользователя и пароль), поэтому доступ к электронной почте можно регулировать. Если пользователь посылает команду USER, то за ней следует пароль в виде открытого ASCII-текста. В расширении протокола POP3 есть команда APOP, предусматривающая шифрование пароля перед передачей. Когда используется команда APOP, вновь подключающийся клиент получает от POP3-сервера «приветствие» (ASCII-текст). Оно содержит строку, уникальную для каждого подключения клиента. Клиент добавляет к этой строке свой пароль в открытом виде и применяет к полученной строке хеш-функцию MD5. После этого имя пользователя и только что вычисленный дайджест используются как параметры команды APOP.

Таблица 9-1. *Команды протокола POP3*

Команда	Параметры	Состояние	Описание
USER	Имя пользователя	Авторизация	Указывает почтовый ящик, с которым желает работать пользователь
PASS	Пароль	Авторизация	Передает серверу в открытом виде пароль для почтового ящика, указанного командой USER. Успешное выполнение переводит автомат в рабочее состояние
APOP	Имя, дайджест	Авторизация	Позволяет безопасно пересылать пароль. Параметр *Имя* указывает почтовый ящик, а параметр *Дайджест* — это MD5-дайджест (об MD5 см. RFC 1321). Эта команда переводит автомат в рабочее состояние

Таблица 9-1. *Команды протокола POP3* (продолжение)

Команда	Параметры	Состояние	Описание
STAT	Нет	Работа	Запрашивает у сервера статистику о почтовом ящике, например количество сообщений и их общую длину в байтах
UIDL	[№ сообщения]	Работа	Если параметр указан, то сервер возвращает уникальный идентификатор для данного сообщения. Если параметра нет, сервер возвращает уникальный идентификатор для всех сообщений. Все идентификаторы в течение одного POP-соединения уникальны
LIST	[№ сообщения]	Работа	Если параметр указан, то сервер возвращает номер и размер данного сообщения. Если параметра нет, сервер возвращает номера и размеры всех сообщений
RETR	№ сообщения	Работа	Сервер возвращает полный текст сообщения с указанным номером
DELE	№ сообщения	Работа	Сервер помечает указанное сообщение как удаленное. Непосредственно удаление происходит, когда клиент посылает команду QUIT
RSET	Нет	Работа	Сервер снимает пометки со всех «удаленных» сообщений. Это — отмена (undo) всех команд DELE, выполненных до настоящего момента
TOP	№ сообщения, *n*	Работа	Сервер возвращает первые *n* строк указанного сообщения. Параметр *n* должен быть неотрицательным целым числом
NOOP	Нет	Работа	Сервер возвращает положительный ответ
QUIT	Нет	Работа и авторизация	Клиент собирается завершить сеанс связи. Если сервер находится в рабочем состоянии, то переходит в состояние обновления и удаляет все помеченные для удаления сообщения, а потом отсылает положительный ответ. Эта команда вызывает переход из рабочего состояния в состояние обновления, а затем в состояние авторизации. Если сервер при получении этой команды уже находится в состоянии авторизации, то сеанс связи завершается, при этом сервер не переходит в состояние обновления

Другое расширение протокола POP3 определяет команду XTND XMIT. Это расширение можно использовать вместо протокола SMTP (описанного в следующем разделе) для отправки исходящей почты от клиента серверу. Однако оно не получило широкого применения.

С другой стороны, сам протокол POP3 применяется очень широко. Это означает, что почти все поставщики услуг Интернета предлагают конечным пользователям доступ к электронной почте по протоколу POP3. POP3-серверы «слушают» 110-й порт TCP. Протокол POP3 описан в RFC 1957, 1939 и 1725.

Протокол SMTP

Как следует из названия, SMTP (Simple Mail Transfer Protocol) — простой протокол передачи электронной почты. Обычно его применяют для передачи почты от клиента серверу, а также от одного сервера другому.

SMTP это протокол «запросов и ответов». Команды и ответы являются ASCII-строками, завершающимися символами CR и LF. Ответы также содержат трехзначное число — код возврата. Его можно использовать для управления автоматом протокола.

В модели ISO/OSI протокол SMTP размещается над TCP. Обычно после установления TCP-соединения отправитель посылает SMTP-команду HELO, чем идентифицирует себя. Затем он отправляет SMTP-команду MAIL. Если SMTP-сервер на другом конце соединения ответит OK, это означает, что он готов принимать почту. Тогда отправитель посылает SMTP-команду RCPT, которой сообщает адресата электронного сообщения. Получатель отвечает, готов ли он принимать почту для данного адресата. Если сообщение электронной почты предназначено нескольким получателям, то такой процесс согласования выполняется для каждого из них. После этого происходит собственно передача сообщения — SMTP-командой DATA. Командой VRFY можно выяснить существование почтового ящика и получить подробную информацию о его пользователе. Для расширения списков рассылки служит SMTP-команда EXPN. Кроме того, существует SMTP-команда TURN, позволяющая изменить направление соединения на обратное, то есть сделать так, чтобы сервер-отправитель стал получателем. Некоторые администраторы из соображений безопасности запрещают команды VRFY и TURN.

При получении почты для некоторого адресата SMTP-сервер может сообщить о неточности адреса. (Значение кода возврата равно 251, за ним следует строка, в которой указан корректный адрес получателя письма.) При этом принимающий SMTP-сервер берет на себя ответствен-

ность за дальнейшую доставку сообщения. Принимающий SMTP-сервер может выдать также другой ответ, с кодом 551 (строка с адресом тоже присутствует). Это означает, что он отказывается принимать электронную почту и ответственность за сообщение остается на отправителе.

Протокол SMTP разработан так, что позволяет эффективно, за один сеанс связи между клиентом и сервером, передавать много сообщений одному или нескольким адресатам. В RFC 821 описан транспортный протокол SMTP, а в RFC 822 — структура сообщений, используемых SMTP. Другими словами, в RFC 822 изложен синтаксис и семантика электронных сообщений, передаваемых посредством протокола, описанного в RFC 821. Более поздние RFC расширяют структуру сообщений.

SMTP-серверы маршрутизируют электронную почту на основе имени домена из адреса получателя (о DNS см. главу 6). Делается это при помощи записей типа MX в базе DNS. Такие записи регистрируют имя домена и связывают с ним SMTP-узел, которому следует направлять всю почту, поступающую в этот домен.

В табл. 9-2 представлена сводка SMTP-команд. В именах команд регистр символов не имеет значения, но в параметрах он важен (например, в названии почтового ящика).

Таблица 9-2. *SMTP-команды*

Команда	Описание
HELO	Сообщает серверу о клиенте
MAIL	Начинает передачу почты
RCPT	Указывает одного получателя сообщения; обычно следует за командой MAIL
DATA	Следует за командой RCPT; сообщает о завершении перечня адресатов сообщения и начинает передачу данных
SEND	Замена команды MAIL; используется редко
SOML	Замена команды MAIL; используется редко (SOML обозначает SEND OR MAIL)
SAML	Замена команды MAIL; используется редко (SOML обозначает SEND AND MAIL)
VRFY	Применяется клиентом для проверки существования данного пользователя или почтового ящика; некоторые серверы по соображениям безопасности не отвечают на эту команду
EXPN	Применяется клиентом для проверки существования списка рассылки и дополнения его новыми данными
HELP	Вопрос серверу: «Какие команды ты поддерживаешь?»
NOOP	Нет операции; сервер должен ответить OK
QUIT	Клиент посылает эту команду для завершения сеанса

(см. след. стр.)

Таблица 9-2. *SMTP-команды* (продолжение)

Команда	Описание
RSET	Сеанс инициализируется заново; текущая передача (если велась) прерывается
TURN	Клиент предлагает серверу поменяться ролями, чтобы переслать почту в обратном направлении; используется редко

SMTP-серверы «слушают» 25-й порт TCP. Протокол SMTP описан в RFC 821.

Расширения SMTP

В настоящее время описан целый ряд расширений протокола SMTP, зарегистрированных IANA (Internet Assigned Numbers Authority) — подразделением IAB (Internet Architecture Board). (Подробности — по адресу *http://www.iana.org*.) Расширения SMTP обеспечивают обратную совместимость. То есть если хотя бы один из пары SMTP-серверов не поддерживает конкретное расширение, то оно не будет использоваться. Если SMTP-серверу не требуются дополнительные возможности, то не нужно реализовывать расширение. Описаны следующие расширения SMTP:

- RFC 2197 — пакетная обработка SMTP-команд. Позволяет посылать несколько команд за одну операцию протокола TCP. В Интернете часто случается, что соединение по протоколу SMTP проходит через сети с большой задержкой (то есть интервал времени между передачей запроса и его приемом на другом конце велик). Пакетная обработка SMTP-команд в таких сетях повышает эффективность обмена почтой. В RFC 2197 описан способ, которым серверы сообщают о том, поддерживают ли они пакетную обработку команд;
- RFC 1830 — передача больших объемов данных. Благодаря этому RFC, отпала необходимость завершать каждую строку символами CR и LF. Кроме того, появилась возможность передавать большие двоичные вложения «как есть», без применения кодировок Base64 или Quoted-Printable;
- RFC 1845 — пометка SMTP-трафика как транзакции. Если TCP-сеанс (а следовательно, и SMTP-сеанс) прервется, то есть возможность установить новый сеанс связи и продолжить транзакцию (последнюю). Это повышает эффективность передачи больших объемов данных;

- RFC 1869 — протокол, посредством которого SMTP-сервер сообщает клиенту, какие расширения SMTP он поддерживает. Этот протокол описан так, что существующие версии SMTP можно не модернизировать до тех пор, пока не понадобится использовать или реализовать дополнительные услуги;
- RFC 1870 — позволяет SMTP-клиенту указывать размер передаваемого сообщения. Сервер сам решает, принимать это сообщение или нет. Такая возможность необходима, поскольку при использовании MIME SMTP-сообщения становятся гораздо объемнее, и принимающему SMTP-серверу иногда не хватает ресурсов (памяти, дискового пространства и т. п.) для их обработки;
- RFC 1891 — позволяет SMTP-клиенту указывать природу сообщений DSN (Delivery Status Notifications — «сообщения о состоянии доставки») и необходимость их создания. Клиент вправе указать, что DSN должны включать содержимое сообщения и всю информацию, позволяющую отправителю идентифицировать получателей сообщения и транзакцию, в которой отправлено исходное сообщение;
- RFC 1985 — позволяет двум SMTP-узлам в защищенном виде передать электронную почту в одном направлении, а потом — в обратном. Для этого вводится новая SMTP-команда ETRN;
- RFC 2034 — способ информирования SMTP-клиента о том, что сервер возвращает расширенные коды состояния. Эти коды не описаны в оригинальной спецификации протокола SMTP, и, возможно, не все реализации SMTP распознают их.

Протокол UUCP

Строго говоря, протокол UUCP (UNIX-to-UNIX Copy Protocol) предназначен для передачи файлов, но его также используют для передачи электронной почты, для чего сообщения дополняются адресной информацией. Обычно UUCP применяют в качестве альтернативы SMTP, когда электронная почта пересылается по телефонной линии. Подробности — в RFC 976.

Протокол IMAP версии 4

Протокол IMAP (Internet Message Access Protocol) предпочтительнее в том случае, когда ПО почтового клиента выполняется на портативном компьютере. При применении IMAP пользователь может загружать сообщения выборочно и даже частично. Это очень полезно при доступе

к электронной почте по низкоскоростной телефонной линии с портативного компьютера. В спецификации RFC 2060 протокол IMAP описан в терминах конечного автомата. Для каждого состояния существует ограниченное число команд, которые клиенту разрешено направлять почтовому серверу. Некоторые команды вызывают переход в иное состояние, для которого применяется другой набор команд. На рис. 9-3 показана схема конечного автомата протокола IMAP согласно описанию в RFC 2060.

Рис. 9-3. *Конечный автомат IMAP согласно описанию в RFC 2060*

Посредством IMAP клиент может загрузить с почтового сервера на свой компьютер только выбранные сообщения, а затем отключиться. Отсоединившись, клиент вправе модифицировать любое из принятых сообщений. Чтобы обеспечить синхронизацию, IMAP присваивает каждому сообщению уникальный идентификатор, действительный в любом IMAP-сеансе.

Команды IMAP состоят из идентификатора, названия и, если необходимо, параметров. Ответ сервера содержит метку, которая позволяет клиенту сопоставлять ответы с командами. Это необходимо, поскольку IMAP

разрешает пользователю отдавать несколько команд, не получая ответы на предыдущие. Естественно, клиент должен следить за тем, чтобы каждая метка была либо уникальна, либо, если используется множество меток, оно (множество) было достаточно большим, чтобы исключить повторное использование какой-то метки раньше, чем будет получен ответ на команду, содержащую ее же. Сервер способен генерировать ответы, не привязанные ни к одной отданной команде. Такие ответы называют *непомеченными* (untagged), и они содержат специальную метку «звездочка» (*).

Вот какие ответы IMAP бывают:

- OK — успешное выполнение ранее отправленной команды; в непомеченном ответе означает наличие дополнительной информации;
- NO — неудачное выполнение предыдущей команды; в непомеченном ответе используется для передачи предупреждений;
- BAD — неверная команда или аргумент; в непомеченном ответе сообщает о серьезной ошибке;
- PREAUTH — содержится только в непомеченных ответах; указывает на ненадобность команды LOGIN;
- BYE — содержится только в непомеченных ответах; информирует, что сервер завершает сеанс.

Команды протокола IMAP описаны в табл. 9-3. Параметры в квадратных скобках необязательны.

Таблица 9-3. *Команды протокола IMAP*

Команда	Состояния	Параметры	Описание	Результат
NOOP	Все	Нет	Используется как сообщение «еще жив» для сброса таймеров сервера	OK, BAD
CAPABILITY	Все	Нет	Сервер выдает непомеченный ответ, в котором указывает свои возможности; в настоящее время определена только одна возможность — IMAP 4	OK, BAD
LOGOUT	Все	Нет	Клиент сообщает о завершении сеанса; сервер возвращает непомеченный ответ «BYE»	OK, BAD

(см. след. стр.)

Таблица 9-3. *Команды протокола IMAP* (продолжение)

Команда	Состояния	Параметры	Описание	Результат
AUTHENTICATE	Не аутентифицировано	Название механизма аутентификации	Клиент указывает механизм аутентификации; допустимы — Kerberos V4, S/KEY и GSSAPI (см. главу 6)	OK, NO, BAD
LOGIN	Не аутентифицировано	Идентификатор пользователя, пароль	Предоставляет идентификатор пользователя и пароль в открытом виде; менее безопасна, чем команда AUTHENTICATE	OK, NO, BAD
CREATE	Аутентифицировано	Название почтового ящика	Создает почтовый ящик	OK, NO, BAD
DELETE	Аутентифицировано	Название почтового ящика	Удаляет почтовый ящик	OK, NO, BAD
SELECT	Аутентифицировано	Название почтового ящика	Клиент указывает почтовый ящик, к которому будут применяться последующие команды; сервер возвращает ответ на эту команду, а также непомеченные ответы, содержащие подробную информацию об этом почтовом ящике	OK, NO, BAD
EXAMINE	Аутентифицировано	Название почтового ящика	Аналогична команде SELECT, но почтовый ящик открывается только для чтения	OK, NO, BAD
RENAME	Аутентифицировано	Старое название почтового ящика, новое название почтового ящика	Переименовывает указанный почтовый ящик	OK, NO, BAD
SUBSCRIBE	Аутентифицировано	Название почтового ящика	Заносит данный почтовый ящик в список «подписанных»	OK, NO, BAD
UNSUBSCRIBE	Аутентифицировано	Название почтового ящика	«Отмена» команды SUBSCRIBE; удаляет данный почтовый ящик из списка «подписанных»	OK, NO, BAD
APPEND	Аутентифицировано	Почтовый ящик [флаги] [дата/время], сообщение	Заносит выбранное сообщение в указанный почтовый ящик; другие параметры (если есть) добавляются к сообщению	OK, NO, BAD

Таблица 9-3. *Команды протокола IMAP* (продолжение)

Команда	Состояния	Параметры	Описание	Результат
LIST	Аутентифицировано	Контекст, почтовый ящик	Позволяет пользователю просмотреть подмножество допустимых имен; первый аргумент предоставляет серверу дополнительные данные	OK, NO, BAD
LSUB	Аутентифицировано	Контекст, почтовый ящик	Аналогична команде LIST, но ответ сервера затрагивает только потовые ящики, «подписанные» командой SUBSCRIBE	OK, NO, BAD
STATUS	Аутентифицировано	Почтовый ящик	Запрашивает статус указанного почтового ящика	OK, NO,
FETCH	Выделено	Набор сообщений, название элемента сообщения	Запрашивает данные относительно сообщения; это могут быть как сообщения целиком, так и их части	OK, NO, BAD
STORE	Выделено	Набор сообщений, название элемента сообщения, значение элемента сообщения	Изменяет данные, связанные с указанным сообщением; по умолчанию измененное значение возвращается в непомеченном ответе	OK, NO, BAD
CHECK	Выделено	Нет	Вызывает сохранение на диске состояния выделенного почтового ящика (механизм сохранения состояния зависит от реализации)	OK, BAD
EXPUNGE	Выделено	Нет	Удалят все сообщения, отмеченные на удаление	OK, NO, BAD
SEARCH	Выделено	[набор символов], [критерий поиска]	Ищет в почтовом ящике сообщения, удовлетворяющие данному критерию	OK, NO, BAD
COPY	Выделено	Набор сообщений, название почтового ящика	Копирует указанные сообщения в указанный почтовый ящик	OK, NO, BAD

(см. след. стр.)

Таблица 9-3. *Команды протокола IMAP* (продолжение)

Команда	Состояния	Параметры	Описание	Результат
UID	Выделено	Название команды, аргументы	Говорит, что заданная команда (COPY, FETCH, STORE или SEARCH) должна быть исполнена, а в качестве входных данных получить уникальные идентификаторы вместо порядковых номеров сообщений	OK, NO, BAD
X	Выделено	Определяются реализацией	Экспериментальная команда, зависящая от реализации	OK, NO, BAD
CLOSE	Выделено	Нет	Удаляет все сообщения, отмеченные для удаления; вызывает переход в состояние «Аутентифицировано»	OK, NO, BAD

Протокол IMAP гораздо сложнее POP3, и его труднее реализовать. Он предъявляет повышенные требования к устройствам хранения данных, поскольку старые сообщения могут накапливаться на сервере. С другой стороны, очень хорошо, что неплохая стратегия резервного копирования уже реализована на почтовом сервере — это защищает целостность данных. IMAP поддерживает обработку почты сервером и коллективные почтовые ящики, что позволяет разрабатывать приложения для рабочих групп. (Коллективный почтовый ящик — это почтовый ящик, используемый несколькими адресатами.) Также IMAP позволяет реализовать поиск по электронной почте, находящейся на почтовом сервере, причем клиенту не обязательно загружать сами сообщения.

Если используется протокол TCP, то IMAP-сервер «слушает» 143-ий порт.

Во время написания этой книги, компании Microsoft и Netscape заявили о начале поставки клиентского и серверного ПО для протокола IMAP. Программные реализации IMAP также есть у компаний SunSoft, ICL и NetManage.

Протокол IMAP описан в RFC 2060, 1731 и 1730.

Технология MIME

В RFC 822 описана структура сообщений, передаваемых по протоколу SMTP, но она обладает некоторыми недостатками. В RFC 2049, 2048, 2047, 2046 и 2045 определен альтернативный способ описания структу-

ры сообщений, передаваемых протоколом SMTP, — MIME (Multipurpose Internet Mail Extensions).

До появления технологии MIME возможности Интернета были ограничены передачей только ASCII-данных. То есть данные передавались байтами, но старший бит каждого байта должен был равен 0. Технология MIME устраняет эту неполноценность, позволяя передавать двоичные данные, то есть любые значения всех восьми бит одного байта. Заметьте, что MIME не заменяет собой RFC 822, но лишь расширяет возможности электронной почты, описанные в этом RFC. Более того, такое расширение обеспечивает обратную совместимость.

Типы сообщений MIME

В RFC 822 сообщение определяется просто как текст без какой-либо структуры. В MIME для текстовых и двоичных данных определены разные типы содержания. Каждый базовый тип может иметь дополнительные подтипы. Размер сообщения ограничен 64 килобайтами, поэтому более крупные сообщения перед отправкой разбиваются на меньшие и восстанавливаются на приемном конце. Технология MIME допускает рекурсию. Другими словами, сообщение может содержать элемент, который сам по себе является MIME-сообщением. В следующих разделах описаны шесть типов содержимого.

Text

Тип Text используется для передачи тела сообщения. Подтип Plain является основным и соответствует формату RFC 822. Среди других возможных подтипов есть соответствующий формату RTF (Rich Text Format). Он поддерживает специальное форматирование символов, например курсивный и жирный шрифт. Тип Text допускает наборы символов только US-ASCII или от ISO-8859-1 до ISO-8859-10.

Image

Тип Image означает, что указанные данные являются изображением. Возможные подтипы: GIF (Graphic Image Format) и MPEG (Motion Picture Experts Group).

Audio

Тип Audio обозначает звуковые данные, например голос или музыку.

Application

Тип Application позволяет передавать данные в формате некоторых приложений, например таблицы Microsoft Excel или документы Microsoft Word. Возможные подтипы:

- Octet-Stream (байтовый поток) — обозначает данные не связанные ни с одним приложением;
- ODA (Office Document Architecture) — обозначает данные для приложений Microsoft Office;
- PostScript (от Adobe) — обозначает данные для высококачественной печати.

Structured

Тип Structured иногда называют составным типом. Он переносит не данные, а комбинации типов, перечисленных выше. Для этого типа определены четыре подтипа:

- Alternative — сообщает о том, что одни и те же данные представлены в нескольких форматах, например в обычном ASCII, RTF или в формате Word (.doc). Принимающее почтовое приложение вправе само выбрать одно из этих представлений в зависимости от своих способностей;
- Digest — обозначает, что содержимое представляет собой подборку сообщений. Каждая часть такого сообщения сама по себе является полноправным сообщением. Шлюзы должны уделять особое внимание этому подтипу, поскольку такие подборки сообщений необходимо правильно обрабатывать;
- Parallel — означает, что разные части сообщения должны воспроизводиться одновременно, например звук и видео;
- Mixed — означает, что сообщение состоит из частей разных типов, например текста, звука и видео.

Message

Тип Message обозначает тело сообщения. Возможные подтипы:

- RFC 822 — обозначает сообщение электронной почты;
- Partial — обозначает «частичное» сообщение. Используется для отправки по частям сообщений, превышающих размер 64 кбайт;

- External Body — используется для ссылки на файл, являющийся внешним по отношению к самому сообщению. Так обычно указывают ссылки на большие файлы, которые можно принять по протоколу FTP.

Методы MIME-кодирования

Методы MIME-кодирования используются для перевода данных из вида, где значащими являются все 8 бит, в 7-разрядный ASCII. Это позволяет гарантировать, что данные успешно пройдут через Интернет, где многие почтовые системы обрабатывают только ASCII-данные. До появления MIME для кодирования использовался метод UUENCODE, а для декодирования — UUDECODE (см. главу 4). Поле **Content-Transfer-Encoding** MIME-заголовка может принимать одно из шести значений, описанных в следующих разделах.

Quoted-Printable

Кодировка Quoted-Printable («отображаемая с выделениями») используется, когда большинство символов являются 7-разрядными ASCII-символами (например для сообщения на скандинавском языке). При такой кодировке все ASCII-символы не меняются, а кодируются только те, у которых старший бит (8-разрядного байта) равен 1. В результате большую часть сообщения можно прочитать даже без декодирования.

Base64

Кодировка Base64 делает невозможным прочтение данных без декодирования, а сообщение удлиняется примерно на одну треть. Алгоритм кодирования преобразует группы по три 8-разрядных символа (24 бита) в группы по четыре 6-разрядных ASCII-символа (24 бита). То есть биты исходного символа распределяются по нескольким выходным символам. Результирующий текст состоит только из символов, общих для кодировок US-ASCII, EBCDIC и ISO 646. Эти символы (их всего 65) известны как *алфавит Base64* (табл. 9-4).

При декодировании символы закодированного сообщения, не входящие в алфавит Base64, игнорируются. Это позволяет добавлять в закодированный поток символы CR и LF (возврат каретки и перевод строки), что гарантирует успешное прохождение сообщение через любые промежуточные почтовые шлюзы.

Таблица 9-4. *Алфавит Base64*

Код	Символ	Код	Символ	Код	Символ	Код	Символ
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v		
14	O	31	f	48	w	Нет	=
15	P	32	g	49	x		
16	Q	33	h	50	y		

Binary

Значение Binary означает отсутствие кодирования, возможность появления не-ASCII-символов, а также то, что строки могут быть очень длинные, и SMTP не сможет успешно передать сообщение.

7Bit

Значение 7Bit означает отсутствие кодировки, использование только ASCII-символов и длину строк, достаточную для успешной передачи сообщения по протоколу SMTP.

8Bit

Значение 8Bit означает отсутствие кодировки, возможность наличия не-ASCII-символов и короткие строки.

X-Token

Значение X-Token означает, что способ кодирования согласован отправляющим и принимающим SMTP-серверами в частном порядке.

Цель явного определения форматов Binary, 7Bit и 8Bit, которые не подразумевают никакого кодирования, в том, чтобы будущие реализации могли распознать природу нижележащих данных и правильно их обработать.

Стандарт S/MIME

Стандарт S/MIME (Secure/Multipurpose Internet Mail Extensions) описывает протокол защиты информации в сообщениях электронной почты. Конфиденциальность обеспечивается при помощи шифрования сообщений, а аутентификация — при помощи цифровых подписей. В настоящее время S/MIME развивается под патронажем рабочей группы IETF. Подробнее о ней — по адресу *http://www.ietf.org/html.charters/smime-charter.html*. Там Вы сможете найти несколько черновых RFC, предложенных на обсуждение этой рабочей группой. Дополнительную информацию о S/MIME можно получить по адресу *http://rsa.com/smime*.

В стандарте S/MIME предлагается использовать симметричный алгоритм для шифрования содержимого сообщения и алгоритм с открытым ключом — для шифрования ключа. В силу экспортных ограничений США, S/MIME рекомендует использовать алгоритм RC2 в режиме CBC с 40-битным ключом. (Алгоритм RC2 обсуждался в главе 5.) Если реализуется отдельная версия приложения, ограниченная для использования только на территории США, то для шифрования содержимого сообщений рекомендуется использовать DES или Triple DES.

Для входящих сообщений реализация S/MIME должна обеспечивать поддержку алгоритмов MD2 и MD5 (тоже описаны в главе 5). Для исходящих сообщений в качестве алгоритма дайджеста сообщений рекомендован MD5.

Также для входящих сообщений реализация S/MIME должна обеспечивать поддержку сертификатов X.509 версии 1. Дополнительно стандарт рекомендует для входящих сообщений поддерживать сертификаты X.509 версий 2 и 3. Для исходящих сообщений рекомендованы сертификаты X.509 версии 1.

Для исходящих сообщений необходима поддержка RSA с ключами длиной от 512 до 1024 бит, что попадает под ограничения правительства США. Стандарт S/MIME рекомендует поддерживать ключи длиной до 2048 бит.

В S/MIME применяются сертификаты X.509. Вся необходимая инфраструктура для выдачи и обслуживания таких сертификатов реализована компанией VeriSign. Компания RSA Data Security продает инструментальный набор TIPEM, который содержит объектный код на языке C, реализующий цифровые подписи, цифровые сертификаты и форматирование сообщений. Многие компании, в том числе Microsoft, Lotus, VeriSign, Netscape и Novell, объявили о поддержке S/MIME.

MOSS, или PEM-MIME

Спецификация MOSS (MIME Object Security Services) это черновой стандарт (RFC 1848), предназначенный для замены PEM (Privacy Enhanced Mail). (Описание PEM см. в следующем разделе.) Напомню, согласно спецификации MIME, сообщение электронной почты состоит из нескольких частей. В стандарте S/MIME ко всему сообщению, то есть ко всем его частям, применяется один метод защиты. MOSS дает возможность шифровать разные части сообщения при помощи разных алгоритмов или одним алгоритмом, но на разных ключах. Иногда MOSS называют PEM-MIME. Единственный часто отмечаемый недостаток MOSS — неполная проработка стандарта. Из-за этого не исключена ситуация, когда сообщение, созданное одним MOSS-совместимым почтовым агентом, будет невозможно прочитать в другом, также MOSS-совместимом.

PEM — защищенная почта

Началом разработки PEM (Privacy Enhanced Mail) можно считать труды исследовательской группы Privacy and Security под руководством IAB, опубликованные в середине 80-х годов. Описание PEM находится в RFC 1421.

PEM предоставляет следующие возможности:

- конфиденциальность, когда сообщение понятно только тем, кому оно адресовано. Это касается не только пересылки сообщения (например, по локальной сети), но также всего времени хранения сообщения в почтовом ящике получателя;
- услуги аутентификации, гарантирующие, что сообщение отослано именно тем отправителем, который указан в сообщении;
- обеспечение целостности, гарантирующее, что сообщение принято именно в том виде, в котором было отправлено, то есть оно не менялось;
- гарантия того, что автор сообщения позднее не сможет заявить, будто никогда не отправлял такое сообщение. Другими словами, можно доказать, что полученное сообщение послал конкретный абонент.

Обратите внимание, PEM предполагает, что в качестве транспортного механизма используется SMTP, а сообщения представлены в формате RFC 822. Вся информация, относящаяся к PEM, заключена между двух строк — «-----*BEGIN PRIVACY-ENCHANCED MESSAGE*-----» и «-----*END PRIVACY-ENCHANCED MESSAGE*-----». Чтобы гарантировать успешную

доставку сообщения, пользовательский агент использует каноническую форму содержимого, то есть сообщение переводится в ASCII-текст, а в конце каждой строки вставляются символы CR и LF. После того как сообщение переведено в канонический вид, оно шифруется и подписывается, а затем все переводится в ASCII при помощи кодировки Base64 (см. раздел о MIME выше в этой главе).

Для шифрования сообщений PEM использует DES в режиме CBC. Симметричный ключ для DES рассылается каждому получателю зашифрованным при помощи алгоритма цифровой подписи с открытым ключом. Для проверки подлинности открытых ключей используются сертификаты X.509.

Некоммерческая реализация PEM, названная RIPEM, свободно распространяется компанией RSA Data Security. Подробности — на сервере *ftp://ftp.rsa.com/pub/*.

Система PGP

Система PGP (Pretty Good Privacy) использует криптографию с секретным и с открытым ключом, а также алгоритмы дайджеста сообщений (все описаны в главе 5). PGP гарантирует конфиденциальность сообщения (только адресат сумеет его расшифровать и прочитать) и аутентификацию сообщения (получатель может быть уверен в личности отправителя). Но помните, формально PGP используется для защиты файлов, а не электронной почты.

Конфиденциальность обеспечивается благодаря шифрованию сообщения симметричным блочным шифром IDEA (International Data Encryption Algorithm) при помощи случайно выработанного 128-битного ключа. Этот ключ затем помещается в заголовок сообщения, который шифруется алгоритмом RSA с открытым ключом. При этом используется открытый ключ получателя электронной почты. Адресат расшифровывает заголовок при помощи своего личного ключа и извлекает из него секретный ключ.

Аутентификация обеспечивается цифровыми подписями (см. главу 5). Сообщение обрабатывается посредством алгоритма MD5, который создает 128-битный хеш-код. Этот хеш-код шифруется алгоритмом RSA (используется личный ключ отправителя), после чего помещается в начало сообщения. Получатель извлекает зашифрованный хеш-код и расшифровывает его открытым ключом отправителя. Также адресат расшифровывает само сообщение и независимо вычисляет его хеш-код (посредством MD5). Если отправитель истинный, то вычисленный хеш-код совпадет с расшифрованным.

В PGP используются три вида ключей:

- обычный — 384 бита;
- коммерческий — 512 бит;
- военный — 1 024 бита.

Открытый и личный ключи пользователя хранятся на диске компьютера в файлах, называемых *keyrings*. Эти файлы зашифрованы алгоритмом IDEA, и каждый пользователь для доступа к ключам обязан ввести пароль. По введенному паролю при помощи алгоритма MD5 вычисляется хеш-код, который применяется для расшифровки ключей. Однако у PGP имеется недостаток, свойственный всем криптографическим системам с открытым ключом: нельзя гарантировать, что открытый ключ истинный и принадлежит конкретному пользователю.

В цифровой подписи, вычисляемой PGP, и в зашифрованном сообщении старший бит каждого байта может быть равен 1. Такие сообщения не всегда успешно проходят через почтовые серверы. В PGP предусмотрена схема представления каждых трех символов в виде набора их четырех символов, старший бит которых равен 0.

Система PGP реализована на многих платформах, в том числе Microsoft Windows, DOS, UNIX, Macintosh и VMS. Существуют как бесплатные, так и коммерческие версии PGP. На некоторые из них наложены экспортные ограничения США. Международная версия PGP, названная PGP5I, создана путем распечатки исходного кода PGP5 и экспорта полученной книги, а не самого программного кода. Затем текст программы ввели в компьютер через сканер. Эта версия попадает под экспортные ограничения США.

Система PGP описана в RFC 2015 и 1991. Подробности — по адресу *http://www.pgp.com*.

Списки рассылки и серверы рассылки

В предыдущих разделах описывались различные протоколы передачи электронной почты. Примером их использования *служат серверы рассылки* (list server) и *списки рассылки* (mailing list).

Серверы рассылки предоставляют пользователям удобный способ сбора информации и участия в дискуссиях. Пользователю достаточно уметь отправлять и принимать электронную почту. Серверы рассылки лишь распространяют электронную почту по списку рассылки. На одном сервере могут существовать несколько списков. Сообщения, отправляемые

серверу рассылки, попадают ко всем членам списка. Серверы рассылки существуют для разных операционных систем, включая многие разновидности UNIX и Windows NT. Каждая запись в списке сервера рассылки имеет длину до восьми символов. (В целях совместимости эта длина не зависит от используемой операционной системы.)

Чтобы присоединиться к списку рассылки или исключить себя из него, человеку достаточно отправить сообщение на специальный адрес сервера рассылки. Обычно такие письма «читают» программы, они автоматически обрабатывают запросы и оставляют администраторам только сообщения нестандартной формы. Для пользователя нет удобного способа узнать у почтового клиента или сервера рассылки «Членом каких списков рассылки я являюсь?». Приходится самостоятельно отслеживать такую информацию при помощи дневника или как-то еще. Существуют команды, позволяющие пользователю исключить себя из всех списков рассылки. Периодически серверы рассылки посылают пробные сообщения всем адресатам списка. Ответ о невозможности доставки конкретного сообщения означает, что адрес этого клиента больше недействителен. Те же, кто получат такие сообщения, вправе их просто игнорировать. Полученные ответы о невозможности доставки сообщений применяются сервером рассылки для «подчистки» списка рассылки. Если, например, пользователь уезжает в отпуск, он может указать серверу, чтобы тот сохранял подписку, но не посылал никаких сообщений. Все взаимодействие с сервером рассылки, в том числе задание параметров, выполняется посредством электронной почты.

Единственный недостаток работы с серверами рассылки в том, что почтовый ящик пользователя иногда переполняется сообщениями, рассылаемыми по списку. Чтобы избежать этого, подписчик может вместо самих документов заказать только сводку (в одном письме) о том, что отправлено по списку рассылки. Преимущество же очевидно — это достаточно простой способ получения множества интересной и полезной информации. И все только посредством электронной почты: пользователю не надо изучать новые системы и их команды.

Различные списки рассылки можно отыскать при помощи поисковых машин для Web. Также удобны протоколы поиска и сбора информации, например Gopher и Veronica. Кроме того, пользователь может подписаться на получение (по электронной почте) «списка списков», содержащего информацию о создании каждого нового списка рассылки. Тематика списков рассылки — самая разная, включая программирование, общественную деятельность и развлечения.

Новости и Usenet

Usenet позволяет пользователям участвовать в публичных дискуссиях — *группах новостей* (newsgroup). Сообщение, отправленное в группу новостей, рассылается всем пользователям, подписанным на нее. Usenet — так называется слабо связанная сеть, в которой компьютеры обмениваются по электронной почте сообщениями, имеющими тему или заголовок. Заголовки указывают, к какой группе новостей оно принадлежит. Сообщения называют *статьями* (article). Взаимный обмен статьями между серверами и между серверами и клиентами осуществляется при помощи протокола NNTP (Network News Transfer Protocol), описанного в следующем разделе.

Серверы рассылки отличаются от Usenet способом доступа и получения информации конечными пользователями. Так в Usenet пользователь должен запустить просмотровую программу, а затем загрузить сами сообщения. Подписчик имеет возможность ознакомиться только с заголовками статей и принять лишь избранные. А при работе с сервером рассылки пользователь получает электронную почту напрямую в свой почтовый ящик.

Группы новостей не предполагают централизованного администрирования. Названия группам новостей присваивают, исходя из следующего соглашения: имя состоит из нескольких частей, разделенных точками. Первая часть описывает категорию группы новостей (например, *soc* в названии *soc.culture.india*). В табл. 9-5 описаны основные категории групп новостей (начальные элементы названия).

Таблица 9-5. *Основные категории групп новостей*

Название категории	Описание
biz	Вопросы бизнеса
comp	Компьютеры, программное обеспечение, сети и другие вопросы, связанные с компьютерами
sci	Вопросы научного характера
soc	Вопросы социологии и культуры
talk	Затяжные дискуссии и дебаты
news	Обсуждение новостей и другой информации
rec	Творческие виды деятельности, например изобразительное искусство, хобби и т. д.
misc	Вопросы, не попадающие ни в какую другую категорию
alt	Вопросы ограниченного хождения; здесь содержимое гораздо разнообразней, чем в других категориях

Подробности — в RFC 1036.

Протокол NNTP

Протокол NNTP (Network News Transfer Protocol) используется для:

- размещения статей на сервере;
- поиска и просмотра списка статей сервера;
- обмена статьями между серверами.

Протокол NNTP очень похож на SMTP. Так, например, сообщения и команды NNTP используют набор символов ASCII. Команды — это строки ASCII-текста, в которых за названием команды следуют дополнительные параметры. Каждая команда заканчивается ASCII-символами CR и LF. Также как в SMTP, NNTP-ответы состоят из трехзначного кода и ASCII-текста. Первый применяется для управления автоматом, реализующим протокол, тогда как текст можно демонстрировать пользователю (человеку). Ответы также завершаются символами CR и LF. Если при ответе на команду сервер посылает дополнительные данные, например текст статьи, то каждая строка завершается последовательностью CR/LF. Конец ответа обозначается строкой с одним символом «точка» (.) и символами CR/LF. Также как в SMTP, используется «дополнение точками»: при передаче, если строка начинается с точки, то эта точка будет продублирована и в конце добавлены символы CR/LF. Программа-приемник лишнюю точку отсекает.

Каждый NNTP-ответ содержит трехзначное число — код возврата. Первая цифра отражает успех или неудачу сообщения (табл. 9-6).

Таблица 9-6. *Первая цифра кода возврата NNTP*

1xx	Информационное сообщение
2xx	Команда выполнена успешно
3xx	Команда выполнена успешно, но не полностью; посылайте остальные данные
4xx	Команда корректна, но не может быть исполнена
5xx	Недопустимая или неподдерживаемая команда; серьезная ошибка

Вторая цифра кода возврата указывает категорию выполняемого действия (табл. 9-7).

Протокол NNTP содержит 15 команд, которые клиент может отдавать серверу. Обратите внимание, NNTP-сервер иногда способен действовать как клиент, например при ретрансляции NNTP-сообщений. Команды протокола NNTP описаны в табл. 9-8. В квадратных скобках указаны необязательные параметры; угловые скобки проставлены там, где требует синтаксис.

Таблица 9-7. *Вторая цифра кода возврата NNTP*

x0x	Подключение, настройка и дополнительные сообщения
x1x	Выбор группы новостей
x2x	Выбор статьи
x3x	Действия по распространению информации
x4x	Размещение информации на сервере
x8x	Нестандартные дополнения, зависящие от реализации
x9x	Отладочный вывод данных

Таблица 9-8. *NNTP-команды*

Команда	Аргументы	Описание
ARTICLE	*<идентификатор_сообщения>*	Запрашивает текст сообщения с заданным идентификатором
ARTICLE	[*nnnn*]	Запрашивает текст сообщения с заданным номером статьи
BODY	*<идентификатор_сообщения>*	Запрашивает тело, соответствующее сообщению с заданным идентификатором
BODY	[*nnnn*]	Запрашивает тело, соответствующее сообщению с заданным номером статьи
GROUP	Название группы	Выбирает указанную группу в качестве текущей и возвращает номера первой и последней статьи, а также количество статей в группе
HEAD	*<идентификатор_сообщения>*	Запрашивает заголовок сообщения с заданным идентификатором
HEAD	[*nnnn*]	Запрашивает заголовок сообщения с заданным номером статьи
HELP	Нет	Выдает подробный список команд, реализуемых сервером (в формате ASCII)
IHAVE	*<идентификатор_сообщения>*	Клиент сообщает серверу, что у него есть сообщение с указанным идентификатором; при положительном ответе клиент передает серверу все сообщение; при отрицательном клиент не должен передавать сообщение
LAST	Нет	Перемещает указатель текущей статьи на предыдущую статью в той же группе новостей
LIST	Нет	Возвращает список названий групп; для каждой группы возвращаются идентификаторы первого и последнего сообщения и индикатор возможности размещения информации в этой группе

Таблица 9-8. *NNTP-команды* (продолжение)

Команда	Аргументы	Описание
NEWSGROUPS	Дата, время, [GMT], [<*количество*>]	Возвращает список групп новостей, созданных после определенных даты и времени; время указано во временной зоне сервера, но можно — и в формате GMT; последний параметр (количество) — не обязательный, а используется для ограничения объема возвращаемого списка групп
NEWNEWS	Группы новостей, дата, время	Возвращает идентификаторы новых сообщений (появившихся после указанных даты и времени) из перечисленных групп
NEXT	Нет	Перемещает указатель текущей статьи на следующую статью в той же группе новостей
POST	Нет	Используется для размещения сообщений на сервере; сервер может ответить положительно или отрицательно
QUIT	Нет	Завершает сеанс связи
SLAVE	Нет	Сообщает серверу, что клиент — это сервер в режиме ведомого
STAT	*Идентификатор_сообщения*	Аналогична команде ARTICLE, но не возвращает никакого текста; обычно используется для позиционирования указателя текущей статьи
XOVER	Нет	Не описана в RFC 977; команда стала стандартом «де-факто», используется для получения заголовков нескольких статей при помощи одной команды

NNTP описан в RFC 977.

Будущие приложения

В силу повсеместной стандартизации электронная почта становится необычайно важным приложением. В будущем ожидается появление стандартного метода хранения и обмена информацией из адресной книги. Кроме того, ощущается необходимость создания приложений для обмена через Интернет информацией из электронных календарей и планировщиков. Первым шагом к созданию таких приложений можно считать протокол ACAP (Application Configuration Access Protocol), описанный в RFC 2244. Вероятно, этот протокол найдет широкое применение.

Ссылки

POP

RFC 2195, «IMAP/POP AUTHorize Extension for Simple Challenge/Response»

RFC 1957, «Some Observations on Implementations of the Post Office Protocol (POP3) »

RFC 1939, «Post Office Protocol Version 3»

RFC 1734, «POP3 AUTHentication Command»

IMAP

RFC 2195, «IMAP/POP AUTHorize Extension for Simple Challenge/Response»

RFC 2193, «IMAP4 Mailbox Referrals»

RFC 2180, «IMAP4 Multi-Accessed Mailbox Practice»

RFC 2060, «Internet Message Access Protocol—Version 4»

RFC 1733, «Distributed Electronic Mail Models in IMAP4»

RFC 1732, «IMAP4 Compatibility with IMAP2 and IMAP2BIS»

RFC 1731, «IMAP4 Authentication Mechanisms»

SMTP

RFC 2197, «SMTP Service Extension»

RFC 2034, «SMTP Enhanced Error Codes»

RFC 1985, «SMTP Service Extension—ETRN»

RFC 1891, «SMTP Delivery Status Notifications»

RFC 1870, «SMTP Size Declaration»

RFC 1869, «SMTP Service Extensions»

RFC 1845, «SMTP Checkpoint/Restart»

RFC 1830, «Binary and Large Message Transport»

RFC 0822, «Standard for the Format of ARPA Internet Text Messages»

RFC 0821, «Simple Mail Transfer Protocol»

MIME

RFC 2231, «MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations»

RFC 2184, «MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations»

RFC 2112, «The MIME Multipart/Related Content-Type»

RFC 2049, «Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples»

RFC 2048, «Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures»

RFC 2047, «MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text»

RFC 2046, «Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types»

RFC 2045, «Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies»

RFC 1927, «Suggested Additional MIME Types for Associating Documents»

RFC 1896, «The Text/Enriched MIME Content-Type»

RFC 1741, «MIME Content Type for BinHex Encoded Files»

RFC 1740, «MIME Encapsulation of Macintosh Files—MacMIME»

RFC 1641, «Using Unicode with MIME»

RFC 1556, «Handling of Bi-Directional Texts in MIME»

RFC 1437, «The Extension of MIME Content-Types to a New Medium»

RFC 1428, «Transition of Internet Mail from Just-Send-8 to 8-bit-SMTP/MIME»

RFC 1344, «Implications of MIME for Internet Mail Gateways»

Безопасность электронной почты

RFC 2312, «S/MIME Version 2 Certificate Handling»

RFC 2311, «S/MIME Version 2 Message Specification»

RFC 2015, «MIME Security with Pretty Good Privacy (PGP)»

RFC 1991, «PGP Message Exchange Formats»

RFC 1848, «MIME Object Security Services»

RFC 1847, «Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted»

RFC 1424, «Privacy Enhancement for Internet Electronic Mail: Part IV: Key Certification and Related Services»

RFC 1423, «Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifiers»

RFC 1422, «Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management»

RFC 1421, «Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures»

Другие ссылки

RFC 2244, «ACAP—Application Configuration Access Protocol»

RFC 1211, «Problems with the Maintenance of Large Mailing Lists»

RFC 1036, «Standard for Interchange of USENET Messages»

RFC 977, «Network News Transfer Protocol»

RFC 976, «UUCP Mail Interchange Format Standard»

RFC 0402, «ARPA Network Mailing Lists»

ГЛАВА 10

Передача файлов и файловые системы

Начало этой главы посвящено протоколам, используемым для передачи файлов между двумя компьютерными системами. До появления World Wide Web протоколы FTP (File Transfer Protocol) и TFTP (Trivial File Transfer Protocol) чаще всего применялись для передачи файлов по Интернету. Сейчас, конечно, для поиска файла и его сохранения в определенном месте можно использовать Web-браузер.

Далее обсуждаются файловые системы, в том числе CIFS (Common Internet File System) и WebNFS (NFS обозначает Network File System). Обе эти системы, созданные для совместного использования файлов в локальной сети, со временем стали применять (с соответствующими изменениями) в Интернете.

Протокол FTP

Протокол FTP (File Transfer Protocol) — популярный и относительно безопасный способ перемещения файлов между различными компьютерами. В качестве транспортного механизма для передачи данных FTP применяет протокол TCP. Его средствами пользователь может предъявлять «удостоверения личности» серверу, а затем просматривать папки и передавать файлы в обоих направлениях. FTP позволяет передавать данные как между клиентом и FTP-сервером, так и между двумя удаленными компьютерами.

FTP-модель

Спецификация RFC 959 определяет FTP-модель в терминах модели «пользователь — сервер», как показано на рис. 10-1.

FTP — достаточно сложный протокол, устанавливающий два TCP-соединения: по одному передаются данные, а по другому — команды. Каж-

дый FTP-хост снабжен модулем интерпретатора протокола (Protocol Interpreter, PI), который отвечает за исполнение FTP-команд и интерпретацию ответов. Кроме того, каждый FTP-хост содержит модуль передачи данных (Data Transfer, DT), отвечающий за обработку информации. Модули PI и DT одного хоста взаимодействуют некоторым способом, определение которого выходит за рамки RFC 959. Протокол FTP в качестве PI использует протокол Telnet — посредством выполнения независимого модуля Telnet или полной его реализацией. FTP-команды передаются открытым текстом, как и FTP-ответы. Последние состоят из трехзначного числа, за которым следует некоторый текст. Число используется программой для определения очередного действия. Текст же предназначен пользователю. На каждую команду генерируется один или несколько ответов.

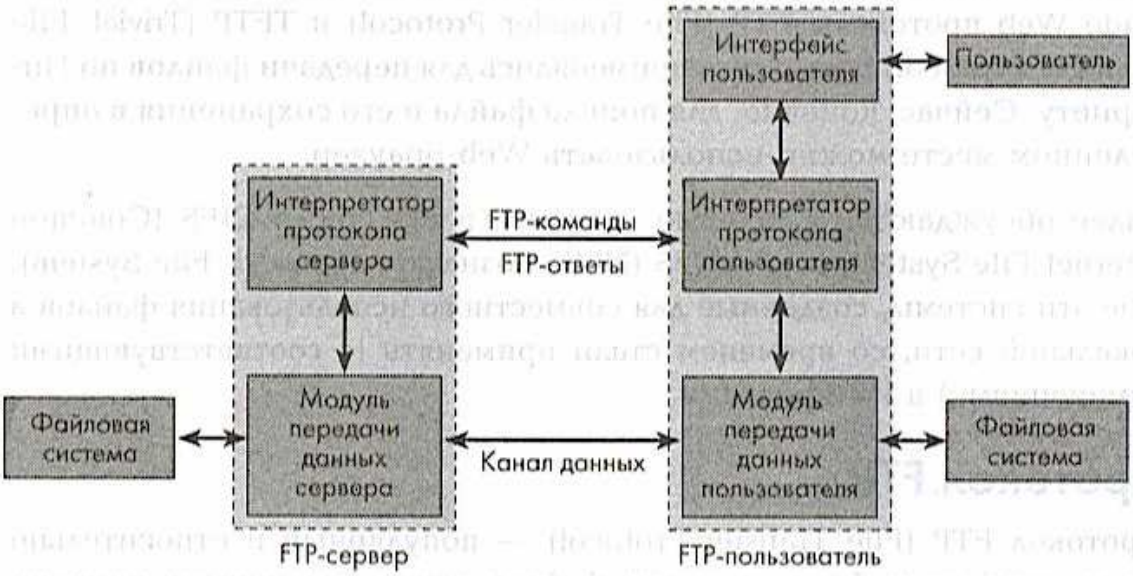


Рис. 10-1. *FTP-модель «пользователь — сервер»*

Типы данных

При передаче файлов по протоколу FTP тип представления данных источника преобразуется к нейтральному формату (на время передачи). Нейтральный формат затем преобразуется к типу данных, характерному для принимающего компьютера. Если общее число компьютеров, поддерживаемых FTP, обозначить через n, то матрица прямого преобразования потребовала бы реализации $n \times n$ преобразований. Использование нейтрального формата — например, NVT-ASCII (Network Virtual Terminal ASCII) — позволяет сократить число реализованных преобразований до $2 \times n$.

FTP определяет следующие типы данных:

- NVT-ASCII;
- EBCDIC — используется в основном на мэйнфреймах;
- IMAGE — описывает представление двоичных исполняемых файлов в виде последовательностей 8-разрядных байтов, или октетов;
- LOCAL — описывает представление двоичных исполняемых файлов с указанным размером байта.

Чтобы помочь серверу, клиент способен задать тип данных посредством FTP-команды TYPE.

Структура файлов

Для поддержки большего числа разнородных компьютеров FTP определяет несколько файловых структур следующим образом:

- *File* (файл) — файл передается как сплошной поток байт;
- *Record* (запись) — файл передается как последовательность записей;
- *Page* (страница) — файл передается в виде блоков данных; этот режим используется для файлов прямого доступа (в них мало данных и много неиспользованного пространства).

Режимы передачи

Протокол FTP определяет три возможных режима передачи данных.

- *Поточный режим* (stream mode) — файл рассматривается как последовательность байт. Этот режим допустимо использовать с любым типом данных. При файловой структуре типа Record применяются специальные двухбайтные управляющие символы, обозначающие конец записи (EOR) и конец файла (EOF). При файловой структуре типа File конец файла обозначается закрытием канала данных передающим компьютером.
- *Блочный режим* (block mode) — файл передается как набор блоков, каждому из которых предшествует заголовок, содержащий размер блока и дескриптор. Дескриптор определяет, является ли этот блок последним в файле или в записи, а также содержит бит, показывающий надежность данных. Этот бит используется в случае, когда чтение данных большого объема, например определение геологических или атмосферных характеристик, заканчивается ошибкой среды передачи. Дескриптор может также указать, является ли данный блок

указателем перезапуска (Restart Marker). Это позволит при необходимости повторной передачи выполнять ее от момента приема последнего указателя перезапуска.

- *Сжатый режим* (compressed mode) используется для эффективной передачи файлов и оптимального использования сети. Механизм сжатия достаточно прост: строка из x повторений одного байта заменяется двумя байтами.

Восстановление при ошибках

Заботу о представлении данных в нужном порядке, без пропусков или повторений FTP возлагает на TCP. Сам FTP позволяет заново передать файл, если использовался блочный либо сжатый режим передачи.

FTP-команды и ответы

FTP-команды — это ASCII-строки, дополненные необязательными параметрами. Существуют FTP-команды для предоставления «удостоверения личности» пользователя, управления папками, сохранения и запроса файлов, задания типов данных и режима передачи, определения состояния сервера и т. д. Детали Вы можете уточнить в RFC 959, где также указано, что в реализации протокола FTP лишь часть команд — обязательные, остальные же — дополнительные.

Все FTP-команды требуют ответа от сервера, причем некоторые вызывают несколько ответов. Каждый ответ состоит из трехзначного числового кода, после которого следует текстовая строка. Последняя может отображаться для удобства пользователя, а числовой код позволяет управлять конечным автоматом.*

Спецификация RFC 2228 описывает набор необязательных FTP-команд, которые могут выполняться FTP-клиентом и FTP-сервером. Эти команды, если они реализованы и клиентом и сервером, обеспечивают аутентификацию (сервер доподлинно будет знать, что клиент именно тот, кем представляется), целостность данных и их конфиденциальность как в канале данных, так и в канале команд (это два TCP-соединения, используемые протоколом FTP для передачи данных и команд). Таким образом, и сервер и клиент сумеют выявить искажение данных или команд третьей стороной. Данные и команды можно шифровать, так что третья сторона, получив доступ к TCP-пакету, не сможет понять его смысл.

---

* Работу протокола FTP можно описать в терминах конечных автоматов. — *Прим. перев.*

Ключ для шифрования (о шифровании см. главу 5) предустанавливается или распределяется по каналу, внешнему по отношению к FTP, например, курьерской или электронной почтой.

FTP-серверы «прослушивают» 21-й порт TCP. Протокол FTP описан в RFC 959.

Анонимный FTP

Некоторые узлы работают как хранилища информации, предназначенной для широкого распространения, и доступны без ограничений. Протокол Анонимный FTP (Anonymous FTP) обеспечивает свободный доступ к такой информации. Для этого пользователю достаточно знать лишь имя архивного узла и начало имени файла или путь к каталогу. Архивный узел создает специальную учетную запись с именем пользователя «anonymous» (анонимный), которая позволяет лишь зарегистрироваться, просмотреть ограниченное число каталогов и считать (но не записать) определенное количество файлов. Архивный узел в качестве пароля может принять любую строку, содержащую слово «guest» (гость), либо потребует указать адрес электронной почты. Протокол Анонимный FTP описан в RFC 1635.

Протокол FTP и межсетевые экраны

Как сказано выше, протокол FTP использует два TCP-соединения: одно для данных, другое для команд. Клиент передает команду на 20-й порт FTP-сервера. Для пересылки данных сервер, используя свой порт 21, инициализирует TCP-соединение с тем же самым портом, через который клиент отправлял запрос. Так как разные клиенты обычно пересылают запросы не через одинаковые порты, то невозможно заранее определить, с каким портом FTP-сервер будет инициализировать TCP-соединение. В этом суть проблемы, так как межсетевые экраны обычно блокируют подключения к неизвестным портам. Спецификация RFC 1579 предлагает модификации FTP-клиентов и серверов, благодаря которым эту проблему можно разрешить.

Протокол TFTP

Как видно из названия, протокол TFTP (Trivial File Transfer Protocol) — простой способ передачи файлов между двумя компьютерами. Для управления этим процессом TFTP применяет протокол UDP. Он не содержит механизма идентификации (передачи «удостоверений личности» — имени пользователя и пароля); следовательно, TFTP применяется только в областях, доступных любому пользователю. Протокол TFTP позво-

ляет только перемещать файлы и не допускает просмотр списка каталогов, навигацию по ним и т. д.

Протокол TFTP передает данные блоками фиксированной длины — по 512 байт. Он использует окно фиксированного размера, равное одному блоку данных, то есть необходимо подтверждение приема каждого блока, прежде чем отправлять следующий. Заметим, что пакетами обмениваются оба задействованных компьютера. Один передает блоки данных и принимает подтверждения их успешного приема. Другой принимает блоки данных и передает подтверждения. Каждый компьютер должен выдерживать паузу и повторять передачу, если ожидаемое подтверждение или блок данных не получены. Почти все ошибки останавливают процесс передачи. Если отправка файлов прервалась, то нет возможности ее перезапустить и скопировать оставшуюся часть файла. Спецификация RFC 1783 дополняет протокол TFTP механизмом согласования размера блока для передачи файлов. Если этот механизм применяется, то значение размера блока может устанавливаться в пределах от 8 до 65 464 байт. Спецификация RFC 1784 описывает дополнительный механизм согласования длительности пауз — от 1 до 255. Также в RFC 1784 описано, как определить размер передаваемого файла.

Обычно реализации TFTP-сервера предоставляют некоторые средства задания путей, к которым разрешен доступ по протоколу TFTP. Это разумно, учитывая отсутствие защиты для TFTP. Реализации TFTP просты и достаточно компактны для поставки в ППЗУ (программируемое постоянное запоминающее устройство — PROM), которые используются в бездисковых рабочих станциях.

Серверы TFTP «слушают» 69-й порт UDP. Протокол TFTP описан в RFC 1350, а его дополнения, позволяющие расширить возможности TFTP, — в RFC 1785, 1784, 1783 и 1782.

Протокол CIFS

Протокол CIFS (Common Internet File System) определяет стандарт удаленного доступа к файлам через Интернет и интрасети. Он описывает способы обращения клиентов к файловым службам сервера. Расширение CIFS, названное Common Internet File System/Enterprise, описывает способы обнаружения CIFS-серверов клиентом, а также протоколы доступа к другим службам, например к службам печати. Протокол CIFS основан на протоколе SMB (Server Message Block) и доступен для целого ряда платформ, включая Microsoft Windows 95, Microsoft Windows NT и многие модификации UNIX.

Небольшую путаницу при использовании CIFS вносит наличие нескольких диалектов технологии SMB. Однако при установлении сеанса связи клиент и сервер согласуют используемый диалект. Причем CIFS поддерживает все существующие диалекты SMB, так же как WebNFS поддерживает NFS (Network File System) компании Sun Microsystems версий 1, 2 и 3. Еще стоит упомянуть о том, что каждый диалект обладает дополнительными возможностями. Поэтому простое указание диалекта при согласовании не обязательно полностью определяет набор свойств. Некоторые диалекты SMB были приняты Open Group (прежде X/OPEN) в качестве стандартов. Компания Microsoft настаивает, чтобы CIFS был принят IETF в качестве Информационного RFC (Informational RFC). Протокол CIFS лишь дополняет HTTP и FTP, и не следует его рассматривать как замену какого-либо из них.

Ниже перечислены некоторые преимущества протокола CIFS.

- Он обеспечивает совместное использование файлов с возможностью блокировки, то есть несколько клиентов одновременно могут получать доступ и изменять одни и те же ресурсы. За счет стратегии полного использования кэша, те же самые механизмы позволяют клиентам применять агрессивные методы *опережающего чтения* (read-ahead) и *отложенной записи* (write-behind).
- Протокол пытается кэшировать файлы на стороне клиента, причем для работы с кэшем применяет элегантный механизм *временной блокировки* (opportunistic locking, сокращенно — oplock). Когда клиент обращается к файлу, CIFS открывает этот файл и пробует получить от сервера разрешение на временную блокировку. Если разрешение дано, то клиент кэширует запись данных (на стороне клиента), а также выполняет опережающее чтение. Когда другой клиент просит сервер открыть тот же файл, сервер посылает первому клиенту SMB-прерывание временной блокировки. После этого первый клиент может сохранить свои данные и дальше блокировать файл частями. С этого момента клиент прекращает кэшировать файл, и сервер способен предоставить второму клиенту требуемый доступ к файлу.
- Способен эффективно работать на медленных телефонных линиях связи.
- Поддерживает имена файлов и папок из любого набора символов (не только для английского языка).

- Может работать поверх протоколов, как ориентированных, так и не ориентированных на установление логических соединений (например TCP и UDP).
- Позволяет клиенту группировать запросы, а потом все разом передать серверу. Цель — минимизация числа обращений к каналу.

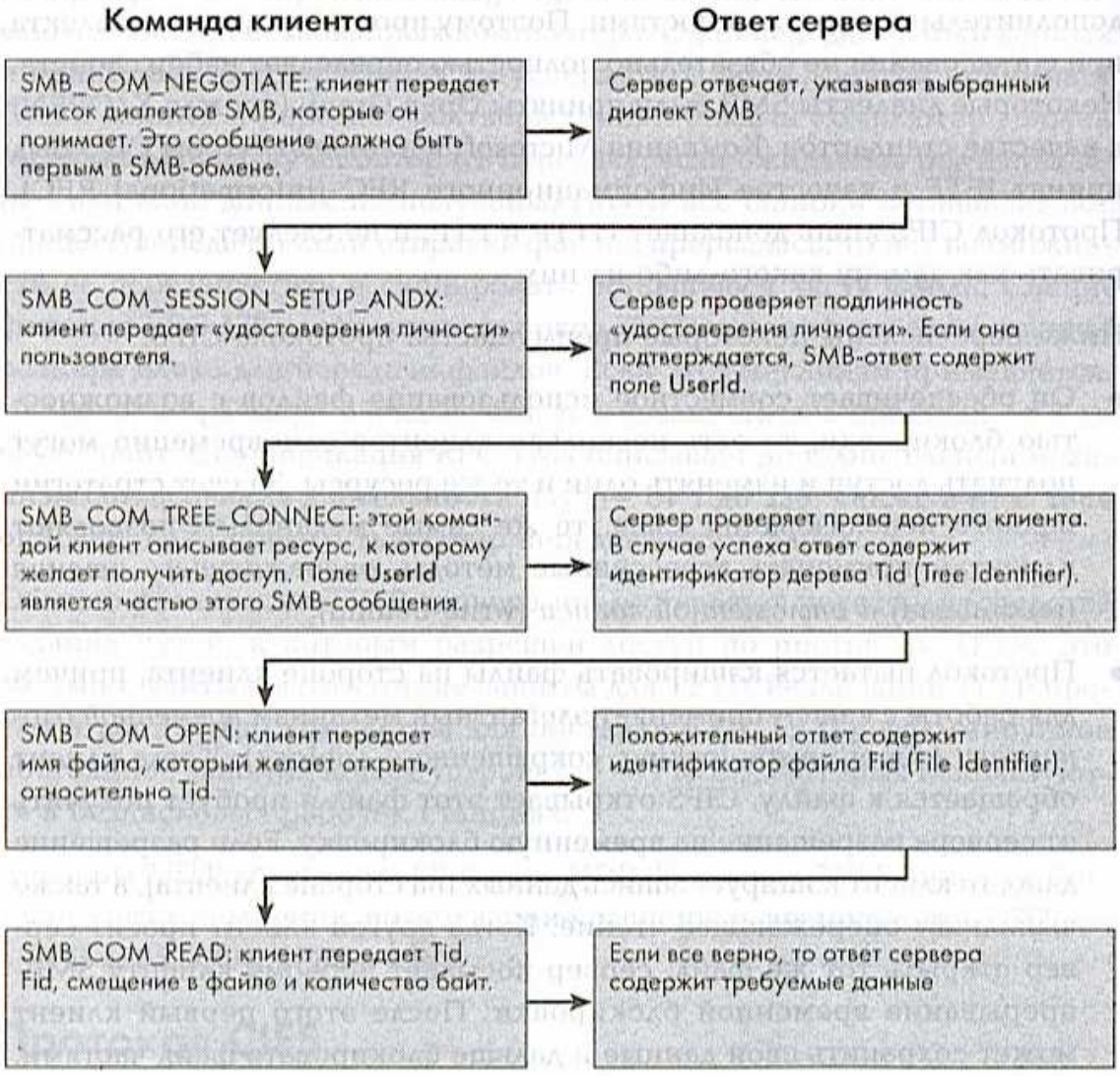


Рис. 10-2. *Некоторые SMB-команды и ответы в простом сценарии CIFS*

Для доступа к файлу на сервере клиент должен уметь анализировать URL (Uniform Resource Locator) и извлекать из него имя сервера. CIFS предоставляет клиенту два способа разрешения имени сервера в адрес: первый — разрешение имен NetBIOS; второй, и более предпочтительный, поскольку обеспечивает большую совместимость, — разрешение имен DNS (Domain Name System). (О DNS см. главу 6.)

CIFS позволяет считывать и записывать отдельные участки файла. Передача файла при помощи CIFS легко перезапускается с последней контрольной точки. FTP потребовал бы копировать все заново. Тот факт, что Интернет нельзя считать надежной средой (соединения часто прерываются), — дополнительный аргумент в пользу CIFS.

На рис. 10-2 изображен обмен SMB-сообщениями по довольно простому сценарию и пояснены некоторые важные поля, передаваемые и получаемые в SMB-командах и ответах. Многие детали на рисунке опущены; за более полной информацией обратитесь к последней спецификации CIFS.

Реализации протокола CIFS существуют в каждом компьютере, работающем под управлением Microsoft Windows (включая Windows 3.1, Windows 95 и Windows NT). Ряд производителей, в том числе Intergraph, Data General, Intel и Network Appliance, объявили о поддержке CIFS. Другие уже давно торгуют программным обеспечением, совместимым с CIFS. Это — IBM (IBM LAN Server), Novell, Banyan, Digital Equipment (PATHWORKS), AT&T (Advanced Server for UNIX), Samba (NetNames), Santa Cruz Operation (LAN Manager) и Hewlett-Packard (Advanced Server 9000).

Файловая система WebNFS

Файловая система WebNFS основана на технологии NFS, разработанной компанией Sun Microsystems. Файловая система NFS отображает удаленные диски или совместно используемые ресурсы на локальный драйвер. После публикации RFC 1094 стандартом считается NFS версии 2. Спецификация RFC 1813 описывает NFS версии 3. NFS основана на механизме удаленного вызова процедур (Remote Procedure Call, RPC) и часто описывается как протокол без состояний, то есть типа запрос/ответ; если ответ получен или отправлен, то никакой информации о другом конце соединения сохранять не требуется.

Определение WebNFS описывает использование любой версии NFS (1, 2 или 3), применительно к Web. NFS версий 1 и 2 в качестве транспортного механизма использует UDP, тогда как NFS версии 3 применяет TCP. В исходном виде NFS плохо работает в сетях с большим временем задержки и низкой пропускной способностью, например телефонных подключениях к Интернету. Некоторая проблема NFS в том, что для получения номера порта NFS-сервера, к которому клиент может направить свой запрос, необходимо выполнить PRC-вызов службы отображения портов (port mapper service). Файловая система WebNFS отличается

от NFS тем, что WebNFS-сервер всегда «прослушивает» порт 2049, и клиент об этом знает. Еще один недостаток NFS — дополнительные затраты на выполнение RPC-вызова для подключения удаленной файловой системы и получения *указателя* (handle). В файловой системе WebNFS это решается посредством предопределенного указателя нулевой длины. Третья проблема состоит в том, что NFS требует от клиента указывать путь по частям и каждый раз получать новый указатель. Рассмотрим, например, случай, когда клиент пытается распечатать файл */folder1/folder2/myfile.txt*. Файловая система NFS сначала получает указатель на *folder1*, затем на *folder2* и, наконец, на *myfile.txt*. В файловой системе WebNFS, в отличие от NFS, клиент сразу указывает полный путь. В этом примере WebNFS может получить указатель на *myfile.txt* за один раз.

У WebNFS также есть некоторые преимущества перед FTP и HTTP. При восстановлении соединения после его потери WebNFS позволяет повторно открывать файлы и продолжать копирование с последней известной позиции. Серверы WebNFS лучше масштабируются, обслуживая больше клиентов и делая это быстрее, чем FTP- и HTTP-серверы. До тех пор пока функции выборочного чтения протокола HTTP не будут реализованы шире, за WebNFS останется позиция лидера, так как его клиентам доступно не только чтение файлов целиком, но и их заданных участков. Компания Sun заявляет, что WebNFS гораздо производительнее HTTP, и соотношение в скорости составляет 3:1 в пользу WebNFS. Однако, несмотря на все сказанное и сделанное, WebNFS следует рассматривать не как замену, а как дополнение HTTP и FTP. Компания Sun анонсировала планы по реализации WebNFS на языке Java.

Ссылки

http://www.cifs.com

ftp://ietf.org/internet-drafts/draft-leach-cifs-v1-spec-01.txt

RFC 2228, «FTP Security Extensions»

RFC 1813, «NFS version 3 Protocol Specification»

RFC 1785, «TFTP Option Negotiation Analysis»

RFC 1784, «TFTP Timeout Interval and Transfer Size Options»

RFC 1783, «TFTP Blocksize Option»

RFC 1782, «TFTP Option Extension»

RFC 1635 «How to Use Anonymous FTP»

RFC 1579, «Firewall Friendly FTP»

RFC 1350, «The TFTP Protocol»

RFC 1094, «NFS: Network File System Protocol Specification»

RFC 959, «File Transfer Protocol»

ГЛАВА 11

Текстовые конференции

Одно из основных свойств Интернета — возможность общения людей друг с другом. «Разговор» не всегда происходит в режиме реального времени, вместо этого часто используются текстовые сообщения, отправляемые через группы новостей или по электронной почте. В этой главе обсуждаются еще два метода текстовой интерактивной связи — *Internet Relay Chat* (IRC) и *многопользовательские миры* (multiuser dungeons, MUD).

Internet Relay Chat

IRC — это простая, текстовая система, позволяющая группе пользователей общаться в режиме реального времени. IRC берет свое начало в интерактивных «беседах» через электронные доски объявлений (Bulletin Board System, BBS).

На любом сервере IRC имеется много каналов («бесед»), в каждом из которых может участвовать группа клиентов. Тематика весьма разнообразна: вложение капитала, технические вопросы, игры, спорт, региональные проблемы и т. д. Скорее всего, Вы найдете подходящий IRC-канал для обсуждения любой интересующей Вас темы.

Каналы, по своей природе, намного динамичнее *списков рассылки* (mailing lists) или *групп новостей* (newsgroups). Учитывая такое непостоянство, некоторым виртуальным событиям назначают определенную дату и время, чтобы пользователь смог в нужный момент подключиться к каналу.

В простейшем случае группа IRC-клиентов общается через один и тот же IRC-сервер. Как правило, несколько IRC-серверов (соединенных в неориентированный граф без циклов) составляют IRC-сеть. Хотя IRC-клиент соединяется только с одним сервером IRC-сети, он имеет доступ к каналам и пользователям других серверов той же сети. Существует несколько самостоятельных IRC-сетей, например Eris-Free Net (EFnet) и Undernet. Технически возможно соединить эти отдельные сети с помощью шлюза, однако на практике этого делают.

Большинство пользователей предпочитает специальную программу — IRC-клиент, а не Telnet, как более удобную для ведения бесед и простую для работы с командами и макросами. *IRC-бот* (bot), или *робот*, — это особый вид IRC-клиента, созданный и управляемый компьютером, а не человеком.

Протокол IRC

Клиент-серверный протокол IRC описан в RFC 1459, он использует TCP/IP и порты с 6660 по 6669. Это 8-разрядный протокол, в большинстве команд которого применяется набор символов US-ASCII.

Протокол IRC допускает три различных метода связи: клиент — клиенту, один — многим (по списку, всему каналу, или по маске), а также широковещание (всем IRC-серверам этой сети).

Существует несколько диалектов протокола IRC. В Undernet, например, для связи клиент-сервер применяется обычный протокол, но для связи серверов — особый.

Протокол IRC предназначен, в основном, для передачи текстовых сообщений, однако, используя другие его диалекты, можно обмениваться двоичными файлами. Вариант такого диалекта — протокол DCC (Direct Client to Client).

Команды IRC

Протокол IRC содержит набор команд, которые условно разделяют на несколько категорий. Основные — это соединение сервера с другим сервером или клиентом, получение информации о пользователе или сервере, управление каналами и отправка сообщений клиента. Некоторые команды — дополнительные и поддерживаются не всеми IRC-серверами и клиентами. В табл. 11-1 перечислены все IRC-команды.

Таблица 11-1. *Команды IRC*

Название команды	Синтаксис
Admin	ADMIN [<*Сервер*>]
Away (дополнительная)	AWAY [<*Сообщение*>]
ChannelMode	MODE <*Канал*> {[+ | −] | o | p | s | i | t | n | b | v} [<*Предел*>] [<*Пользователь*>] [<*Маска_фильтра*>]
Connect	CONNECT <*Сервер-цель*> [<*Порт*> [<*Удаленный_сервер*>]]
Error	ERROR <*Сообщение*>
Info	INFO [<*Сервер*>]

Таблица 11-1. *Команды IRC* (продолжение)

Название команды	Синтаксис
Invite	INVITE <*Имя*> <*Канал*>
Ison (дополнительная)	ISON <*Имя*> {<*Имя*>}
Join	JOIN <*Канал*> {,<*Канал*>} [<*Ключ*> {,<*Ключ*>}]
Kick	KICK<*Канал*> {,<*Канал*>} <*Пользователь*> {,<*Пользователь*>} [<*Комментарий*>]
Kill	KILL <*Имя*> <*Комментарий*>
Links	LINKS [[<*Сервер*>] <*Маска*>]
List	LIST [<*Канал*> {,<*Канал*>} [<*Сервер*>]]
Names	NAMES [<*Канал* > {,< *Канал* >}]
NickName	NICK <*Имя*> [<*Число_переходов*>]
Notice	NOTICE <*Имя*> <*Текст*>
Operator	OPER <*Пользователь*> <*Пароль*>
OperWall (дополнительная)	WALLOPS <*Сообщение*>
Part	PART <*Канал*> {,<*Канал*>}
Password	PASS <*Пароль*>
Ping	PING <*Сервер1*> [<*Сервер2*>]
Pong	PONG <*Сервер1*> [<*Сервер2*>]
PrivateMessage	PRIVMSG <*Получатель*> {,<*Получатель*>} <*Сообщение*>
Quit	QUIT [<*Сообщение*>]
Rehash (дополнительная)	REHASH
Restart (дополнительная)	RESTART
ServerQuit	SQUIT <*Сервер*> <*Комментарий*>
Server	SERVER <*Сервер*> <*Число_переходов*> <*Информация*>
Stats	STATS [<*Запрос*> [<*Сервер*>]]
Summon (дополнительная)	SUMMON <*Пользователь*> [<*Сервер*>]
Time	TIME [<*Сервер*>]
Topic	TOPIC <*Канал*> [<*Тема*>]
Trace	TRACE [<*Сервер*>]
UserMode	MODE <*Имя*> {[+ | -] | i | w | s | o}
User	USER <*Пользователь*> <*Хост*> <*Сервер*> <*Настоящее_имя*>
UserHost (дополнительная)	USERHOST <*Имя*> { <*Имя*>}
Users (дополнительная)	USERS [<*Сервер*>]
Version	VERSION [<*Сервер*>]
Who	WHO [<*Имя*> [o]]
WhoIs	WHOIS [<*Сервер*>] <*Маска_имен*> [,<*Маска_имен*> [,...]]
WhoWas	WHOWAS <*Имя*> [<*Число*> [<*Сервер*>]]

MUD

MUD (multiuser dungeon) — это виртуальная среда, в которой пользователи взаимодействуют друг с другом, а также с персонажами и декорациями, созданными компьютером. MUD можно также назвать *многопользовательским измерением* (multiuser dimension) или *многопользовательским диалогом* (multiuser dialog), в зависимости от того, с кем Вы общаетесь, и от предназначения конкретного MUD.

MUD позволяет каждому пользователю, создав в виртуальном мире собственный персонаж, исследовать этот мир или же взаимодействовать с другими персонажами, управляемыми компьютером или людьми. Игры MUD весьма разнообразны, наиболее популярны виртуальные сражения.

Среда MUD основана преимущественно на текстовых сообщениях, с помощью которых пользователи общаются. Клиент MUD взаимодействует с сервером, вводя с клавиатуры короткие команды.

Каждый экземпляр MUD обладает собственным именем. Во-первых, с начала 80-х, когда появилась самая первая MUD, разработаны десятки программных вариантов этой игры — MUD, LPMUD, DikuMUD, TinyMUD, TinyMUCK, TinyMUSH, MOO и многие другие. Большинство из них — всего лишь слегка модифицированные версии первой MUD, однако существует несколько действительно новых программ (разработчики и пользователи часто бывают очень преданы «своей» версии). Во-вторых, каждый мир, воплощенный в MUD, имеет уникальное название, обычно оканчивающееся суффиксом «MUD» (например, FooMUD). Новичков даже иногда ошеломляет количество названий.

MUD приводится «в движение» MUD-сервером, который хранит MUD-мир и координирует взаимодействие группы пользователей. Участники соединяются с MUD-сервером посредством Telnet-клиента (с исключительно текстовым интерфейсом) или MUD-клиента (с более дружественным интерфейсом). Клиент обычно взаимодействует с сервером через 4201-й порт по протоколу TCP/IP, но сам протокол варьируется в зависимости от программной реализации MUD.

Согласно правилам «виртуального мира», персонажи некоторых пользователей могут стать «волшебниками» (wizard), таким образом, обретая в своем мире административные права. Другие игроки превращают своих персонажей в «киборгов» (cyborg), передавая управление MUD-клиенту. В том мире можно встретить и «ботов» (bot), которыми управляют компьютеры.

Некоторые MUD-серверы предоставляют дополнительные услуги RWHO-сервера (Remote WHO). Клиент обращается к такому серверу через Telnet, порт 6889 и запрашивает список пользователей MUD-сервера в формате WHO.

Web-чат

Помимо IRC и MUD, в Интернете существуют и другие способы текстового общения. С ростом популярности Web-протоколов многие люди организовали на своих Web-страницах «комнаты для бесед» — чаты. Иногда для этого применяются технологии, подобные IRC, использующие шлюз для соединения с сервером HTTP. В других случаях — совершенно новые программы. Реализации могут сильно отличаться друг от друга. Иногда это, например, элементарный сценарий, принимающий ввод текста от пользователей и записывающий его в общий файл диалога, который обычно и отображается Web-браузером.

Ссылки

<news:alt.irc>

<news:rec.games.mud.announce>

RFC 1459, «Internet Relay Chat Protocol»

Часть 4

World Wide Web

ГЛАВА 12

Основы World Wide Web

В середине 90-х годов очень популярной стала WWW (World Wide Web) — «Всемирная паутина». Это набор протоколов и программ для Интернета, представляющих информацию в гипертекстовом формате. Знаменитый браузер Mosaic, созданный в Национальном центре по применению супер-ЭВМ (National Center for Supercomputer Applications, NCSA), был первым графическим Web-браузером и способствовал популяризации WWW. Web разработана в 1989 году в Европейской лаборатории физики частиц (European Laboratory for Particle Physics, CERN) Тимоти Бернерсом-Ли (Timothy Berners-Lee). В настоящее время всеми стандартами, имеющими отношение к Web, ведает Консорциум World Wide Web (W3C).

Гипертекст применяется для создания документов, взаимосвязанных ссылками, — с ними легко управляться даже новичкам. В протоколах для Web в одних из первых в Интернете стали использовать гипертекст для упрощения и повышения эффективности работы с большими объемами непоследовательной информации. Теперь пользователю не обязательно заучивать команды базового протокола FTP, а для получения информации из Интернета не надо быть асом-программистом — достаточно просто щелкнуть ссылку на странице.

Для упаковки и передачи данных в Web применяются протоколы MIME (Multipurpose Internet Mail Extensions) и TCP/IP (Transmission Control Protocol/Internet Protocol), а также и другие, например FTP и Telnet. Специально для Web разработаны указатели URL (Uniform Resource Locator), протокол HTTP (Hypertext transfer Protocol), язык HTML (Hypertext Markup Language) и интерфейс CGI (Common Gateway Interface) — они описаны в этой главе.

Изучение материала главы предполагает знание TCP/IP и MIME.

Унифицированные указатели ресурсов (URL)

Указатель URL (Uniform Resource Locator) — это адрес сетевого ресурса. Он похож на имя файла, но дополнительно содержит имя сервера и информацию о сетевом протоколе, используемом данным ресурсом. В некоторых случаях URL включает сведения об имени пользователя, а также специальные аргументы и параметры протокола.

На Web-страницах URL используются для ссылок на другие страницы. В виде URL можно описать многие распространенные сетевые команды, указатели на файлы (доступные через FTP) и на сообщения из групп новостей Usenet (всемирной сети UNIX-систем, функционирующей как электронная доска объявлений для групп пользователей), запросы Finger и Gopher и т. д.

Все это было доступно и раньше, но появление URL значительно упростило программы, представляющие ценную информацию в гипертекстовой среде. Действительно до эры URL процесс передачи всей информации о сервере, файле, протоколе, пользователе и аргументах был достаточно неудобен, особенно для новичков в Интернете. С появлением URL значительно упростился весь механизм.

Указатель URL состоит из следующих частей:

<схема>:*<специальное_имя>*

где *<схема>* — это название схемы (используемый протокол, например http, ftp и т. д.), а *<специальное_имя>* — имя в формате, зависящем от используемой схемы.

Многие URL имеют следующий формат:

<протокол>://*<пользователь>*:*<пароль>*@*<хост>*:*<порт>*/*<путь>*

где *<пользователь>* — это имя пользователя, если оно необходимо (например, для FTP с не анонимной регистрацией); *<пароль>* — пароль этого имени пользователя; *<хост>* — доменное имя хоста, например «*fictionalcorp.com*», или его IP-адрес в числовом формате вида x.x.x.x; *<порт>* — номер IP-порта для соединения (если он не указан, используется стандартное значение для данного протокола); *<путь>* — связанные с URL данные, часто это указание подкаталога и имени файла.

Для Web-страницы URL выглядит, например, так:

`http://www.fictionalcorp.com/corpinfo/sales.html`.

Фрагмент http указывает, что URL использует протокол HTTP; *www.fictionalcorp.com* — имя сервера, к которому желает подключиться пользователь; */corpinfo/sales.html* — подкаталог и имя HTML-файла, хранящего Web-страницу.

В табл. 12-1 описаны некоторые известные URL-схемы.

Таблица 12-1. *Некоторые популярные URL-схемы*

Схема	Описание
http	Протокол HTTP
https	Протокол HTTP, зашифрованный с помощью SSL (Secure Socket Layer)
malito	Адрес электронной почты
ftp	Протокол FTP
finger	Протокол Finger
gopher	Протокол Gopher
wais	Глобальный информационный сервер WAIS
news	Новости Usenet
nntp	Новости Usenet по протоколу NNTP (Network News Transfer Protocol)
snews	Новости Usenet по протоколу NNTP, зашифрованному с помощью SSL
file	Имена файлов определенного хоста
jdbc	Объект базы данных JDC (Java Database Connector)
irc	Сеанс IRC (Internet Relay Chat)
telnet	Интерактивный сеанс Telnet
tn3270	Интерактивный сеанс терминала IBM 3270
afs	Имена файлов глобальной файловой системы AFS (Andrews File System)
nfs	Имена файлов файловой системы NFS (Network File System)
cid	Идентификатор содержимого (сообщения на основе Content-ID)
mid	Идентификатор сообщения (сообщения на основе Message-ID)
z39.50r	Запрос на выборку данных по протоколу Z39.50
z39.50s	Интерактивный сеанс по протоколу Z39.50.

Более короткий URL — *http://www.fictionalcorp.com* — указывает на «основную страницу» этого сервера. Если явно не задано имя файла, то HTTP-серверы используют значение по умолчанию (часто это default.html или index.html). Приведенный URL можно преобразовать к более явному виду: *http://www.fictionalcorp.com/default.html*.

Для протокола FTP применяется сходный синтаксис. Обращение к файлу *bar.txt* подкаталога */foo* FTP-сервера *ftp.fictionalcorp.com* на языке URL выглядит следующим образом: *ftp://ftp.fictionalcorp.com/foo/bar.txt*.

Из-за огромной популярности World Wide Web в Интернете многие браузеры предполагают наличие префикса «*http://*» в URL, не содержащем явного указания протокола.

Обычно имена доменов в явном виде содержат информацию о протоколе. Например, FTP-сервер компании называется *ftp.fictionalcorp.com*, а HTTP-сервер — *www.fictionalcorp.com*. Однако с появлением URL компания вправе использовать одно и то же имя *fictionalcorp.com* для обоих серверов. Тогда при использовании их ресурсов необходимо явно задавать соответствующий протокол, например *ftp://fictionalcorp.com/public/foo.txt* или *http://fictionalcorp.com/*.

Частичный или относительный URL — тот, в котором не указан протокол, хост, порт или путь, а лишь относительное имя ресурса. Например, если Web-страница *http://www.fictionalcorp.com/public/foo/bar.html* ссылается на *bletch.html*, это не что иное, как относительная форма от *http://www.fictionalcorp.com/public/foo/bletch.html*.

Как уже говорилось ранее, синтаксис URL меняется в зависимости от URL-схемы. Например, в протоколе HTTP символ «#» после имени HTML-файла обозначает точку привязки (закладку). Так, *http://www.fictionalcorp.com/foo.html#disclaimer* указывает на фрагмент *disclaimer* документа *foo.html*.

Вместо подкаталога и имени файла URL может содержать другую информацию о ресурсе. Так выгладит URL для протокола NNTP:

`nntp://<хост>:<порт>/<группа_новостей>/<статья>`

где <*группа_новостей*> — имя группы новостей, а <*статья*> — номер статьи.

Вообще говоря, в URL следует использовать только алфавитно-цифровые символы, так как большинство специальных символов зарезервированы или их прямое использование небезопасно. К первым относятся: «;», «/», «?», «:», «@», «=», «&». Ненадежные символы — «<», «>», «"», «#», «%», «{», «}», «|», «\», «^», «~», «[», «]», «`».

Если имя ресурса содержит зарезервированный символ или символ, не принадлежащий US-ASCII, то перед использованием в URL имя надо закодировать. При кодировании символ заменяется тремя новыми — знаком процента (%) и двумя шестнадцатеричными числами, представляющими код заменяемого символа.

Протокол HTTP

Протокол HTTP — основной и достаточно простой способ передачи данных между Web-сервером и клиентом. До появления Web и HTTP для передачи файлов в Интернете в качестве протокола ввода/вывода (input/output, I/O) чаще всего применяли FTP.

HTTP — это компактный, быстрый протокол ввода/вывода, работающий с URL и предназначенный для сред гипертекст/гипермедиа. В отличие от FTP, это протокол без состояний и имеет лишь несколько команд (методов). Благодаря использованию MIME, HTTP приспосабливается ко многим форматам данных и различным задачам ввода/вывода.

HTTP — клиент-серверный протокол, реализующий модель запрос/ответ. HTTP-клиент, или пользовательский агент (обычно это Web-браузер), подключается к HTTP-серверу с помощью URL и запрашивает ресурс, например HTML-документ.

Для инкапсуляции данных в этой модели применяются расширения MIME. Структура данных, пересылаемых между клиентом и сервером, напоминает электронную почту. Она состоит из тела сообщения и метаданных (заголовков сообщений). Протокол HTTP передает информацию в формате MIME. Мета-данные содержат информацию, необходимую для передачи данных между HTTP-сервером и клиентом. Однако HTTP допускает двоичный формат, чего обычный MIME (из-за 7-битных ограничений почтовых шлюзов) не позволяет.

В сеансе связи, как правило, участвуют HTTP-клиенты (Web-браузеры) и HTTP-серверы (Web-серверы) и реже — прокси-серверы. Последние выступают в качестве сервера по отношению к клиенту и в качестве клиента по отношению к другому серверу, передавая исходный запрос клиента через шлюз (например, межсетевой экран между интрасетью компании и Интернетом).

Традиционно HTTP-клиенты и серверы общаются через 80-й порт TCP/IP, по умолчанию зарезервированный для HTTP. Однако могут использоваться и другие порты, явно указанные в URL. В дополнение замечу, что HTTP не предполагает применение именно TCP/IP и отлично функционирует с другими протоколами гарантированной доставки.

Web-браузер часто просматривает Web-страницы, состоящие из многих объектов, например самого HTML-документа и нескольких изображений (GIF, JPEG, PNG и др.). Большинство HTTP-клиентов для чтения начального HTML-документа создают только один поток (с одним подключением к серверу), а затем еще несколько потоков (каждый с от-

дельным подключением к серверу) для получения остальных необходимых файлов. Соединение устанавливается по запросу клиента и разрывается после ответа сервера.

Сообщения

Сообщения протокола HTTP обычно представляют собой запросы HTTP-клиентов и ответы HTTP-серверов. В сообщениях-запросах есть «строка запроса» (Request-Line), содержащая сам запрос, а в сообщениях-ответах — «строка состояния» (Status-Line), содержащая собственно ответ, а также тело сообщения (или сущность), содержащее исходные данные.

Команды

По сравнению с другими протоколами ввода/вывода, HTTP содержит мало команд (или методов). Обязательной реализации подлежат только три метода — GET, HEAD и POST. Другие четыре — PUT, DELETE, LINK и UNLINK — тоже описаны, но они не так популярны.

GET

Команда GET осуществляет поиск ресурса на сервере и его передачу клиенту. Синтаксис команды следующий:

```
GET <URL> HTTP/1.0
```

Вот, например, команда GET для обращения к файлу *foo.html* сервера *www.fictionalcorp.com*:

```
GET www.fictionalcorp.com/foo.html HTTP/1.0
```

Для выборки только тех ресурсов, которые изменились после определенного времени, используется команда «условный GET», реализуемая при помощи поля **If-Modified-Since** заголовка.

HEAD

Команда HEAD очень похожа на GET, но возвращает лишь мета-информацию об указанном URL и не возвращает сам файл, то есть ответ не содержит тела. Команду HEAD следует применять, если клиента интересует только возможность доступа к URL или его изменения.

POST

Команды GET и HEAD передают информацию от сервера, POST же используется для передачи данных от клиента к серверу. Большинство Web-документов предназначены только для чтения, и пользователи Web-браузеров обычно не пересылают на сервер новые файлы. Однако они

часто заполняют различные HTML-формы (например, различные заявки). Информация из HTML-форм передается HTTP-клиентом к HTTP-серверу при помощи команды POST.

PUT

Команда PUT менее распространена по сравнению с POST (и не так широко поддерживается). Она пересылает данные от HTTP-клиента к HTTP-серверу.

DELETE

Эта команда используется для удаления определенного URL на сервере. Команда DELETE не слишком популярна из-за анонимности HTTP-клиентов, а также потому, что большинство Web-документов доступно только для чтения.

LINK

Команда LINK используется для связи определенного URL с другими ресурсами. Она не распространена повсеместно.

UNLINK

Командой UNLINK отключается определенный URL от других ресурсов. Она не распространена повсеместно.

Коды состояния

Протокол HTTP определяет набор кодов состояния, которые должны быть понятны и клиенту и серверу, чтобы те могли успешно передавать сообщения. Коды разбиты на категории, перечисленные в табл. 12-2.

Таблица 12-2. *Категории кодов состояния HTTP*

Категория кода состояния	Номера кодов состояния	Описание
Информационные	100 – 199	Сообщения конкретных приложений
Успешные	200 – 299	Запрос успешно обработан
Перенаправление	300 – 399	Для обработки запроса требуются дополнительные действия клиента. Обычно они выполняются без участия пользователя
Ошибка клиента	400 – 499	Проблемы на стороне клиента
Ошибка сервера	500 – 599	Проблемы на стороне сервера

Каждый код состояния HTTP представляет собой число, после которого следует текстовая строка, содержащая дополнительную мета-информацию. В табл. 12-3 приведены коды состояния и их описания. Помимо

кодов состояния, включенных в спецификации HTTP, приложения способны определять свои коды состояния.

Таблица 12-3. *Описание кодов состояния HTTP*

Код состояния	Описание
200 OK	Нет ошибки, запрос успешно обработан
201 Created	Выполнена команда POST
202 Accepted	Получен асинхронный запрос: он уже получен, но не обязательно обработан
204 No Content	Запрос успешно обработан, но клиенту нечего отобразить. Это иногда полезно в качестве метаинформации для ответов, которые не нужно показывать пользователю
300 Multiple Choices	Требуемый ресурс доступен из многих мест. В ответе возвращается список альтернатив. Предпочтительный выбор сервера включен в поле **Location** ответа
301 Moved Permanently	Требуемый URL перемещен на новый URL (указанный в поле **Location** ответа). Все последующие ссылки на этот ресурс должны использовать новый URL
302 Moved Temporarily	Требуемый URL временно перемещен на новый URL (указанный в поле **Location** ответа). Последующие ссылки на этот ресурс должны использовать старый URL
304 Not Modified	Выполнена команда «условный GET», однако, документ не изменялся со времени, указанного в поле **If-Modified-Since**
400 Bad Request	Запрос не распознан. Клиенту следует послать исправленный запрос
401 Unauthorized	Если запрос был анонимным, то его следует аутентифицировать. Если запрос был аутентифицирован, то это — отказ в доступе
403 Forbidden	Сервер отказывается обработать запрос. Обычная причина — нарушение прав доступа
404 Not found	Сервер не нашел указанный URL
500 Internal Server Error	Произошла непредвиденная ошибка сервера
501 Not Implemented	Сервер не поддерживает этот запрос
502 Bad Gateway	Прокси-сервер (или шлюз) получил неправильный ответ от сервера, к которому он подсоединен
503 Service Unavailable	Сервер временно недоступен или отказывается обработать запрос. Обычная причина — перегрузка сервера или его обслуживание

Поля заголовка

Заголовки HTTP-сообщений имеют различные поля, используемые в запросах и ответах. Одни из них специально предназначены для запро-

сов клиента, другие — для ответов сервера. Некоторые из этих полей не поддерживаются определенными клиентами или серверами. Подробнее поля описаны ниже.

Accept

В поле **Accept** перечислены возможные форматы данных при ответе на запрос. Большинство клиентов показывают допустимость всех форматов, водя в это поле звездочку (*). Нераспознанные форматы данных передаются пользователю с просьбой сопоставить их с некоторым MIME-типом.

Accept-Charset

Здесь перечислены наборы символов (помимо US-ASCII и ISO-8859-1), поддерживаемые клиентом.

Accept-Encoding

Это поле сходно с полем **Accept**, но описывает допустимые кодировки содержимого ответа (см. **Content-Encoding**).

Accept-Language

Это поле сходно с полем **Accept**, но указывает языки, допустимые в ответе.

Allow

В поле **Allow** перечислены поддерживаемые сервером команды (GET, HEAD и т. д.). Клиенту разрешено их использовать.

Authorization

Его применяют HTTP-серверы, не разрешающие анонимный доступ к некоторым ресурсам. В этом поле запроса передается «удостоверение личности» пользователя.

Для случаев, когда анонимный доступ невозможен, HTTP предоставляет простой механизм аутентификации типа «запрос-ответ». Допустимы различные механизмы аутентификации. В базовом для HTTP механизме аутентификации имя пользователя и пароль шифруются нестойким MIME-кодированием Base64. Некоторые HTTP-клиенты и серверы поддерживают другие, более безопасные методы, например Microsoft Windows NTLM.

В заголовок ответа, содержащего код «401 Unauthorized», сервер обязательно включает поле **WWW-Authenticate**, где описана причина отказа и поддерживаемые сервером методы аутентификации. После этого

пользователь повторно передает запрос, заголовок которого содержит дополнительное поле **Authorization** с «удостоверением личности» для указанного механизма аутентификации. Если сервер не принимает это «удостоверение личности», то он отвечает кодом «403 Forbidden».

Content-Encoding

Здесь указан механизм кодирования (zip, compress и т. д.), который следует использовать для декодирования данных.

Content-Language

Здесь указан естественный язык предполагаемой аудитории.

Content-Length

Здесь определен размер тела переданного сообщения. В случае команды HEAD в этом поле указан размер данных, который имел бы место при использовании команды GET.

Content-Type

Здесь определен тип тела переданного сообщения. Для HTML-документов в этом поле обыкновенно находится «text/html». В случае команды HEAD в этом поле указан тип данных, который бы имел место при использовании команды GET.

Date

Здесь указаны дата и время создания сообщения.

Expires

Здесь указаны дата и время, когда данные теряют актуальность. После наступления указанного момента и до обновления данных HTTP-клиенты не должны эти данные кэшировать. Если в этом поле записан 0 (хотя это и неестественный вид), то данные теряют актуальность немедленно.

From

Здесь указан адрес электронной почты запрашивающего пользователя. Это поле применяется не в целях аутентификации, а для регистрации пользователей. На случай возникновения ошибок автоматизированные HTTP-клиенты, например роботы, должны содержать адрес электронной почты лица, ответственного за запуск робота.

If-Modified-Since

Это поле даты и времени используется командой GET для доступа к ресурсу только в том случае, если он был изменен. Это поле полезно

для клиентов, применяющих кэширование. Если изменений нет, то возвращается код состояния «304 Not Modified».

Last-Modified

Здесь указана дата последнего изменения данных.

Link

Это поле позволяет устанавливать взаимосвязь данных с другим ресурсом, иерархической структурой или путями просмотра.

Location

В этом поле содержится точный URL прежнего местонахождения ресурса для случаев автоматического перенаправления (коды состояния 300 – 399).

MIME-Version

Содержит номер версии используемого протокола MIME.

Pragma

Это многоцелевое поле для директив конкретной реализации. Одной из распространенных директив является «no-cache», показывающая, что данные не следует кэшировать.

Refer

Позволяет клиенту определять URL, от которого получен запрашиваемый URL. Это помогает выявлять обратные ссылки, позволяющие отслеживать ошибки и определять доходы от рекламы. Обратные ссылки иногда содержат конфиденциальную информацию, поэтому пользователи должны иметь возможность отключать это поле. К сожалению, большинство распространенных в настоящее время HTTP-клиентов не позволяют этого сделать.

Retry-After

Указывает интервал времени, в течение которого службы недоступны. Используется совместно с кодом состояния «503 Service Unavailable».

Server

Поле указывает название и версию HTTP-сервера.

Title

Поле указывает описательное имя объекта.

URI

Здесь перечислены некоторые или все унифицированные идентификаторы ресурсов URI (Uniform Resource Identifier), доступные для данного ресурса.

User-Agent

Поле указывает название и версию HTTP-клиента.

WWW-Authenticate

Реализует не анонимный доступ с аутентификацией типа «запрос/ответ». В этой схеме «удостоверение личности» не шифруется. Подробности — ранее в разделе «Authorization».

Протокол HTTP 1.1

В настоящее время используется версия 1.1 протокола HTTP. Ее поддерживают все основные клиенты (браузеры) и Web-серверы. Протокол HTTP 1.1 описан в RFC 2068 и превосходит предыдущую версию — HTTP 1.0, — прежде всего, по производительности. (Подробности о повышении производительности см. по адресу *http://www.w3.org/Protocols/HTTP/Perfomance*.) Однако есть и другие отличия, некоторые из них описаны ниже.

- Постоянные соединения. Протокол HTTP 1.1 устанавливает меньше TCP-соединений, чем HTTP 1.0. Версия 1.0 устанавливает и разрывает TCP-соединение для каждого HTML-запроса, а HTTP 1.1 создает TCP-соединение, сохраняющееся на протяжении многих запросов. Это также позволяет передавать несколько запросов в одном TCP-сегменте. Постоянные соединения обеспечивают более высокую производительность, чем расширение Netscape под названием HTTP «Keep Alive», так как они лучше работают с прокси-серверами.
- Протокол HTTP 1.1 поддерживает сжатие данных. Это означает, что файлы между клиентом и сервером могут передаваться сжатыми, что снижает нагрузку на сеть.
- Создание виртуальных хостов. Протокол HTTP 1.1 позволяет одному Web-серверу (с одним IP-адресом) иметь несколько доменных имен. В настоящее время эта ситуация распространена достаточно часто, например, когда поставщик услуг Интернета содержит несколько доменов.
- Протокол HTTP 1.1 поддерживает многие языки.

- Протокол HTTP 1.1 поддерживает выборочную передачу, что подразумевает пересылку только выделенного участка файла или документа. Это особенно полезно при утере TCP-соединения, так как не приходится отправлять весь документ заново — передача возобновляется с последней контрольной точки.

Язык HTML

HTML (Hypertext Markup Language) — это простой язык разметки гипертекста (т. е. формат данных), используемый в Web. Формат HTML позволяет представить целый ряд гипертекстовых документов. Обычно HTML-файлы — статические документы. Используя шлюзы (см. ниже раздел «Интерфейс CGI»), в формате HTML можно отобразить динамическую информацию, например выборку из баз данных. Подробнее о Dynamic HTML (DHTML) и использовании языков типа Microsoft Visual Basic Scripting Edition (VBScript), не требующих интерфейса CGI, — в главе 13.

HTML — упрощенная версия обобщенного языка SGML (Standard Generalized Markup Language), формально определяющего структуру документов. Язык HTML прост, но достаточно мощен для представления большинства документов общего назначения. В полях, описывающих тип содержимого, формат HTML обозначается типом *text/html*.

Базовую основу Web составляют HTML-документы с гиперссылками (в виде URL), передаваемые по протоколу HTTP.

Ранние версии HTML описывались неформально. Первым формально описанным и широко поддерживаемым стал язык HTML версии 2.0. Следующая версия — HTML 3.2 — также получила широкую поддержку. Совсем недавно вышла версия HTML 4.0. Подробнее возможности HTML 4.0 описаны в главе 13. Вот наиболее значительные различия между версиями 3.0 и 4.0:

- стандартизованные таблицы стилей;
- поддержка двунаправленной письменности;
- усовершенствованные фреймы;
- усовершенствованные таблицы;
- поддержка математических символов;
- поддержка дополнительных возможностей для инвалидов (озвучивание и чтение вслепую).

Консорциум W3C создал службу HTML-сертификации. Ее можно использовать для проверки соответствия HTML-объекта стандартам HTML 4.0. Подробности — по адресу *http://validator.w3.org*.

Специальные символы

Язык HTML содержит некоторые зарезервированные символы. Для ввода любого из них в тексте или в URL используйте строку-название или числовую строку, приведенную в табл. 12-4.

Таблица 12-4. *Специальные символы в HTML-тексте*

Символ	Строка-название	Числовая строка	Описание
HT	Нет		Табуляция
LF	Нет
Перевод строки
CR	Нет
Возврат каретки
SP	Нет		Пробел
NBSP			Неразрывный пробел
"	"	"	Кавычка
&	&	&	Амперсанд
<	<	<	Меньше
>	>	>	Больше
®	®	®	Символ зарегистрированного товарного знака
©	©	©	Символ авторского права
<любой>		&#<код>;	Любой символ из ISO 8859-1.

Теги

Для описания и разметки документов в языке HTML применяют *теги* (tags) — они позволяют создавать заголовки, абзацы, списки, гиперссылки и форматировать символы. Теги не чувствительны к регистру символов.

Большинство тегов HTML используются парами, причем каждая пара состоит из открывающего и замыкающего тегов. Открывающий выделяется символами «<» и «>». Закрывающий — символами «</» и «>». Например, для того чтобы представить текст «Foo» жирным шрифтом, используется пара тегов <B> и </B>:

```
<B>Foo</B>
```

Некоторые теги бывают только открывающими. Так, для принудительного перевода строки сразу за текстом «Bar» можно использовать тег
:

```
Bar<BR>
```

Некоторые теги не могут быть вложенными.

В HTML-документах допустимы комментарии. Они заключаются между «<!--» и «-->».

Структура документа

Все HTML-документы имеют формальную структуру, она проиллюстрирована в следующем примере.

```
<!DOCTYPE HTML PUBLIC "--//W3C//DTD HTML 3.2 Final//EN">
<HTML>
<HEAD>
<TITLE>
<!-- Заголовок документа -->
</TITLE>
<!-- Остальные заголовки -->
</HEAD>
<BODY>
<!-- Тело документа -->
</BODY>
</HTML>
```

Документы HTML версии 3.2 используют заголовок <!DOCTYPE>. Весь документ ограничен парой тегов <HTML> и </HTML>. Мета-данные документа, в том числе пара тегов <TITLE> и </TITLE>, заключаются между тегами <HEAD> и </HEAD>, данные документа — между <BODY> и </BODY>.

Заголовки

В части документа, ограниченной тегами <HEAD>, содержатся различные мета-теги, в том числе <TITLE> и </TITLE>, описывающие документ.

BASE

Этот тег — полный URL, который используется в качестве базового адреса при разрешении относительных URL.

ISINDEX

Этот тег представляет набор гиперссылок, в которых браузер проводит поиск по заданному пользователем ключевому слову.

LINK

Этот тег связывает страницу с другим ресурсом. Ссылаться можно на таблицу стилей, основную страницу, информацию об авторских правах, на другие версии документа и т. д.

META

Этот тег предоставляет мощный способ описания мета-данных (и даже HTTP-заголовков), используемых HTTP-клиентами и серверами. Каждый тег <META> состоит из пары имя/значение.

```
<META HTTP-EQUIV="<имя>" CONTENT="<значение>">
<META NAME="<имя>" CONTENT="<значение>">
```

Здесь *<имя>* — это имя заголовка, а *<значение>* — это значение заголовка. Если вместо NAME указано HTTP-EQUIV, то этот тег <META> будет использован в качестве HTTP-заголовка.

SCRIPT

Применяется для включения в документ сценария, написанного на языке VBScript или JavaScript. Подробности — в главе 13.

STYLE

Этот тег включает в документ таблицу стилей. Подробности — в главе 13.

TITLE

Это описание документа. Часто Web-браузер показывает его в заголовке своего окна.

Ссылки

Пара тегов <A> и </A> задает «точку привязки» гиперссылки. Обычно это ссылка на другой документ в виде либо абсолютного, либо относительного URL. Например:

```
<A HREF="http://www.fictional.com/default.html"></A>
```

Иногда дается ссылка на определенный фрагмент другого документа. Тогда в URL входит суффикс «*#<название_фрагмента>*»:

```
<A HREF="http://www.fictional.com/default.html#copyright"></A>
```

В пределах одной страницы название фрагмента можно задавать следующим образом:

```
<A NAME="foo"></A>
```

После этого ссылку на фрагмент внутри одного документа определяют так:

```
<A HREF="#foo"></A>
```

Извне данного документа ссылка выглядит следующим образом:

```
<A HREF="http://www.fictional.com/default.html#foo"></A>
```

Изображения

Для вставки изображения в документ используется тег <IMG>. Допустимы следующие форматы изображений: GIF (Graphics Interchange Format), JPEG (Joint Photographic Experts Group), PNG (Portable Network Graphics) и т. д. Формат тега <IMG> следующий:

```
<IMG SRC="<URL>" ALT="<имя>:">
```

где *<URL>* — URL на файл с изображением, а *<имя>* — это текстовое описание изображения. Атрибут ALT задается для пользовательских агентов, которые не могут показать изображение, например для текстовых браузеров.

Графические карты

Элементарное свойство графической гипертекстовой среды — возможность щелкать мышью изображения. Пользователю гораздо проще вместо просмотра, например, списка всех стран, просто открыть изображение карты мира и щелкнуть мышью интересующие регионы.

Для воплощения этого в HTML необходима возможность, во-первых, использовать изображение как гипертекст, во-вторых, вычислять координаты выбранной пользователем части изображения для применения определенной гиперссылки.

Это реализуется в HTML при помощи графических карт. Для этого требуется само изображение, которое затем разбивают на области. Карту мира, например, удобно делить, определяя многоугольники, соответствующие границам разных стран.

Используя различные системы координат, карту можно составить из нескольких геометрических фигур. Для задания прямоугольника необходимы координаты левой верхней и правой нижней точки, для круга — координаты центра и значение радиуса, для многоугольника — пары значений x1/y1, x2/y2,..., xN/yN (по количеству вершин).

Данные координат затем используются HTML для связи элемента <MAP> с соответствующей гипертекстовой ссылкой. Например, многоугольник, соответствующий Индии, можно связать с URL Индии.

Существует два типа графических карт: клиентские и серверные. Сначала появились вторые. При их использовании все вычисления и переход по гиперссылке выполняются сервером. Клиентские графические карты появились позже в качестве расширения HTML (они не описаны в основной спецификации). Для них не нужно участие сервера — все

операции выполняются на стороне клиента. Применение клиентских графических карт в современных браузерах обосновано двумя причинами. Во-первых, они позволяют автономным клиентам просматривать локальные файлы (когда сервера вообще нет); во-вторых, они сокращают сетевой трафик.

Общие принципы форматирования

Язык HTML определяет шесть уровней заголовков. Тег <H1> задает заголовок первого уровня, а тег <H6> — шестого. Для разделения абзацев используется тег <P>, для указания конца строки — тег
, для разделения частей текста горизонтальной полосой — тег <HR>.

Для отображения текста курсивом служит тег <I>. Текст, на который Вы хотите обратить внимание, то есть акцентированный (обычно отображаемый курсивом), описывается тегом <EM>. Жирное начертание представляет тег <B>, а сильный акцент (обычно тоже отображаемый жирным) — тег <STRONG>.

Для укрупнения размера шрифта используется тег <BIG>, для уменьшения шрифта — <SMALL>. Нижние индексы обозначает тег <SUB>, верхние — <SUP>.

При упоминании заголовка книги или цитировании фрагмента, применяется тег <CITE>. При вставке фрагмента исходного кода, который чаще всего печатают моноширинным шрифтом, задействуют тег <CODE>. Для вставки текста, введенного пользователем, указывают элемент <KBD>. Для вставки последовательности алфавитных литералов — элемент <SAMP>. Для вставки форматированного текста с переносами строк, что удобно при использовании моноширинного шрифта, используется элемент <PRE>.

Для обозначения примера применяют элемент <XMP>. В этом режиме на строке помещается 132 символа. Листинг обозначают элементом <LISTING> — на строке помещается 80 символов.

При вставке адресной информации (имя автора, контактная информация, подпись и т. д.) используется элемент <ADDRESS>. Для указания цитаты из другого источника — элемент <BLOCKQUOTE>.

BODY

После тега <HEAD> документа, но перед содержимым вставляют элемент <BODY>. Он инициализирует различные характеристики документа: определяет стиль документа, включая фоновое изображение; тип шрифта, его размер и цвет; границы документа. В настоящее время

многие возможности <BODY> утратили свою актуальность. Это связано с появлением каскадных таблиц стилей.

Списки

В HTML можно реализовывать различные типы списков. Ненумерованные списки используют тег <UL>, нумерованные списки — тег <OL>. Для списков определений применяется тег <DL>, для списков каталогов — тег <DIR>. Для перечисления пунктов меню задействован тег <MENU>. В пределах одного списка каждый элемент заключен в пару тегов <LI> и </LI>.

Таблицы

Таблицы, состоящие из столбцов и колонок с данными, создают с помощью тега <TABLE>. Для создания строки заголовка таблицы используют тег <TH>, а для основных строк — <TR>. Каждая ячейка строится средствами тега <TD>.

Формы

Формы позволяют пользователю осуществлять ввод данных. Обычно формы передает на сервер HTTP-команда POST. Форма представляет собой шаблон для ввода данных. Для работы большинства форм необходимо выполнение CGI-приложения на сервере. (Подробности ниже, в разделе «Интерфейс CGI».)

Определение формы выглядит так:

```
<FORM ACTION="<URL>" METHOD=GET|POST>
<!-- Данные формы -->
</FORM>
```

где ACTION описывает URL, который должен выполняться при запуске формы, а METHOD — метод HTTP (либо GET, либо POST).

Формы HTML поддерживают различные поля ввода данных. Поле Text предназначено для обычного текста. Пароли и другой невидимый текст вводят в поле Password. Для этих полей можно указать ограничение по длине вводимой информации, а также значение по умолчанию. Еще есть кнопки-флажки и радио-кнопки; им тоже можно задать значение по умолчанию (отмечена или не отмечена). Списки из нескольких элементов задают в поле Select. Здесь можно указать значение по умолчанию, а также установить флаг — способ выбора (одиночный или множественный). Поле Hidden предназначено для той необходимой серверу информации, которую не нужно показывать пользователю. Для выполнения форм существуют поля Submit и Reset.

Фреймы

Изначально HTML-документ отображался как одна страница в одной области экрана. Фреймы позволяют отображать несколько HTML-окон либо как части главного окна, либо как самостоятельные окна.

Обычно фреймы используются для разделения Web-страницы на области. Одну строку или столбец, отделенный от остальной страницы, можно прокручивать независимо. Часто в маленьком фрейме собирают ссылки для навигации по узлу, а главный фрейм используют для отображения собственно информации.

Фреймы задаются путем описания их структуры. Это специальный HTML-документ, где вместо тега <BODY> применяется <FRAMESET>. Он задает различные фреймы в общей структуре. Для каждого фрейма задается URL, а также формат строк, столбцов и границ. Дополнительно этот документ может описывать представление исходного документа в браузерах, не поддерживающих фреймы. Для этого используется элемент <NOFRAME>.

Для описания отдельного окна из набора фреймов применяется элемент <FRAME>. Он задает имя окна и такие параметры, как возможность прокрутки, изменения размеров и изображения границы.

Из-за ограничений и линейного характера действия кнопок **Back** (Назад) и **Next** (Вперед) модель навигации Web-браузера не очень хорошо приспособлена к использованию многооконных фреймов. «Коряво» спроектированный Web-узел, использующий фреймы, скорее, затруднит, а не облегчит работу пользователя. Хотя фреймы и являются мощным дополнением к HTML, применять их надо осмотрительно.

Подробнее о тегах <EMBED>, <APPLET>, <APP>, <OBJECT>; Java-апплетах; подключаемых модулях (Plug-In); элементах управления ActiveX и о многом другом написано в главе 13.

Интерфейс CGI

Платформо-независимый интерфейс CGI (Common Gateway Interface) используется для исполнения программ совместно с HTTP-сервером. Он обеспечивает «шлюзовую» совместимость между не-HTTP-серверами и другими формами данных. Пожалуй, наиболее распространенный вариант — шлюз баз данных, используемый для Web-представления информации из баз данных.

HTTP-сервер выполняет CGI-процесс, который зачастую представляет собой самостоятельное приложение. HTTP-клиент задает URL, указывающий на CGI-приложение, и тем самым запускает его. HTTP-сервер выполняет это CGI-приложение и возвращает HTTP-клиенту результаты (вывод).

Интерфейс между HTTP-серверами и CGI-приложениями состоит из командной строки, параметров (переменных окружения), ввода и вывода.

Часто используются сценарии, например написанные на языке Perl. Поэтому CGI-приложения часто называют *CGI-сценарии* (CGI scripts).

Аргументы

Вот пример URL для CGI-приложения без параметров:

```
http://www.samplecorp.com/cgi-bin/foo.cgi
```

Параметры для CGI-приложения передаются в той же строке URL, но отделяются знаком вопроса (?), например:

```
http://www.samplecorp.com/cgi-bin/foo.cgi?a=1
```

Доступ к командной строке осуществляется через аргументы *argc* и *argv* процедуры *main*().

Переменные окружения

Данные от HTTP-сервера CGI-приложению передаются при помощи переменных окружения. Программы на языке C обращаются к этим переменным через функцию *getenv*(). В табл. 12-5 описаны переменные окружения CGI.

Таблица 12-5. *Переменные окружения CGI*

Переменная	Описание
AUTH_TYPE	Описывает механизм аутентификации HTTP
CONTENT_LENGTH	То же, что поле **Content-Length** HTTP-заголовка
CONTENT_TYPE	То же, что поле **Content-Type** HTTP-заголовка
GATEWAY_INTERFACE	Версия CGI, поддерживаемая сервером; обычно это CGI/1.1
HTTP_*	Способ обращения к HTTP-заголовкам — надо указать префикс «HTTP_» и название заголовка
PATH_INFO	Путь к CGI-приложению, выделенный из URL
PATH_TRANSLATED	Путь к CGI-приложению, записанный в соответствии с правилами именования файлов операционной системы сервера

(см. след. стр.)

Таблица 12-5. *Переменные окружения CGI* (продолжение)

Переменная	Описание
QUERY_STRING	Закодированный для URL поисковый запрос; запросная часть URL
REMOTE_ADDR	IP-адрес запрашивающего агента, обычно — HTTP-клиента
REMOTE_HOST	Полное доменное имя запрашивающего агента (если доступно), обычно это имя HTTP-клиента
REMOTE_IDENT	Имя (если доступно) запрашивающего агента
REMOTE_USER	Имя пользователя запрашивающего агента (если используется аутентификация)
REQUEST_METHOD	Запрошенная HTTP-команда (метод)
SCRIPT_NAME	Имя CGI-приложения
SERVER_NAME	Имя сервера — полное доменное имя или IP-адрес
SERVER_PORT	Номер порта, через который принят запрос
SERVER_PROTOCOL	Название и версия протокола, принявшего запрос (обычно это HTTP/x.x)
SERVER-SOFTWARE	Название и версия серверного программного обеспечения

Ввод и вывод

Вводимые клиентом данные (их размер указан в CONTENT_LENGTH) считываются из стандартного потока ввода (stdin). Выходные данные для клиента записываются в стандартный поток вывода (stdout). Иногда выходные данные вкладываются в тело MIME-сообщения. В остальных случаях, если значение REQUEST_METHOD определено как GET либо HEAD, выходные данные представляются в виде отдельного файла, на который указывает поле **Location** HTTP-заголовка. Интерфейс CGI использует коды состояния HTTP.

Безопасность

Ввиду того что интерфейс CGI позволяет клиентам выполнять на сервере программы, следует предпринять меры предосторожности при написании и хранении CGI-приложений. Администраторы сервера должны оградить HTTP-клиентов от опасного программного обеспечения. Следует тщательно проверять CGI-программы на предмет изъянов в защите.

Будущие протоколы

Консорциум W3C работает над протоколом HTTP-NG (Next Generation), который, как предполагается, заменит HTTP. К HTTP-NG предъявляются следующие требования:

- простота — протокол HTTP-NG должен быть прост для реализации и обслуживания;
- расширяемость — на случай ситуации, не предусмотренной в процессе разработки;
- масштабируемость — вне зависимости от того, используется ли HTTP-NG в маленькой локальной интрасети или в Интернете;
- эффективность — ожидается, что протокол HTTP-NG будет намного эффективнее HTTP. Последний плохо работает в сетях с большим временем задержки. Причина в том, что HTTP — протокол одиночных запросов и ответов, кроме того, он перегружен информацией. Протокол HTTP-NG призван устранить эти и другие недостатки.

Подробную информацию о протоколе HTTP-NG можно получить на Web-узле консорциума W3C — *http://w3.org./Protocols/HTTP-NG/Activity.html*.

Ссылки

http://www.w3.org

http://www.w3.org/MarkUp

http://www.w3.org/Protocols

http://hoohoo.ncsa.uiuc.edu/cgi/overview.html

RFC 2070, «Internationalization of the Hypertext Markup Language»

RFC 2068, «Hypertext Transfer Protocol—HTTP 1.1»

RFC 1945, «Hypertext Transfer Protocol—HTTP 1.0»

RFC 1942, «HTML Tables»

RFC 1867, «Form Based File Upload in HTML»

ГЛАВА 13

Дополнительные сведения о Web

Глава 12 посвящена базовой архитектуре Web. Вы узнали, что для обращения к ресурсам используются указатели URL, для «упаковки» ресурсов — расширения MIME, для передачи — протокол HTTP, для отображения — язык HTML, а для расширения базовых возможностей HTTP-сервера — интерфейс CGI. Эти технологии — основа Web, тем не менее новые протоколы тоже создаются. Здесь описаны многие новшества касающиеся базовой архитектуры, в том числе расширение Web-клиента и Web-сервера, сценарии и Java-приложения.

Расширение Web-клиента

Изначально Web-браузеры лишь отображали текст и позволяли путешествовать по гиперссылкам. Однако со временем их возможности расширились. Сейчас Web-браузер часто рассматривают в качестве универсального клиента, так как он способен выполнять функции, для которых раньше требовалось несколько специализированных клиентов.

Вспомогательные MIME-приложения

Существенно возросли возможности Web-браузеров, предназначенные для чтения вложенной в HTML-документ информации. Для сообщения Web-серверу о понятных ему типах данных Web-браузер использует поле **Accept** HTTP-заголовка. Большинство браузеров понимают по крайней мере HTML-документы и несколько форматов изображений. Изначально не ставилась задача, чтобы браузеры распознавали такие типы содержимого, как звук, видео и другие специализированные файлы конкретных приложений (например, документы Microsoft Word, электронные таблицы Microsoft Excel, CAD-файлы [computer-aided design]). Однако большинство браузеров способно принимать от Web-серверов любые форматы данных независимо от того, понятны они или нет. Это позволяет использовать Web не только для обычных форматов HTML и

изображений, но и для специфических и «пока не определенных» форматов данных.

Если браузер встречает неизвестный для него формат данных, то он ищет в базе данных MIME (ее создает и обслуживает браузер и/или операционная система клиента) имя «вспомогательного MIME-приложения» (MIME helper application), которое отвечает за работу с данным форматом. Например, в системе Microsoft Windows работать с фильмами в формате MPEG (Moving Picture Experts Group) позволяет Media Player, а с документами — Word. Если для какого-то типа данных нет соответствующего «вспомогательного MIME-приложения», то Web-браузеры обычно запрашивают пользователя, что делать с данным типом. Тому же приходится либо связать с данным типом определенное приложение, либо скомандовать, чтобы браузер прервал передачу данных.

Специальные маркеры HTTP

Поскольку изначально HTTP создавался как протокол без состояний (stateless), содержимое Web-узлов было доступно только для чтения. Но вскоре владельцам Web-узлов понадобилось, чтобы сервер умел запоминать и в дальнейшем использовать информацию о сеансе Web-клиента. Например, создатели торговых Web-узлов поняли, что если приписывать каждому клиенту-покупателю уникальный идентификатор транзакции (transaction ID), то удастся сопоставить различные транзакции с конкретным покупателем. Рекламодателям на Web-узлах для упрощения построения демографических баз данных и персонализации узлов при помощи динамического содержимого потребовалась информация о посещаемых пользователями страницах.

Все эти чаяния воплотились в разработке компании Netscape — *специальных маркерах HTTP* (HTTP Cookies). Это — небольшое количество данных, сохраняемых и впоследствии восстанавливаемых из системы клиента. Специальные маркеры быстро стали популярными. В конечном счете, прежнюю спецификацию заменила RFC 2109, а вопросы защиты личной информации пользователя были успешно разрешены.

Специальные маркеры обычно используются CGI-программами, работающими на серверах. Управляют маркерами Web-браузеры — *пользовательские агенты HTTP* (HTTP User Agent). Как правило, браузеры сохраняют такую информацию в базе данных в одном или нескольких файлах. Более сложные браузеры позволяют пользователю отказываться от предложенных серверами специальных маркеров и удалять ранее сохраненные маркеры.

Специальные маркеры реализуются двумя новыми HTTP-заголовками: Cookie и Set-Cookie. Передача специального маркера происходит по инициативе сервера, предлагающего клиенту сохранить этот маркер. Сервер передает данные в заголовке Set-Cookie. Если клиент способен и желает сохранить информацию, то он делает это. Впоследствии, когда пользователь повторно посещает узел или любой URL, связанный с данным специальным маркером, браузер возвращает данные в заголовке Cookie.

Такие заголовки характеризуются множеством пар «атрибут-значение», например:

- <*имя*>=<*значение*>: <*имя*> — имя специального маркера, <*значение*> — содержащиеся в специальном маркере данные;
- domain=<*домен*> — имя сервера, с которым следует связывать специальный маркер;
- path=<*путь*> — относительный путь к серверу, с которым следует связывать специальный маркер;
- secure — если присутствует эта характеристика, то специальный маркер можно использовать только при защищенном подключении; в настоящее время под этим обычно подразумевают HTTPS-транзакцию (применяется HTTP, зашифрованный с помощью SSL);
- comment=<*комментарий*> — описывает назначение специального маркера;
- Max-Age=<*секунды*> — время жизни специального маркера;
- version=<*версия*> — версия спецификации специального маркера. Для RFC 2109 значение равно 1. В исходной спецификации данная характеристика не использовалась.

Извлечение данных клиентом и рассылка данных сервером

Извлечение (pull) данных клиентом и *рассылка* (push) их сервером — это два взаимосвязанных метода, благодаря которым HTML-документы становятся более динамичными. Для их работы требуется дополнительная поддержка расширений MIME со стороны Web-клиента. Эти методы предназначены для использования приложениями стороны сервера, понимающими новый формат MIME.

Web-клиенты обычно получают данные от Web-сервера за одну HTTP-транзакцию. Методы извлечения и рассылки данных увеличивают дли-

тельность логической транзакции следующим образом: информация передается несколько раз, при этом каждый новый набор сведений заменяет предыдущий.

При извлечении данных клиентом сервер передает ему сами данные и управляющую информацию — как правило, способ обновления данных или новый URL, по которому будет обращаться клиент. Временем обновления управляет новый заголовок HTTP-ответа (он содержит характеристику Refresh = <*секунды*>).

При рассылке данных сервером клиент постоянно поддерживает открытым подключение к серверу. Получив данные, клиент отображает их и ожидает поступления следующих — их он использует для обновления предыдущих. Так продолжается до тех пор, пока сервер не решит закрыть соединение. При реализации этих механизмов применяется MIME-тип multipart/mixed.

На сегодняшний день эти два метода не очень широко распространены. Чаще применяются другие клиентские и серверные технологии, кроме того, не все браузеры поддерживают механизмы извлечения данных клиентом и рассылки их сервером.

В настоящее время появился целый ряд программных продуктов, предназначенных для рассылки. Учитывая то, что основа Web — извлечение данных (другими словами, клиент сам осуществляет поиск и выборку информации), иногда довольно трудно выявить источник рассылки. Реально же так называемые *технологии рассылки* (push technologies) представляют собой технологии извлечения, но с некоторыми, весьма изощренными, дополнениями. Последние могут включать фильтрование, выполняемое либо клиентом, либо сервером. Клиент подключается к Web-серверу, который, по его мнению, содержит необходимую информацию, или к какой-то программе-посреднику, чтобы сообщить ей о том, что его интересует, при помощи механизма предпочтений или другим способом. Впоследствии программа-посредник уведомляет клиента о наступлении интересующего события (в соответствии с указанными предпочтениями). В порядке возрастания сложности различные программные продукты подразделяются на три категории:

- *уведомление* (notification);
- *плановое извлечение* (scheduled pull);
- *информационные каналы* (channels).

Они не являются взаимоисключающими, то есть конкретный программный продукт может относиться к нескольким категориям.

Уведомление

Клиент регистрируется на Web-сервере и ожидает уведомлений об интересующих его событиях. Из всех событий отбираются те, которые происходят почти всегда. Например, обновилась главная страница Wall Street Journal, или цена на авиабилеты отвечает определенному критерию, или стоимость ценных бумаг достигла определенного уровня. Уведомлением может быть сообщение электронной почты — простое или с вложением, содержащим информацию с интересующей Web-страницы, или, например, вызов на пейджер с просьбой к клиенту обратиться к серверу и считать страницу.

Плановое извлечение

Плановое извлечение — это ряд технологий, которые подразделяют на «тупые» и интеллектуальные.

При «тупом» извлечении клиент периодически посещает интересующий его Web-сервер и «скачивает» Web-страницы для автономного просмотра. Программные продукты, использующие данный тип извлечения данных, разнятся, во-первых, по глубине поиска на Web-узлах; во-вторых, по тому, производят ли они поиск на Web-узлах, требующих аутентификацию (например, идентификатор пользователя и пароль); в третьих, используют ли кэш совместно с браузером или нет.

При интеллектуальном извлечении клиент либо загружает только измененные Web-страницы, либо начинает загрузку Web-страницы лишь после получения уведомления.

Информационные каналы

Информационные каналы удобны для рассылки больших объемов данных. Если в качестве транспортного механизма используется HTTP, то клиент периодически проводит опрос сервера на предмет изменений в документах или интересующих его объектах. Информация группируется по темам в потоки (или каналы), которые имеют собственный формат, зависящий от производителя. Похожий механизм реализован в спутниковом телевидении: различные компании устанавливают собственные спутниковые антенны.

Компания Microsoft применяет формат CDF (Channel Definition Format — см. далее). Программный продукт компании Netscape называется

Netcaster. Многие компании создают свои информационные каналы, освещающие всевозможные темы. Например, на канале компании McAfee — SecureCast — Вы найдете все о вирусах. Канал компании CyberMedia — Oil Change — просветит Вас об обновленных версиях программного обеспечения.

Одной из основных проблем существующих технологий рассылки является их слабая масштабируемость. Web-сервер и его инфраструктура не справляются с большим количеством клиентских запросов. Существуют временные решения, позволяющие снизить нагрузку на сеть, например, клиенту предлагается опрашивать либо программу посредник, либо локальный сервер. Программы-посредники копируют информацию, интересующую многих клиентов, на ближайший к ним сервер. При этом рассылка данных проводится лишь один раз, что позволяет снять нагрузку с части сети. Ранее же данные рассылались каждому клиенту по отдельности. В будущем для рассылки информации предполагается использовать групповое вещание протокола IP, которое обеспечивает лучшую масштабируемость.

Дополнительные модули от Netscape

Изначально для расширения Web-браузеров использовались только вспомогательные MIME-приложения, которые функционировали как самостоятельные процессы в отдельных окнах. Дополнительные модули (plug-ins) от Netscape можно подключать к браузеру. И хотя это отдельные программы, создается впечатление, будто они интегрированы в браузер.

Интерфейс дополнительных модулей создан для незаметной интеграции дополнительных приложений в браузер. Их применяют для обработки тех типов данных, которые сам браузер не поддерживает, а также для отделения программных модулей, созданных для конкретных операционных систем, что облегчает перенос существующих приложений, работающих с Web, на другие платформы.

Когда браузер встречает непонятный для него MIME-тип, он проверяет: не зарегистрирован ли какой-либо дополнительный модуль, поддерживающий данный тип. Если да, то браузер загружает его или создает новый экземпляр уже загруженного. При первичной загрузке браузер вызывает функцию NPP_Initialize дополнительного модуля, а если же создавался новый экземпляр — функцию NPP_New. Когда Вы закроете окно, в котором действует дополнительный модуль, для удаления экземпляра браузер вызовет функцию NPP_Destroy, а после удаления всех экземпляров — функцию NPP_Shutdown.

Программные интерфейсы (API) дополнительных модулей позволяют обмениваться информацией с браузером и совместно использовать системные ресурсы, предоставляя следующие возможности: обработку событий, управление окнами, памятью, буфером данных и печать.

Технология Shockwave

Технология Shockwave («ударная взрывная волна»), разработанная компанией Macromedia, позволяет привнести мультимедийные возможности в Интернет вообще и на HTML-страницы в частности. Эта технология включает целый ряд программных продуктов. Shockwave-объект создают с помощью одного из нескольких инструментов компании Macromedia, например хорошо известного Director. Затем на Web-страницу помещают ссылку на Shockwave-объект. Web-браузеру, просматривающему Web-страницу со ссылкой на Shockwave-объект, необходима специальная программа просмотра. Для Microsoft Internet Explorer программа просмотра Shockwave-объектов является объектом ActiveX, для Netscape Navigator — дополнительным модулем. Технология Shockwave обеспечивает работу с аудио, видео, анимацией, а также ввод данных пользователем при помощи мыши. Технология Shockwave также включает язык сценариев и механизм компрессии/декомпрессии объектов.

Компания Macromedia продает технологию создания Shockwave-объектов, а технологию просмотра объектов распространяет бесплатно. Для создания Shockwave-объектов предлагается два коммерческих программных продукта — Director и Authorware, также от компании Macromedia. Программы просмотра существуют для целого ряда платформ, в том числе для всех версий Windows и Macintosh. Поставляются они совместно с Web-браузерами и компьютерами.

Компания Macromedia предлагает использовать свои продукты в Интернете и интрасетях, а также для разработки мультимедийного обучающего программного обеспечения. Дополнительная информация — по адресу *http://www.macromedia.com*.

Java-апплеты

Java-апплеты — это небольшие, платформо-независимые программы, которые выполняются в контексте обрабатываемого браузером HTML-документа. С помощью таких программ к статическим документам добавляют динамическую информацию, например аудио, видео, доступ к электронным таблицам и базам данных и т. д. Как и HTML-документы, Java-апплеты хранятся на Web-серверах и загружаются Web-клиентом в локальную систему по протоколу HTTP.

Язык программирования Java — это компактный, объектно-ориентированный язык, очень похожий на C++. Вместо компиляции в исполняемый код, исходный текст на Java (из файла с расширением .java) компилируется в байт-код (в файл с расширением .class) и выполняется виртуальной Java-машиной (Java Virtual Machine, JVM).

Язык Java содержит набор API, переносимых на целый ряд операционных систем и платформ. Эти программные интерфейсы реализуют базовые функции (управление окнами и памятью, ввод данных пользователем и т. д.) и группируются в пакеты.

Java-апплет — это особый вид Java-приложения. Он не может выполняться как самостоятельная программа, его необходимо встраивать в другое приложение. Обычно апплет встраивается в HTML-документ и выполняется Web-браузером, поддерживающим Java. В целях безопасности возможности апплетов ограничены: они почти не имеют доступа к клиентской системе и способны общаться только с сервером, с которого были загружены. Все Java-апплеты используют специальный пакет java.applet, который обеспечивает интерфейс между апплетом и его окружением (Web-браузером и HTML-документами).

Java-апплеты встраиваются в HTML-документ средствами тега <APPLET>. Этот тег описывает начальный размер окна апплета, параметры, место, откуда браузер сможет загрузить соответствующий class-файл, и многое другое.

GIF-анимация

Несмотря на то, что существует множество различных технологий, делающих Web-клиент более интеллектуальным, к сожалению, большинство из них не нашли столь широкого применения, как графические изображения в формате GIF. Изначально этот формат использовался Web-браузерами для отображения одиночных изображений. Однако менее распространенная версия спецификации GIF позволяет сохранять несколько изображений в одном файле. В этом формате доступна информация о скорости смены кадров, что напоминает форматы видеофайлов, но без звука.

Когда популярные Web-браузеры стали поддерживать GIF-анимацию, это решение получило широкое применение. К примеру, популярный Java-апплет «Tumbling Duke» содержит несколько GIF-файлов и собственно апплет для загрузки изображений и их «оживления». Однако его можно заменить всего одним файлом с GIF-анимацией. Художники Web-узлов творчески используют GIF-анимацию для придания динами-

ки статичному Web-узлу. Рекламодатели средствами анимации сообщают пользователям дополнительную информацию, а некоторые Web-дизайнеры применяют ее, чтобы включить несколько рекламных плакатов в одну панель.

Технология PICS

Технология PICS (Platform for Internet Content Selection) — это методика классификации содержания, позволяющая программному обеспечению управлять доступом детей к объектам сомнительного содержания. Этот метод можно применять и для других видов фильтрования и идентификации материалов.

Служба оценки предоставляет различные способы классификации содержания. В HTML-документах отметка связывается с содержимым при помощи тега <META>, создающего HTTP-заголовок PICS-Label. Служба оценки использует систему оценок (шкалу отметок). Например, кинофильмы в США классифицируются по шкале G, PG, PG-13 и т. д. Для телевидения существует похожая система оценок.

Когда содержимое связано со службой оценки и конкретной системой оценок, для управления доступом к нему можно применить фильтрующее программное обеспечение — ряд программ, позволяющих родителям контролировать доступ детей к объектам сомнительного содержания. Многие современные браузеры содержат встроенную поддержку PICS.

Проект P3P

Проект P3P (Platform for Privacy Preferences Project) появился по инициативе консорциума W3C и определяет стандартный способ, которым Web-узел демонстрирует свой режим обеспечения секретности, а пользователь — реализует свои предпочтения в рамках этого режима. Например, при запросе пользователем какой-либо информации, узел может потребовать взамен некие данные. Пользователь вправе передать лишь часть их либо отправить необходимую информацию только после подтверждения. В проекте P3P не описаны новые технологии, а использованы уже существующие. Например, язык XML (Extensible Markup Language), специальные маркеры (cookies) и цифровые подписи. Подробности — по адресу *http://www.w3.org/p3p/overview.html*.

Механизм CSS

Механизм CSS (Cascading Style Sheets — Каскадные таблицы стилей) — спецификация языка HTML, его используют для сопоставления стилей

(таких, как шрифт, цвет, разрядка, поля, расположение и т. д.) с элементами HTML-документа.

Таблица стилей (style sheet) — это набор правил форматирования, применяемых к нескольким документам. В результате она действует как шаблон в компьютерной издательской системе. Как и шаблон, таблица стилей позволяет автоматически отформатировать определенный элемент во всех документах, с которыми она связана. Например, Вы вправе использовать для всех заголовков верхнего уровня шрифт Arial, размером в 20 пунктов, жирного начертания. Суть «каскадности» этого механизма в том, что допустимо использование нескольких таблиц стилей со своими правилами для каждого конкретного документа.

Таблица стилей связывается с HTML-документом несколькими способами. Допустимо хранить ее в отдельном файле, на который ссылаются несколько документов. Для этого в заголовочных частях этих HTML-документов используется тег <LINK>, например:

```
<LINK REL="таблица_стилей" TYPE="text/css" HREF="foo.cs">
```

Информацию из таблицы стилей можно вставить непосредственно в HTML-документ. Для этого в заголовке документа задают тег <STYLE>. Затем этот стиль применяется ко всему документу. Кроме того, задавать стиль можно в пределах одного элемента HTML-документа, для чего у этого элемента используют атрибут STYLE.

Механизм CSS существенно расширяет возможности управления HTML, тем самым расширяя диапазон его применения для публикаций в Web.

Язык XML

Язык XML (Extensible Markup Language) — это технология, развивающаяся под патронажем консорциума W3C. Язык XML дополняет HTML и основан на языке SGML (Standard Generalized Markup Language). В то время как HTML описывает набор команд, определяющих расположение данных на странице, язык XML позволяет описать данные HTML-страницы с помощью типов представляемой информации. Прямым следствием этого стало то, что поисковые машины возвращают более осмысленную информацию. Например, используя XML, поисковая машина в состоянии определить, когда слово «cookie» на HTML-странице относится к Интернету, а когда к сладкому десерту.

С помощью XML браузеры способны однократно загружать HTML-страницу, а затем работать с ней автономно, не обращаясь к серверу. Клиент вправе любым образом просматривать данные и управлять ими. В

идеале XML позволит клиенту извлечь из результата поиска, например, название и адрес гостиницы, а затем загрузить эти сведения в картографическую программу для составления маршрута.

Синтаксис XML

Для кодирования символов в XML используется кодировка Unicode. Это означает, что XML совместим со множеством языков мира. Для передачи данных по каналам связи совместно с Unicode применяют несколько видов перекодирования. По умолчанию XML задействует UTF-8 (Universal Transformation Format-8). (Подробности о Unicode и UTF-8 — в главе 4.)

В отличие от HTML, в языке XML имеют значение регистр символов и отступы пробелами.

Некоторые символы в XML зарезервированы, например «<» и «>» — обрамление тегов. Эти символы в текстовых строках следует кодировать как «&lt» и «&gt» соответственно. Таким образом, строка «25 > 24 И 25 < 26» кодируется в XML как «25 &gt 24 И 25 &lt 26».

Синтаксис языка XML очень похож на синтаксис HTML и состоит из последовательностей тегов, текста и комментариев. Основное отличие в том, что теги XML, скорее, характеризуют то, что представляют данные, чем то, как их следует представлять. Еще одно отличие XML — потенциально неограниченный набор тегов: пользователь может определять собственные теги, в то время как в HTML они фиксированы.

Синтаксис языка XML представлен серией элементов, каждый из которых состоит из открывающего тега, содержания и закрывающего тега, например:

```
<PERSON>
<LASTNAME>Gates</LASTNAME>
<FIRSTNAME>Bill</FIRSTNAME>
</PERSON>
```

Здесь элемент PERSON (человек) начинается в первой строке и заканчивается в последней. Как видно из примера, один элемент может содержать вложенные другие элементы. Простые документы могут содержать сами себя и сами себя описывать. Сложные документы описываются во внешнем DTD-файле *описания типа документа* (Document Type Definition).

DTD-файл

DTD-файл определяет действующий в XML-документе синтаксис. В частности, в нем перечислены элементы, которые могут встретиться в

конкретном документе, и определяются взаимосвязи между различными элементами. С конкретным XML-документом DTD-файл может связываться, а может и не связываться.

Связь XML и SGML

Как уже говорилось ранее, язык XML основан на SGML (Standard Generalized Markup Language), но гораздо проще, что позволяет с успехом использовать XML для передачи информации по сети и в то же время обеспечивает легкость применения и функциональную совместимость различных платформ и приложений. Для XML-документов типа well-formed (вполне сформированные) не требуется связанный с ними DTD-файл и, благодаря этому, их легче передавать по сети. В SGML нет эквивалента этого типа XML-документа. Для XML-документов типа valid (правильные) требуется связанный с ними DTD-файл, и для них существует эквивалент в SGML.

XML-словари

Можно сказать, что DTD-файл представляет собой *словарь* — совокупность описанных элементов и правил составления из них новых элементов. В следующих разделах рассмотрены некоторые XML-словари. Ожидается, что в перспективе набор словарей будет пополняться.

CDF

CDF (Channel Definition Format) — это словарь, определяющий структурное представление данных, предназначенное для рассылки Web-узлами. Он включает описания различных каналов, расписание рассылки и т. д. Впервые CDF появился в Microsoft Internet Explorer версии 4.

OSD

OSD (Open Software Description) — это стандартный словарь, разработанный совместно компаниями Microsoft и Marimba. Он облегчает организацию рекламных кампаний, загрузку и установку программ через Интернет. Спецификация OSD представлена на рассмотрение в W3C во второй половине 1997 года.

RDF

Словарь RDF (Resource Description Framework) развивается под патронажем W3C, он предназначен для создания средств описания мета-данных (информации о данных), облегчающих поиск ресурсов и информации в Web. Словарь RDF основан на технологии MCF (Meta Content Format) от Apple, впоследствии приобретенной компанией Netscape.

OFX

OFX (Open Financial Exchange) — стандартный словарь для представления обмена данными между финансовым приложением (например, Intuit Quicken или Microsoft Money) и финансовым учреждением. В настоящее время словарь OFX описан в формате SGML, но вскоре ожидается версия в формате XML.

Пространства имен XML

Пространства имен XML (XML Namespaces) используются тогда, когда толкование XML-тегов неоднозначно. Например, если для музыкального магазина тег <GROUP> означает записи какой-либо группы на компакт-дисках (CD), то для большой корпорации он же может означать отдел компании, в котором работает конкретный служащий. Благодаря пространствам имен, в одном документе допустимо присутствие элементов нескольких словарей, а также исключены конфликты при применении одноименных элементов разных словарей.

XML и безопасность

В настоящее время XML безопасен так же, как HTML. Защиту можно усилить, используя шифрование данных сервером перед передачей и последующую расшифровку их клиентом. Другой способ — взаимодействие по защищенному каналу, например, при помощи S-HTTP. Эти технологии применяются в настоящее время. В перспективе предполагается использовать технологии, напоминающие MOSS (MIME Object Security Service). Дело в том, что XML содержит много элементов, из которых формируются объекты. Для передачи содержимого конфиденциальных объектов разрешены различные методы защиты.

Язык XSL

Языки Dynamic HTML (см. раздел «Dynamic HTML, скриптлеты и модель DOM» далее в этой главе) и XML похожи друг на друга в том смысле, что они оба позволяют пользователю динамически просматривать данные различными способами и при этом не обращаться к серверу. Механизм CSS, составляющий неотъемлемую часть Dynamic HTML, «умеет» работать с простыми XML-данными, но не способен использовать все возможности XML (например, добавлять новый элемент данных). Язык XSL (Extensible Style Language) основан на родственном к SGML языке DSSSL (Document Style Semantics and Specification Language) и является CSS-совместимым языком описания таблиц стилей. Он разрабатывается под патронажем W3C в качестве инструмента для форматирования элементов XML. Одна из функций XSL, например, — извле-

чение элемента данных из XML-документа, его форматирование и многократное отображение в другом документе.

Сценарии

Одно из наиболее мощных нововведений — вставка в HTML-документ *сценариев* (script), написанных на *языке сценариев* (scripting language). Сценарий дополняет статичный HTML-документ возможностью взаимодействовать с пользователем.

JavaScript

Язык JavaScript — это относительно простой объектно-ориентированный язык сценариев, позволяющий создавать объекты и управлять ими. Его создала компания Netscape, изначально он назывался Live Script. Хотя внешне JavaScript и похож на Java, но, тем не менее, это разные языки.

Исходный код на JavaScript вставляется в HTML-документ с использованием тега <SCRIPT LANGUAGE="JavaScript">. При отображении документа браузер передает этот код интерпретатору JavaScript.

Основные виды операторов — присваивания, сравнения, арифметические, побитные, логические, строковые и специальные. Язык поддерживает следующие стандартные конструкции: *break, comment, continue, delete, do...while, export, for, for...in, function, if...else, import, return, switch, var, while* и *with*.

JavaScript предоставляет доступ к свойствам и методам объектов. Это применимо к базовым объектам: Array, Boolean, Date, Function, Math, Number, RegExp и String. Для безопасности работы с Web-страницами, имеющими сценарии, браузеры используют дополнительную объектную модель. Для работы с HTML-документами применяются объекты Document и Form. Для работы с окнами браузеров, фреймами, историей посещенных адресов, MIME-типами и связанными с ними функциями — объекты Window и Browser.

Для придания интерактивности в JavaScript существуют объекты типа «событие» и обработчики событий. Существуют события с именами вида onXXX, где XXX может принимать следующие значение: Abort, Click, DblClick, DragDrop, Error, Focus, KeyDown, KeyPress, KeyUp, Load, MouseDown, MouseMove, MouseOut, MouseOver, MouseUp, Move, Reset, Resize, Select, Submit и Unload.

ECMAScript

ECMAScript — это язык JavaScript, одобренный европейским комитетом стандартов ECMA. Компании, образовавшие комитет, — Netscape, Microsoft, Sun Microsystems, IBM, Borland International и др. — работают над созданием стандартного языка сценариев для Web.

Стандарт ECMAScript описывает язык и базовый набор объектов. В то же время он не определяет интерфейсы всех объектов различных хостов. Например, браузер, поддерживающий совместимый с ECMAScript язык сценариев, может также содержать дополнительный набор интерфейсов объектов, не доступных в других браузерах.

ActiveX и JScript

Принимая во внимание популярность JavaScript в Web, а также Microsoft Visual Basic в качестве средства разработки ПО для Windows, компания Microsoft разработала *абстрактный уровень машины сценариев* (script engine abstraction layer), благодаря которому обеспечивается поддержка различных существующих машин сценариев.

Именно благодаря этому Internet Explorer поддерживает несколько языков сценариев. Реализация языка JavaScript/ECMAScript в Internet Explorer называется JScript. Допустимо использовать и другие языки, например Perl, Tcl или REXX.

VBScript

По сути, язык VBScript (Microsoft Visual Basic, Scripting Edition) представляет собой подмножество Microsoft Visual Basic. Он реализован как быстрый, компактный, облегченный интерпретатор, используемый в Web-браузерах. VBScript позволяет оперировать управляющими элементами ActiveX и Java-апплетами.

По аналогии с языком JavaScript исходный код на VBScript вставляется в HTML-документ при помощи тега <SCRIPT LANGUAGE="VBScript">. Браузер передает исходный код из HTML-документа интерпретатору VBScript.

Будучи альтернативой для JavaScript, VBScript позволяет широкому кругу разработчиков ПО для Windows использовать свои навыки программирования на Visual Basic для создания интерактивного содержимого Web-страниц.

Расширение Web-сервера

Web-серверы, подобно Web-клиентам, появились в результате расширения базовых HTTP-серверов посредством CGI. Предлагались разные альтернативы CGI в зависимости от поставленных задач. В настоящее время сценарии как для клиентской стороны, так и для серверной являются дополнительным средством создания HTML. Изначально динамические HTML-документы создавались только при помощи CGI-приложений (их часто писали на языке Perl с использованием модулей для Интернета). Позднее языки сценариев клиентской стороны стали применяться и серверами.

Интерфейс NSAPI

Одним из первых интерфейсов для усовершенствования CGI был NSAPI (Netscape Server API). Интерфейс CGI предполагает наличие отдельного процесса для каждого приложения сервера, поэтому их запуск и остановка требует больших затрат. Кроме того, каждый конкретный CGI-процесс в некотором роде изолирован при доступе к ресурсам, из-за чего ему приходится самостоятельно инициализировать каждый требуемый ресурс (например, базу данных). Для часто используемых CGI-приложений эти проблемы стоят еще более остро.

Сервер, в котором реализован интерфейс NSAPI, разбивает последовательность HTTP-запросов и ответов на серии логических шагов. Запрос Web-клиента Web-серверу обрабатывается поэтапно: проверка прав доступа, преобразование имени, проверка пути, определение типа объекта, ответ на запрос и регистрация транзакции. Интерфейс NSAPI позволяет заменить каждый этап функциональным модулем от сторонних разработчиков. Такое разбиение на слои может применяться как глобально, так и для отдельного каталога или файла. NSAPI-приложения работают быстрее CGI-приложений. Они глубоко интегрированы с Web-сервером, так как имеют доступ к некоторым его функциям и данным, а также способны заменять уровни сервера новыми. Можно добавить новые функции, например специальную обработку ошибок, персональную настройку учета и особые схемы аутентификации.

Программы, тесно взаимодействующие с Web-сервером и требующие высокой производительности, до сих пор используют интерфейс NSAPI. Однако, многие производители, принимая во внимание новые технологии, применяют некоторые виды Dynamic HTML для разработки серверных приложений более высокого уровня.

Интерфейс WinCGI

Для осуществления ввода/вывода CGI-приложения должны иметь доступ к указателям stdin/stdout (стандартные потоки ввода/вывода). Приложения для платформы Win16 (большей частью на Visual Basic) испытывают трудности при получении этой информации. В связи с этим был разработан интерфейс WinCGI, благодаря которому HTTP-приложения можно писать на Visual Basic. Для этих целей WinCGI хранит входные и выходные данные в специальном файле (с расширением .ini).

Некоторое время назад многие Web-узлы на базе Windows использовали серверные приложения с интерфейсом WinCGI. Большинство производителей таких Web-узлов поддерживали WinCGI. Однако в настоящее время Win16 вытеснена более современной платформой Win32, и программисты на Visual Basic применяют новые методы (использующие Dynamic HTML) для создания серверных приложений. По этой причине WinCGI почти полностью заменили другие типы расширений Web-сервера.

Интерфейс ISAPI

Еще одна альтернатива CGI-приложениям — интерфейс ISAPI (Internet Server API), разработанный совместно компаниями Process Software и Microsoft. Он отличается большей производительностью и имеет ряд других преимуществ. Существует два типа интерфейсов ISAPI: ISAPI-приложения и ISAPI-фильтры. Их используют немногие Web-серверы, в том числе Microsoft Internet Information Server (IIS).

На радость пользователям, знакомым с операционной системой Novell NetWare, ISAPI похож на подгружаемые модули (NetWare Loadable Module, NLM) тем, что приложения выполняются как часть системного программного обеспечения (в данном случае Web-сервера). При этом ошибочный NLM-модуль способен вызвать сбой всей системы, а ошибочная DLL-библиотека ISAPI может привести к сбою в работе Web-сервера. На этом аналогия заканчивается, так как NLM-модули выполняются в режиме ядра, а ISAPI — нет. Средства разработки и отладки для ISAPI также более развиты, чем для NLM.

ISAPI-приложения

ISAPI-приложения похожи на CGI-приложения, которые функционируют как отдельные процессы. Однако создание процесса — достаточно трудоемко, особенно для часто вызываемых программ. В системе Windows ISAPI-приложение реализуется в виде динамической библиотеки (DLL) для Win32, вызываемой в контексте Web-сервера. Таким образом,

процесс создается только один раз. Аналогичное повышение производительности характерно и для других операционных систем с похожей моделью процессов.

Модель, на которой основаны все ISAPI-приложения, аналогична модели CGI-приложения. Процедура HttpExtensionProc эквивалентна процедуре main, а процедуры ReadClient и WrightClient осуществляют, соответственно, ввод из потока stdin и вывод в поток stdout. В структуре данных под названием ECB (Extension Control Block) содержатся основные HTTP-заголовки и переменные. Доступ к остальным HTTP-заголовкам реализуется через процедуру GetServerVariable.

Интерфейс ISAPI используется высокопроизводительными, низкоуровневыми серверными приложениями и шлюзами. Для компактных и менее производительных программ продолжают применять CGI. В настоящее время все более популярным становится программирование на более высоком уровне при помощи сценариев, использующих Dynamic HTML для взаимодействия с существующими объектами.

ISAPI-фильтры

ISAPI-фильтр — это процесс (в системе Windows реализованный как DLL-библиотека для Win32), который перехватывает HTTP-потоки ввода/вывода между HTTP-клиентом и сервером. ISAPI-фильтры способны просматривать и изменять эти потоки в тот момент, когда они переданы клиентом, но еще не достигли сервера, и наоборот. При обнаружении запроса от клиента к серверу фильтр может поступить одним из трех способов: не менять запрос; обработать его и передать серверу; обработать запрос и завершить его существование (в этом случае запрос не достигнет сервера).

ISAPI-фильтры предназначены для специализированных серверных приложений, которым необходимо следить за активностью HTTP. К ним относят приложения, реализующие, например, специализированные схемы аутентификации, статистический анализ браузера, особое сопоставление URL и файлов, шифрование и сжатие.

Ниже приведены некоторые варианты ISAPI-фильтров:

- фильтры, заменяющие основной механизм аутентификации клиента специальным;
- фильтры, обеспечивающие передачу содержимого HTML в сжатом виде;

- фильтры, шифрующие передаваемые HTML-данные и обеспечивающие конфиденциальность содержимого;
- фильтры, регистрирующие и анализирующие трафик.

Web-шлюзы баз данных

Web-шлюзы баз данных (Web database gateways) — одно из наиболее простых средств, используемых HTTP-серверами для предоставления динамического доступа через Web к информации из баз данных. На основе этого механизма создано множество решений, так что теперь разработчикам не надо «изобретать колесо» и создавать собственные CGI-приложения для реализации функций Web-шлюза.

Шлюз IDC

Один из Web-шлюзов — IDC (Internet Database Connector), реализованный в виде ISAPI-приложения и входящий в состав IIS и других Web-серверов. Этот простой механизм позволяет избежать создания нового CGI-приложения для выполнения функций шлюза между базой данных и Web-сервером. Шлюз IDC — самостоятельная программа, применяющая ODBC (Open Database Connectivity) для работы с любой ODBC-совместимой базой данных. Вместо написания нового приложения, разработчик Web-узла для хранения информации базы данных использует текстовый файл (с расширением .idc). Формат этого файла напоминает HTML, но содержит дополнительные переменные, хранящие имя базы данных, удостоверения личности пользователя, ODBC-запрос к серверу базы данных и HTML-шаблон для форматирования результирующих данных, передаваемых пользователю.

Технология ASP

Технология ASP (Active Server Pages) компании Microsoft — это серверный эквивалент используемых клиентами языков сценариев и объектов. ASP-страница — это HTML-документ, содержащий сценарий, который позволяет работать с управляющими элементами ActiveX. Сценарий выполняется на сервере, а клиент получает результаты. Сценарий ASP-страницы обычно управляет объектом, результаты работы которого представлены в формате Dynamic HTML и уникальны для каждого клиента.

Технологию ASP используют в основном Web-серверы на базе Windows — IIS и PWS (Personal Web Server), — но иногда и другие Web-серверы на базе иных операционных систем.

Совместно с ASP применяются различные низкоуровневые управляющие элементы ActiveX, в том числе для доступа к базам данных совместимым с ODBC, упрощающие создание динамического содержимого. Наличие низкоуровневых объектов, например управляющих элементов ActiveX, позволяет разрабатывать только сценарии высокого уровня, использующие эти объекты; а применение Web-сервером интерфейса машины сценариев ActiveX позволяет использовать различные языки сценариев. Таким образом, допустимы сценарии на JavaScript, VBScript, Perl, Tcl и других языках.

Одна из наиболее общих задач, возникающих при создании персонализированного HTML-документа, — определение возможностей браузера. Вместе с ASP поставляется файл *browscap.ini*, представляющий собой базу данных возможностей браузеров. Таким образом, для проверки наличия какой-либо функции браузера ASP-сценарию проще просмотреть этот файл, чем обращаться к полю **User Agent** HTTP-заголовка.

Подводя итог, я сравню ASP с другими серверными интерфейсами. Интерфейс CGI подходит для низкопроизводительных, компактных приложений. ISAPI-фильтры — для программ, которым необходим перехват ввода/вывода по протоколу HTTP. ISAPI-приложения — для низкоуровневых, высокопроизводительных приложений, не использующих управляющие элементы ActiveX. Наконец, технология ASP лучше всего годится для построения высокоуровневых сценариев, манипулирующих существующими управляющими элементами ActiveX.

Технология ASP поставляется с целым рядом встроенных «строительных модулей» на базе ActiveX. Ниже перечислены некоторые из них:

- *объект-запрос* (request object) — обеспечивает доступ к информации, указанной клиентом в HTTP-запросе;
- *объект-ответ* (response object) — генерирует ответы, например устанавливает специальный маркер (cookie);
- *объект-сеанс* (session object) — применяется для хранения всех переменных узла для каждого конкретного пользователя. Эти переменные сохраняются все время, пока пользователь просматривает страницы в пределах данного узла;
- *объект-приложение* (application object) — позволяет совместно работать с информацией всем пользователям одного узла;
- *объект-сервер* (server object) — позволяет создавать новые экземпляры объектов ActiveX.

Ниже описаны преимущества технологии ASP:

- позволяет создавать содержимое в формате Dynamic HTML с использованием любого языка программирования или языка сценариев — VBScript, JScript, REXX, Perl, Tcl, Visual Basic, C++ и даже COBOL;
- облегчает создание страниц Dynamic HTML только средствами сценариев;
- позволяет различать способ доступа к данным для создания Web-страницы и способ отображения содержимого страницы. Это упрощает разработку, отладку и сопровождение Web-страниц;
- позволяет вносить изменения без перекомпиляции. Она (компиляция) выполняется автоматически при обращении браузера (клиента) к файлу;
- работает со всеми типами браузеров. ASP содержит встроенные компоненты, определяющие возможности браузера, и функционирует в соответствии с ними. От браузера требуется лишь отображать динамическое содержимое, созданное в рамках ASP-окружения сервера.

Подробности — по адресу *http://www.microsoft.com/iis/default.asp*.

Прокси-серверы

Прокси-серверы обеспечивают эффективные и безопасные средства централизованного доступа в Интернет. Разные организации их средствами обеспечивают взаимодействие локальной сети (LAN) с Интернетом. Обычно скорость и протокол канального уровня модели ISO/OSI в канале связи между прокси-сервером и Интернетом отличаются от применяемых в локальных сетях. Прокси-серверы иногда используются как межсетевые экраны, в качестве средства дополнительной безопасности.

Клиент пересылает прокси-серверу, работающему с Web, запрос на конкретный документ. Тот получает документ по соответствующему протоколу (HTTP, FTP, Gopher и т. д.), и пересылает его клиенту. То, что запросы клиента передаются серверу по протоколу HTTP, не является ограничением, также не имеет значения тип затребованного документа, поскольку в самом запросе содержится полная информация о нем, в частности его тип и необходимый протокол. Прокси-сервер сам анализирует запрос и определяет адрес, по которому следует переслать запрос (другими словами, выполняет разрешение имени домена в IP-адрес).

Прокси-серверы реализованы на базе протокола, разработанного в лаборатории CERN.

Кэширование

Так как несколько клиентов могут обращаться к одному и тому же документу независимо друг от друга, прокси-сервер должен уметь выполнять кэширование, что повышает эффективность работы. Если документ находится в кэше, то сокращается время ожидания клиентом и трафик через Интернет.

Однако не все документы можно кэшировать. Доступ к некоторым страницам в Web открывается, только когда Вы внесете абонентскую плату. Документы, которые нельзя кэшировать, содержат теги WWW-Authenticate, Pragma:no-cache, Cache-control:private, Cache-control:no-cache и SetCookie. Кроме того, нельзя кэшировать устаревшие документы или те, для доступа к которым необходима аутентификация.

Для кэширования применяются различные средства. Одна из технологий — *пассивное кэширование* (passive caching) — заключается в том, что прокси-сервер просто ожидает запрос, а затем осуществляет поиск соответствующего документа и определяет, следует ли проводить кэширование или нет.

Другая технология — *активное кэширование* (active caching). В этом случае прокси-сервер в периоды низкой активности пользователей извлекает те документы, которые, по его мнению, с большой долей вероятности будут запрашивать клиенты.

Крупные компании имеют несколько прокси-серверов. Для управления кэшированием нескольких прокси-серверов применяют специальные протоколы (обсуждаются в следующих подразделах). Замечу, что основные правила взаимодействия между прокси-серверами и клиентами описаны в протоколах ICP (Internet Cache Protocol) и CARP (Cache Array Routing Protocol). Они не препятствуют использованию пассивного или активного кэширования, а наоборот, требуют одно из них.

Протокол ICP

Протокол ICP (Internet Cache Protocol) разработан в Университете Южной Калифорнии (University of Southern California) и описан в RFC 2168. Он задает формат сообщений, используемых для взаимодействия между Web-прокси-серверами. В этих сообщениях обычно содержится информация о наличии или отсутствии определенной Web-страницы в кэше прокси-сервера. Также ICP-сообщения применяются для выбора кэша

какого-либо конкретного Web-сервера в качестве источника определенной Web-страницы. Формат ICP-сообщений описан в RFC 2168. В RFC 2167 указаны подробности реализации протокола ICP, использующего перечисленные в RFC 2168 сообщения. Разработаны ICP-расширения и для протокола IPv6, но они пока не стандартизованы.

Web-прокси-серверы, в которых реализован ICP, обычно, используют три разных порта. Первый — HTTP-порт — применяется клиентами для обращения к кэширующему серверу. Второй — UDP-порт — для обмена ICP-сообщениями. Третий — TCP-порт — для извлечения Web-страниц из кэша прокси-сервера. Спецификации RFC, описывающие протокол ICP, не уточняют тип транспортного протокола, но обычно это UDP.

Протокол ICP весьма полезен и широко распространен, однако последнее время его применение породило определенные проблемы. Одна из них — ограниченная масштабируемость, поскольку число ICP-сообщений между прокси-серверами быстро растет при увеличении количества этих серверов. Другая проблема заключается в чрезмерной избыточности кэшируемых прокси-сервером данных. Она появляется из-за того, что, получив запрос на определенную Web-страницу, прокси-сервер выполняет следующие действия:

- просматривает локальный кэш;
- если требуемая информация не найдена, посылает ICP-сообщение с запросом;
- в случае положительного ICP-ответа производит поиск требуемой информации в кэшах соседних прокси-серверов и помещает ее в локальный кэш;
- удовлетворяет запрос, передавая вновь найденные данные.

Протокол CARP

Протокол CARP (Cache Array Routing Protocol) предоставляет средства эффективного использования нескольких прокси-серверов в качестве единого унифицированного кэша. В этом протоколе исправлены недостатки существующих протоколов кэширования, например ICP. CARP позволяет обойтись без обмена информацией между прокси-серверами, а также избежать ситуации, когда содержимое кэшей разных прокси-серверов становится зеркальным отражением друг друга. Ниже в несколько упрощенной форме описаны основные функциональные возможности протокола CARP.

При использовании CARP прокси-серверы организуются в массив, который периодически обновляется, отражая информацию о вновь подключившихся серверах, а также о тех, в которых произошел аварийный сбой или отключенных. Для каждого прокси-сервера вычисляется хэш-функция его имени. Когда клиент запрашивает какой-либо URL, производятся некоторые вычисления, в которых участвует хэш-функция от этого URL и хэш-функции каждого прокси-сервера. Прокси-сервер, которому соответствует наибольший результат этих вычислений, и обрабатывает запрос данного URL. Необходимая страница либо уже содержится в кэше данного прокси-сервера, либо, по возможности, загружается в кэш. Замечу, что вычисления выполняются для каждого сервера и каждого URL. Это значит, что запросы к *http://www.company.com./page1.html* и *http://www.company.com./page2.html* иногда обрабатывают разные прокси-серверы. После определения прокси-сервера, отвечающего за конкретную Web-страницу, клиент опрашивает его. Если соответствующая страница есть в кэше, то прокси-сервер извлекает ее. Если нет, то эта страница загружается из Web и помещается в кэш.

Протокол CARP часто называют *распределенным кэшированием без запросов* (queryless distributed caching). Такое название связано с тем, что Web-страницы кэшируются несколькими прокси-серверами, и для определения того, какой именно сервер имеет копию конкретной Web-страницы, не надо никаких запросов. Протокол CARP передан на рассмотрение в IETF в качестве чернового стандарта. Самые новые спецификации (на момент издания этой книги) Вы найдете по адресу *http://www.microsoft.com/proxy/guide/CarpSpec.asp*. Официальные документы по CARP — *http://www.microsoft.com/proxy/guide/CarpWP.asp*. Протокол CARP поддерживается продуктами компаний Microsoft и Netscape.

Технология ActiveX

Технология ActiveX — это базовое средство динамического расширения возможностей как Web-клиентов (браузеров), так и Web-серверов. Действительно, последние версии Internet Explorer все шире и шире используют управляющие элементы ActiveX. Например, Internet Explorer версии 3.0 и более поздние для отображения кода HTML применяют управляющий элемент ActiveX. Его же могут задействовать и другие, совершенно независимые приложения, например финансовая программа или приложение, передающее котировки фондового рынка. Для создания управляющих элементов ActiveX не требуется специальный язык программирования. Их реализуют на языках Visual C++, Visual Basic или Java. В дополнение к этому замечу, что технология ActiveX поддер-

живается рядом платформ, в том числе Windows, Macintosh и некоторыми разновидностями UNIX.

Технология ActiveX основана на технологии COM (Component Object Model) компании Microsoft. Технология COM также претерпела ряд модернизаций. Ее исходная версия называлась OLE (Object Linking and Embedding). В OLE 1.0 введено понятие составного документа. В OLE 2 введено понятие COM. Термин ActiveX появился, главным образом, в качестве замены OLE, хотя между ними есть ряд технологических различий (они описаны ниже в этом разделе).

Технология COM обеспечивает взаимодействие двух объектов одного компьютера. Эти объекты могут быть частями одного процесса или принадлежать разным процессам. Технология DCOM (Distributed COM) — аналог COM, но позволяет взаимодействовать объектам разных компьютеров. При этом используется тот же интерфейс связи объектов независимо от того, находится ли второй объект на локальном или на удаленном компьютере.

Еще одна важная часть OLE — концепция «*псевдонимов*» (monikers). «Псевдоним» — это OLE-объект, который знает, как инициализировать другой OLE-объект. Сам «псевдоним» инициализируется создающим его OLE-объектом. В зависимости от типа «псевдонима» (и того, что он делает), иногда проще создать не OLE-объект, а инициализирующий его «псевдоним».

До появления ActiveX для выполнения специальных функций, например отображения HTML, использовались управляющие элементы OLE (OCX). Управляющие элементы ActiveX и OLE похожи, но есть и несколько важных различий:

- управляющие элементы ActiveX более просты. Они не требуют реализации всех интерфейсов, необходимых для OCX. Кроме того, для взаимодействия объектов ActiveX надо меньше затрат;
- для повышения эффективности ActiveX допускает использование асинхронных «псевдонимов»;
- управляющие элементы ActiveX поддерживают механизмы защиты и подписи кода.

Технология ActiveX включает ряд базовых технологий. Некоторые из них применимы только к Web-клиенту, другие — только к Web-серверу, а третьи — к обоими. В следующих разделах эти технологии подробно описаны.

Управляющие элементы ActiveX

Управляющие элементы ActiveX — основные «строительные блоки» технологии ActiveX. Когда упоминают технологию ActiveX, речь, как правило, идет об управляющих элементах. Управляющий элемент ActiveX — это программный модуль, который не может функционировать самостоятельно. Он выполняется в пределах оболочки ActiveX-контейнера, например Web-браузера, текстового процессора или электронной таблицы. ActiveX-контейнер выполняет специальные функции. Управляющий элемент ActiveX способен выполнять практически любые, необходимые его создателю функции: доступ к базам данных, доступ к файлам, поиск по времени (по значению), отображение пользовательского интерфейса и т. д. Управляющий элемент может взаимодействовать с другим управляющим элементом, ActiveX-контейнером или операционной системой.

Существует ряд готовых к использованию управляющих элементов ActiveX. Так, в пакете FastNet (прежде Internet ActiveX Control Pack) компании NetMaster технологии связи в Интернете, например TCP, UDP, FTP и SMTP, реализованы в виде управляющих элементов ActiveX. Это позволяет не вдаваясь в детали создавать приложения, взаимодействующие по перечисленным протоколам.

Сценарии ActiveX

Средствами сценариев ActiveX можно вставлять код непосредственно в HTML-файл, избавляясь от предварительной компиляции. Код интерпретируется и выполняется браузером. В некоторых случаях его сначала приходится загружать.

Сценарии ActiveX поддерживают использование как JavaScript, так и VBScript — подмножество Visual Basic, ограниченное по соображениям безопасности. Сценарии ActiveX также поддерживают ActiveScript API, который обеспечивает плавный переход от существующих OCX-контейнеров к контейнерам сценариев ActiveX.

Строго говоря, в основе сценариев ActiveX лежат две базовые технологии:

- хост сценариев ActiveX — Web-браузер, сервер или другое приложение, например среда разработки или электронная таблица;
- машина сценариев ActiveX — интерпретатор сценариев; при этом поддерживаются разные языки, например Perl, VBScript, JavaScript, Lisp и др.

ActiveX-документы

ActiveX-документы не самая важная часть технологии ActiveX. Их строят на базе OLE-объектов DocObject, а отображают и редактируют средствами ActiveX-контейнера.

Безопасность в ActiveX

Для обеспечения подлинности и целостности динамически загружаемого и выполняемого кода в ActiveX используются разные технологии, в частности Authenticode и криптография с открытым ключом. (О безопасности см. главу 5.) Компания-разработчик может подписать свой управляющий элемент ActiveX. Для этого ей необходимо получить у компании VeriSign сертификат X.509 и подписать управляющий элемент с помощью своего секретного ключа и стандартов PKCS 7 и 10 (Public Key Cryptography Standards).(О PKCS см. главу 5.) При загрузке управляющего элемента контейнером для проверки подписи применяется Authenticode. Управляющий элемент выполняется только в том случае, если подпись подлинная. Пользователь вправе отказаться от подобных проверок, но это опасно.

Связь ActiveX и Java

Управляющие элементы ActiveX программируют на разных языках, в том числе на Java. Таким образом, Java-апплет сам по себе может быть и управляющим элементом ActiveX. Динамически загружаться и выполняться способны управляющие элементы ActiveX, Java-апплеты и компоненты JavaBeans. В технологии ActiveX управляющий элемент перед выполнением проверяется. В Java используется модель «песочницы» (sandbox) — изолированной среды, в пределах которой может «играть» Java-апплет или компонент JavaBeans. «Песочница» ограничивает возможности приложения, выполняющегося в ее пределах, например полностью запрещает ввод/вывод информации на локальный диск.

Принадлежность ActiveX

Технология ActiveX разработана преимущественно компанией Microsoft. Позже представители различных компаний, в том числе и Microsoft, сформировали группу под названием Active Group (подразделение Open Group), которая влияет на решения Microsoft о путях развития технологии ActiveX. Компания Microsoft передала Open Group то, что называют ограниченное право на ActiveX. Теперь Open Group, с благословения Microsoft, лицензирует технологию COM всем желающим.

Dynamic HTML, скриптлеты и модель DOM

Именем Dynamic HTML (DHTML) обозначается сочетание технологии CSS, языков сценариев и HTML. Основная задача DHTM — динамическое создание HTML-страниц, однако возможности гораздо шире. Он также применяется для изменения внешнего вида страницы в зависимости от реакции пользователя и без обращения к серверу.

Реализации DHTML от Microsoft и Netscape отличаются, хотя обе компании заявляют, что придерживаются стандарта HTML 4.0, утвержденного в декабре 1997 года.

Одно из отличий таково: вариант Netscape позволяет изменять таблицу стилей только при загрузке HTML-страницы, а реализация Microsoft — и после загрузки страницы. Netscape допускает создание DHTML только с использованием JavaScript, тогда как Microsoft — допускает и JavaScript и VBScript. Еще одно отличие касается обработки загружаемых или встраиваемых шрифтов. Применяя разные шрифты для HTML-страницы, Вы можете изменить ее вид от удобочитаемого до нечитаемого. Чтобы этого избежать, используемые шрифты связывают с самим документом. В реализации компании Netscape присутствует файл описания шрифтов, имеющий «указатель» на Web-страницу. Microsoft же расширила механизм CSS, добавив ссылки на файлы шрифтов. В основе современных методов сценариев лежат JavaScript/JScript/ECMAScript, VBScript и другие клиентские и серверные языки сценариев. Обычно программы (сценарии), написанные на этих «облегченных» языках программирования, встраиваются в HTML-документы, а затем интерпретируются браузером или сервером. Основная задача таких сценариев — обращение к объектам, встроенным в браузер или сервер, либо вызов локальных объектов, например управляющих элементов ActiveX, Java-апплетов, сервлетов (servlet) и т. д.

Современные механизмы сценариев не позволяют тесно взаимодействовать с HTML-документами, которые их содержат. Они предоставляют разработчикам лишь базовые возможности построения содержимого HTML. Серверные механизмы сценариев в этом смысле обычно немного мощнее: их часто используют для создания содержания для клиента.

Помимо доступа к HTML-странице, DHTML содержит новые мультимедийные управляющие элементы, которые допустимо применять в сценариях: Audio Mixer, Path, Sequencer, Sprite и Structured Graphics. Также DHTML поддерживает специальные графические эффекты (фильтры и переход цветов).

В DHTML вводится концепция объекта типа «событие». Объект типа «событие» вызывается, когда происходит определенное событие, например: кнопка мыши «щелкнула» в определенном месте HTML-страницы или указатель мыши попал в определенную ее область. Браузеры компании Microsoft позволяют писать обработчики событий как на JavaScript, так и на VBScript, тогда как браузеры компании Netscape — только на JavaScript.

Скриптлеты

В DHTML появилось новое понятие — *скриптлет* (scriptlet) — HTML-документ, который ведет себя, скорее, как объект, нежели как документ. Скриптлет пишется на языке сценариев, и он может функционировать как обычный HTML-документ со сценарием. Он также содержит методы, доступные с других Web-страниц со сценариями. Использование скриптлетов полезно в тех случаях, когда для решения какой-либо задачи HTML-документа со сценарием недостаточно, но эта задача не настолько сложна, чтобы создавать новый управляющий элемент ActiveX или класс на языке Java.

Подробнее о скриптлетах — на узле *http://www.microsoft.com/scripting*.

Модель DOM

Модель DOM (Document Object Model) — это относительно новая технология. В настоящее время ее прорабатывает консорциум W3C. Ее цель — создание набора объектов, независимых от платформ и языков, которые позволят программам динамически обновлять документы. Ожидается, что DOM сможет взаимодействовать с Java и ECMAScript, COM, Perl, VBScript и C++.

Язык Java

В следующих разделах описаны Java-технологии и взаимосвязь Java с другими протоколами.

Java RMI и сериализация объектов

Для того чтобы Java-приложения могли выполняться не только в локальных системах, их расширяют средствами Java RMI (Remote Method Invocation). Эта технология предоставляет Java-классам базовые возможности распределенных вычислений, благодаря чему программа, установленная на одном компьютере, будет работать и на другом. Функционально технология RMI похожа на RPC (Remote Procedure Call), но вместо процедур использует Java-объекты и требует установки Java на обоих компьютерах.

Чтобы Java-классы могли осуществлять ввод/вывод через сеть, в RMI применяется *сериализация* (serialization) — кодирование объектов и параметров в байтовый поток и обратно.

Java и CORBA

Архитектура CORBA (Common Object Request Broker Architecture) позволяет вызывать объекты, расположенные в любом месте сети. CORBA содержит язык IDL (Interface Definition Language) от OMG (Object Management Group), применяемый для написания оболочек объектов, специфичных для разных языков. Для упрощения абстрактных интерфейсов эти оболочки используют вспомогательную подпрограмму ORB (Object Request Broker). В Интернете CORBA функционирует через протокол IIOP (Inter-ORB Protocol), разработанный группой OMG.

Благодаря использованию Java IDL и Java ORB, объекты, написанные на Java, могут взаимодействовать с объектами, написанными на других языках.

Средствами Java IDL создаются «заглушки» клиентов и «каркасы» серверов, которые используются подпрограммами ORB.

В качестве «родного» интерфейса между Java-приложениями выступает RMI.

Java и COM

По умолчанию Java-приложение функционирует в одной системе. Для взаимодействия с иной сетевой программой через сокеты Java-приложение использует класс java.net; для взаимодействия с другим Java-приложением — RMI; а для взаимодействия с CORBA-объектами — CORBA.

В качестве альтернативы для связи с другими процессами (локально или по сети) может применяться технология COM. Она предоставляет способ взаимодействия объектов, не зависящий от языка. Изначально COM ориентировали на платформу Windows, но в настоящее время ее приспособили к ряду других платформ.

При интеграции технологии COM в виртуальную машину Java (JVM) ее поддержка может стать прозрачной (незаметной для приложений). В рамках JVM каждый COM-объект предстает перед Java-приложениями как отдельный Java-класс. Вне JVM каждый Java-класс предстает перед COM-объектами как другой COM-объект. Посредством DCOM можно создавать распределенные системы. В случае интеграции COM в JVM (как, например, в Microsoft JVM) уровень описания языково-независимого интерфейса становится прозрачным.

Технология JavaBeans

Технология JavaBeans позволяет Java-приложениям абстрагироваться от интерфейсов различных объектных моделей (CORBA, COM, OpenDoc и др.). Эта технология представляет Java-ориентированную объектную модель, в которой создаются Java-приложения для нее же, а для работы с другими объектными моделями используются соответствующие оболочки. Применяя методы JavaBeans API, можно собирать и публиковать информацию об интерфейсах (об их свойствах и оболочках). Далее данные попадают на уровень описания интерфейса различных объектных систем и используются различными средствами разработки приложений. В дополнение к этому API предоставляет методы обработки событий, хранения компонентов и управления форматом отображения.

Интерфейс JNDI

Интерфейс JNDI (Java Naming and Directory Interface) позволяет Java-приложениям обращаться к службам имен и каталогов. JNDI предоставляет разработчикам унифицированный доступ к пространствам имен разнородных корпоративных сетей. Поставщики услуг, входящие в систему, используют протокол LDAP (версий 2 и 3) и службу NIS (Network Information Service). Разработчики применяют JNDI для добавления своих собственных служб к другим протоколам служб каталогов.

Java Web Server и сервлеты

Изначально Java применялся на стороне клиента, в Web-браузере. Апплет представлял собой маленькое приложение, которое выполнялось в контексте клиентского Web-браузера. Позднее Java стали применять для разработки серверных программ.

Некоторые из написанных на Java серверов, например Java Web Server компании Sun, предоставляют новый Java-ориентированный программный интерфейс Servlet API для выполнения Java-*сервлетов* (servlet). Концептуально сервлеты напоминают CGI-приложения и являются Java-классами, которые выполняются Web-сервером. Servlet API — это маленький программный интерфейс, не предполагающий никакого пользовательского интерфейса (UI) и не зависящий от протоколов. Сервлеты обычно запускаются из локальной файловой системы сервера — они более надежны, чем «посторонние» апплеты, а также используют цифровые подписи, поддерживаемые Java. Утилита Security Manager позволяет администратору сервера устанавливать сервлетам права доступа (например, для ввода/вывода в локальный файл). Как и в случае других, разработанных позднее CGI, интерфейсов Web-серверов, сервлеты спо-

собны загружаться динамически и оставаться в памяти, что сокращает затраты на запуск и завершение процессов.

Сервлеты можно использовать не только с Web-серверами, но и с другими типами серверов, например с почтовыми. Если даже сервер не поддерживает необходимый интерфейс, сервлеты предоставляют удобные средства дополнения функциональных возможностей различных серверных программ.

HotJava Browser и HotJava Views

HotJava Browser — это Web-браузер, полностью написанный на Java. Его можно дополнять новыми обработчиками содержимого или протоколов, что расширяет возможности браузера для работы с различными типами данных и сетей.

Среда HotJava Views нацелена рынок сетевых компьютеров («тонких» клиентов). В ней HotJava Browser используется в качестве рабочего стола (или Webtop). HotJava Views, будучи оболочкой графического интерфейса пользователя (GUI), внешней по отношению к Webtop, содержит, помимо прочих приложений, простейший клиент электронной почты, адресную книгу, Selector (диспетчер процессов) и Web-клиент HotJava Browser. HotJava Views позволяет разработчикам создавать Java-приложения и интегрировать их с Webtop, применяя предоставляемые HotJava интерфейсы.

Технология JavaSpace

JavaSpace — технология обмена данными и объектами между Java-программами. Данные считаются распределенными и постоянно обновляющимися. Использование JavaSpace дает Java-приложениям возможность сохранять и восстанавливать состояния распределенным способом. Хотя JavaSpace отдаленно напоминает сетевую базу данных, но она не является реляционной или объектной базой данных.

Набор интерфейсов JMAPI

JMAPI (Java Management API) — это набор интерфейсов, позволяющих создавать для Java-платформ приложения для управления сетями.

Раньше подобные программы разрабатывались для базового API языка Java и применяли протокол SNMP (Simple Network Management Protocol). Новые управляющие приложения создаются на базе пользовательского интерфейса модуля AVM (Admin View Module), написанного на Java, что позволяет интегрировать их с различными решениями по управлению сетями. Другие базовые объектные интерфейсы предоставляют

доступ к другому коду и данным, необходимым для написания законченного модуля управления сетью.

Ссылки

http://info.internet.isi.edu/in-notes/rfc/files/rfc2187.txt

http://info.internet.isi.edu/in-notes/rfc/files/rfc2186.txt

http://www.netscape.com/newsref/std/cookie_spec.html

http://home.netscape.com/assist/net_sites/pushpull.html

http://www.macromedia.com/shockwave/

http://www.javasoft.com/

http://www.microsoft.com/com/default.htm

http://www.w3.org/PICS/

http://www.w3.org/XML/

http://www.w3.org/

http://www.sgml.org/

http://www.microcsoft.com/standards/xml

http://www.microsoft.com/standards/cdf-f.htm

http://www.microsoft.com/standards/osd/osdspec-f.htm

http://www.w3.org/metadata/rdf/overview.html

http://www.microsoft.com/xml/cdf/

http://developer.netscape.com/one/javascript/

http://www.ecma.ch/stand/ecma-262.htm

http://www.microsoft.com/scripting/

http://www.microsoft.com/vbscript/

http://www.netscape.com/newsref/std/server_api.html

http://tech.west.ora.com/win-httpd/httpddoc/wincgi.htm

http://www.microsoft.com/iis/

http://www.microsoft.com/scripting

http://www.javasoft.com/products/jdk/rmi/index.html

http://www.javasoft.com/

http://www.microsoft.com/java/

http://jserv.javasoft.com/products/java-server/documentation/webserver1.1/servlets/intro.html

ГЛАВА 14

Электронная коммерция

Эта глава посвящена электронной коммерции, которая охватывает все финансовые операции в Интернете. Так как эти операции обычно конфиденциальны, Вам стоит изучить главу 5, посвященную основам криптографии. Также, поскольку большая часть коммерческих операций в Интернете совершается на Web-страницах, Вам полезно прочитать главы 12 и 13 — об основах Web-протоколов.

Торговые Web-серверы

Существует несколько протоколов для серверов, используемых для электронной коммерции в Интернете. В этом разделе я расскажу о наиболее важных из них.

Приложения для совершения покупок

Некоторе время назад электронная коммерция выглядела довольно просто. Web-торговцы создавали HTML-версии форм для заказа, покупатели их заполняли, распечатывали и присылали по обычной почте. Каталоги продавцов были интерактивными, но сам процесс заказа таковым не являлся.

Затем появились формы заказа, которые покупатели могли пересылать по Интернету. Для этого требовались клиентские HTML-формы, взаимодействующие с заказчиками, а также серверные интерфейс CGI и специальные приложения, принимающие и обрабатывающие данные. От пользователей требовалось занести данные в одну или несколько HTML-форм. Часто внешний HTML-интерфейс генерировался динамически, в зависимости от базы данных товаров.

Позднее были разработаны более сложные приложения: теперь пользователи могли просматривать интерактивные каталоги и заносить предметы с нескольких страниц в *виртуальные списки покупок* (virtual shopping cart). Система, установленная на сервере, запоминала все выбранные предметы и объединяла их в общий список по окончании

посещения виртуального магазина. Такие приложения известны как *приложения для совершения покупок* (shopping cart application).

Чтобы сохранить список выбранных предметов, приложения для совершения покупок используют специальные маркеры (cookies) и HTTP-заголовки. Помните, что HTTP — протокол без памяти (см. главу 12), то есть каждая HTTP-команда выполняется независимо от предшествующей. По этой причине обычно используют некоторый маркер, позволяющий логически объединить последовательности действий в HTTP-транзакции. Этот маркер должен быть доступен серверу, для чего его указывают как параметр в последующих URL динамических HTML-страниц или включают в другой HTTP-заголовок.

Открытый HTTP

Web-транзакции могут выполняться с применением обычных Web-протоколов и кредитных карт. Пользователь может посылать продавцу информацию о платеже посредством HTML-страниц и HTML-форм. Однако, поскольку сообщения протокола HTTP никак не защищены (вся информация передается в открытом виде как ASCII-текст), вероятен анализ перехваченных открытых данных и извлечение платежной информации, например номеров кредитных карт.

S-HTTP

Защищенный протокол S-HTTP (Secure HTTP), разработанный компанией Enterprise Integration Technology (EIT), является расширением протокола HTTP. Он позволяет браузерам и серверам подписывать, аутентифицировать и шифровать сетевые HTTP-пакеты. Данный протокол, так же как и обычный HTTP, использует MIME (Multipurpose Internet Mail Extensions) для добавления заголовков, указывающих на необходимость специальной обработки, и для хранения информации о сертификатах и ключах. Процесс установления связи между браузером, поддерживающим протокол S-HTTP, и сервером несколько сложнее аутентификации в обычном HTTP. В настоящее время протокол S-HTTP используется редко, потому что его в значительной степени потеснил SSL (Secure Sockets Layer).

HTTP и SSL

SSL (Secure Sockets Layer) — это метод шифрования, разработанный компанией Netscape Communications для сокетов TCP/IP. Как и S-HTTP, SSL обеспечивает защиту сетевых HTTP-пакетов. Отличие же в том, что S-HTTP работает только на уровне протокола HTTP, а SSL — на уровне

сокетов, обеспечивая безопасность ряда других протоколов Интернета, использующих сокеты. Процесс согласования между клиентом, поддерживающим SSL, и сервером подобен процессу согласования обычных сокетов, но в первом случае (с SSL) клиент и сервер дополнительно передают сообщения для выбора алгоритма шифрования и обмена информацией о сертификатах и ключах.

При применении SSL с протоколом HTTP, Web-браузер использует URL-схемы вида «https://» вместо «http://», как для открытого HTTP. Многие Web-браузеры сообщают пользователю (с помощью диалоговых окон, изображений ключиков скрепленных или порванных связей, и т. д.), что транзакции на основе HTTPS более безопасны, чем обычные HTTP-транзакции .

Сейчас SSL используется повсеместно для обеспечения безопасности Web-транзакций общего назначения и практически заменил S-HTTP. Развитием SSL (и протоколов транспортного уровня) занимается рабочая группа Transport Level Security (TLS) в составе IETF.

Системы торговых серверов

В настоящее время в Web-коммерции используются в основном традиционные Web-протоколы, а для передачи конфиденциальной информации о платежах — HTTP, зашифрованный средствами SSL. Существует много приложений для совершения покупок, которые обеспечивают подобные функциональные возможности. Специализированные серверные системы заменяются коммерческими продуктами для торговых серверов. Эти программы обычно содержат инструментальные средства для помещения информации о товарах из базы данных в Web посредством сценариев и HTML-страниц. Они также включают модульные подсистемы для обработки оплаты, поставки и объединения с другими компьютерными системами.

Более сложные системы выполнят проверку правильности платежа в реальном времени, что требует взаимодействия с банком, обслуживающим счета продавца. Менее сложные системы откладывают проверку правильности платежа до тех пор, пока запрошенная транзакция не будет перенесена в обычную расчетную систему продавца.

Вот несколько примеров крупных коммерческих программ: Microsoft Site Server, Netscape Merchant System, Open Market Commerce Server, Net Commerce от IBM и Electronic Commerce Suite от iCat.

Протокол SET

SET (Secure Electronic Transaction) — это протокол электронных платежей на основе банковских карт, разработанный консорциумом компаний во главе с MasterCard и Visa. Он превращает финансовые операции в виртуальные, чем-то напоминающие манипуляции с кредитной карточкой.

Покупая товары с помощью обычных кредитных карт, покупатели взаимодействует с продавцами. Последние для проверки сделки обращаются к банкам, принимающим карточки к оплате, а банки-эммитенты взаимодействуют с держателями карт для получения денег за покупку. Применив эту, всем знакомую схему к электронным транзакциям, компании MasterCard, Visa и другие разработали протокол SET. Он использует тот же самый базовый процесс, но заменяет реальную кредитную карточку интерактивным протоколом. Протокол SET сохраняет иерархию доверительных отношений, что существует между продавцом и банком, принимающим платеж, а также владельцем карты и банком-эммитентом. В электронном варианте эта иерархия доверия формализована посредством цифровых сертификатов и иерархии сертификационных служб (аналог финансовой системы экономики).

SET заменяет две отдельных технологии: STT (Secure Transaction Technology) от Visa и Microsoft и SEPP (Secure Electronic Payment Protocol) от MasterCard и IBM. К счастью, преимущества единой ифраструктуры были настолько очевидны, что два протокола объединили в один — SET.

В некоторых случаях SET лучше, чем существующий механизм обычных кредитных карточек, особенно при обеспечении конфиденциальности потребителя. Транзакции зашифрованы так, чтобы торговцы видели только информацию о приобретаемых предметах, но не имли доступа к данным о лицевом счете клиента. Аналогично, банкам недоступна информация о покупаемых товарах, им видны лишь запросы кредитования продавца.

Протокол SET не привязан ни к HTTP, ни к Web-коммерции. Его можно использовать с самыми разными протоколами. SET не простой протокол: его сообщения и структуры данных описаны в формате ASN.1 (Abstract Syntax Notation One). В отличие от SSL, SET шифрует сами сообщения, а не канал связи. В результате возможны более сложные шифры, нежели многоцелевые (как в SSL).

Структура SET-запроса разбивается на две части — информация о заказе (Order Information, OI) и информация о платеже (Payment Infor-

mation, PI). Последняя нужна банку; но не продавцу. (Продавцу достаточно получить от банка подтверждение платежа.) Все данные, относящиеся к одной транзакции, скрепляются методом *двойной подписи* (dual signature).

Этот метод использует два различных сообщения, одно из которых адресовано продавцу, а другое — банку. Эти сообщения взаимосвязаны, поэтому банк может сопоставить платеж с заказом, не зная всех деталей последнего. Продавец получает гарантии лишь относительно оплаты заказа и размера платежа, но не владеет никакой другой информацией (например, не знает номер счета — эквивалент шестнадцатизначного номера кредитной карточки — заказчика).

Каждое сообщение обрабатывается алгоритмом дайджеста сообщений, который вырабатывает 160-битный дайджест. Затем два дайджеста соединяются, и результат обрабатывается этим же алгоритмом. Конечный вариант покупатель подписывает с помощью своего личного ключа. Это и есть двойная подпись. Затем покупатель отправляет два сообщения. Одно — продавцу, в нем описаны детали заказа и содержится дайджест сообщения, посланного банку. Другое сообщение — в банк; в нем указана стоимость транзакции и содержится дайджест сообщения, посланного продавцу.

И банк, и продавец могут проверить подлинность полученного сообщения. Для этого необходимо вычислить дайджест принятого сообщения и объединить его с дайджестом другого сообщения. Вычисленный общий дайджест должен совпадать с расшифрованной (при помощи открытого ключа покупателя) двойной подписью. Обратите внимание на то, что этот метод позволяет покупателю торговаться при покупке. Продавец посылает банку сообщение только в том случае, если принимает предложение покупателя. Дайджест сообщения связывает транзакцию между продавцом и покупателем с сообщением межу банком и покупателем, подтверждающим платеж.

SET явно использует два вида шифрования: симметричное и асимметричное, а также сертификаты для сопоставления открытого и личного ключей. Многие компании, производящие программное обеспечение, выпускают API для работы с протоколом SET. Большинство приложений, использующих SET, например системы торговых серверов и аналогичные им объекты клиентской стороны (своего рода виртуальные кошельки), применяют эти программные библиотеки.

Электронные деньги

Помимо существующих методов электронных платежей, копирующих реальные системы оплаты (например, кредитные карточки), развиваются новые виды электронных денег. Некоторые из них полностью программные, а другие, например смарт-карты, требуют дополнительных аппаратных средств.

Для большинства видов электронных денег разработаны специальные протоколы обмена финансовыми заявками между клиентом, продавцом и эммитентом денег (банком). Эти протоколы реализованы как для серверного ПО, так и для клиентского — электронного бумажника.

Вот некоторые примеры существующих технологий электронных денег:

- CyberCash и CyberCoin от CyberCash Inc. (*http://www.cybercash.com*);
- eCash от DigiCash Inc. (*http://www.digicash.com*);
- MilliCent от Digital Equipment (*http://www.millicent.digital.com*);
- iKP (Internet Keyed Payment Protocols) от IBM (*http://www.zurich.ibm.ch/Technology/Security/extern/ecommerce/iKP.html*);
- NetCheque, производитель — University of Southern California's Information Sciences Institute (*http://www.netcheque.org*);
- NetBill, производители — Carnegie Mellon University's Information Networking Institute, Mellon Bank и CyberCash (*http://www.netbill.com*);
- Mondex от Mondex International (*http://www.mondex.com*);
- Interactive Messaging Platform от First Virtual (*http://www.firstvirtual.com/services/imp/index.html*);
- Netscape LivePayment от Netscape (*http://live.netscape.com/comprod/products/iapps/livepayment.html*);
- MiniPay от IBM (*http://www.hrl.il.ibm.com/mpay/*).

Микроплатежи

Электронные деньги имеют значительное преимущество перед реальными: они позволяют осуществлять микроплатежи — платежи, размер которых меньше, чем существующие денежные единицы. В США это означает возможность расчетов на суммы менее одного цента. Система кредитных карт этого не допускает. Микроплатежи часто связаны с «нематериальными товарами», такими, как доступ к Web-страницам.

Существующие системы для обычных платежей не всегда корректно работают со столь малыми суммами.

Электронные бумажники

Многие виды электронных денег требуют наличия у клиента так называемого электронного бумажника — программного обеспечения, которое посылает информацию о финансовой сделке на сервер, используя специальный протокол. Некоторые разработчики программного обеспечения создают API, чтобы для различных электронных денег Вы могли использовать один и тот же «бумажник» — так же, как в реальный бумажник Вы не задумываясь кладете кредитные карты различных платежных систем.

Другие компании применяют электронные бумажники для иных целей. Некоторые Web-узлы имеют гостевые книги, в которые пользователей просят сообщить информацию о себе и о своих предпочтениях для динамического формирования содержания узла. Часть этой информации хранят в «бумажниках». Когда такие сведения помещены в одно место, где четко определены процедуры их запроса и обработки, удобнее использовать электронный бумажник (небольшую клиентскую базу данных, содержащую, например, номер кредитной карточки, информацию о цифровой подписи и сведения для идентификации и аутентификации) для нефинансовой информации. Кстати, в обычном бумажнике часто мы тоже храним массу нужных нам вещей, не имеющих отношения к деньгам: листочки с телефонами, записочки, фотографии и др.

Смарт-карты

Смарт-карты (или «интеллектуальные карты») внешне напоминают обычные кредитные карточки, но содержат микропроцессор, который запоминает определенную информацию. Их применяют в основном при финансовых сделках. Смарт-карты используют цифровые подписи и другие криптогафические методы защиты, что делает их более защищенными от фальсификаций, чем обычные кредитные карточки.

Существует целый ряд стандартов для смарт-карт. Организация ISO разработала стандарты ISO 7816, определяющие совместимость на аппаратном уровне и на уровне канального протокола контактных смарт-карт на основе интегральных микросхем. Europay, MasterCard и Visa разработали спецификации смарт-карт, расширяющие стандарты ISO 7816. Эти спецификации содержат дополнительные типы данных и правила кодирования, позволяющие применять смарт-карты в финансовой сфере. Специалисты GSM (Global System for Mobile Communications) разра-

ботали спецификации, позволяющие с помощью смарт-карт идентифицировать и аутентифицировать пользователей мобильных телефонов. Рабочая группа PC/SC (Personal Computer/Smart Card) определила схему и API для взаимодействия смарт-карт с персональными компьютерами, а OpenCard Framework — для взаимодействия с OpenCard-совместимыми платформами (включая сетевые компьютеры).

Java Commerce

Java Commerce — это совокупность технологий, переносящих электронную коммерцию на платформу Java. Существует расширение платформы Java — Java Electronic Commerce Framework (JECF), позволяющее использовать приложения для электронной коммерции, например Java Wallet. Средствами программного интерфейса Java Commerce API можно создавать торговые системы для платформы Java. Java Commerce API расширен посредством «кассет» — объектов, подобных компонентам JavaBean.

Взгляд в будущее

Перспективы развития электронных денег пока туманны. Технологии, управляющие денежными потоками, разрабатывают осторожнее, чем многие другие. В связи с жесткими требованиями конфиденциальности и аутентификации сложно создать единое решение для государственного, коммерческого и личного использования. Шифрование и сертификаты только добавляют забот. Кроме того, очень важно выбрать программное обеспечение, поддерживающее лишь необходимые серверы и клиенты (не больше того).

Ссылки

Дополнительную информацию об электронной коммерции Вы найдете в книге Давида Козье (David Kosiur) «Understanding Electronic Commerce» (Microsoft Press, 1997).

http://test.team2it.com/rsa/q133.html

http://home.netscape.com/assist/security/ssl/index.html

http://www.ietf.org/html.charters/tls-charter.html

http://www.mastercard.com/set or http://www.visa.com/set

http://www.smartcardsys.com

http://www.opencard.org

http://java.sun.com/commerce

http://www.w3.org/ECommerce

Часть 5

Мультимедиа

ГЛАВА 15

Мультимедиа в Интернете

В этой главе описаны технические подробности реализации мультимедиа в сети Интернет. Цель — обозначить стандарты, а также методы и программные продукты отдельных производителей.

Требования некоторых мультимедиа-приложений к пропускной способности проиллюстрированы в табл. 15-1.

Таблица 15-1. *Требования к пропускной способности*

Задача	Требуемая пропускная способность (без сжатия)	Допустимый метод передачи данных
Телефонный разговор	64 кбит/с	Integrated Services Digital Network (ISDN)
Качество звучания на уровне компакт-диска (CD-качества)	Чуть выше 700 кбит/с	Ethernet, xDSL (Digital Subscriber Line)
Телевидение высокой четкости (High-definition television, HDTV)	2 Гбит/с	Asynchronous Transfer Mode (ATM).

При ограниченной пропускной способности в Интернете (и, что еще важнее, пропускной способности соединения конечного пользователя с Интернетом) большую роль играет *компрессия* (compression), или *сжатие*. Это единственный способ передачи мультимедиа через Интернет. Однако не всякая методика сжатия приемлема. Ниже перечислены основные требования.

Алгоритм компрессии (и декомпрессии) должен работать в режиме реального времени, тогда его можно использовать в приложениях для видеоконференций и видеотрансляций, например спортивных репортажей или прямого эфира в программе новостей. (Особые случаи, например трансляция видеофильмов, записанных и сжатых заранее, позволяют для компрессии использовать длительный и сильно нагружающий процессор алгоритм, но для декомпрессии — только простой.)

Алгоритм компрессии должен обладать эффективным методом обработки ошибок в связи с реальной возможностью потери пакетов данных в Интернете, и не должен зависеть от получения предыдущего и последующего пакетов.

Некоторые алгоритмы сжатия отбрасывают несущественные данные, а значит, после декомпрессии данные не идентичны исходным. Потери допустимы, но в разумных пределах. В случае видео это особенно важно: потеря данных при компрессии/декомпрессии приводит к получению некачественного изображения. Алгоритмы компрессии/декомпрессии часто называют *кодеками* (codec).

Эффективность кодеков проиллюстрирована в табл. 15-2, где указана необходимая пропускная способность для тех же прикладных задач, что в табл. 15-1, но при использовании кодека.

Таблица 15-2. *Требования к пропускной способности при использовании кодеков*

Задача	Требуемая пропускная способность (сжатые данные)	Допустимый метод передачи данных
Телефонный разговор	Меньше 5 кбит/с	Модем
Звук CD-качества	64 кбит/с	ISDN
HDTV	30 Мбит/с	FDDI (Fiber Distributed ata Interface), ATM

Кодеки выполняются программно или аппаратно. Аппаратные кодеки работают чрезвычайно быстро и не загружают центральный процессор. К недостаткам относится возможная высокая стоимость, отсутствие универсальности. Кодек, выполненный программно легко заменить другим или другими. К тому же они дешевле аппаратных. Недостатками программных кодеков считается их медлительность (по сравнению аппаратными) и большая загрузка центрального процессора.

Потоковые технологии

Еще один немаловажный метод адаптации мультимедиа для Интернета — *потоковые технологии* (streaming). Изначально Web представлял собой исключительно текстовую среду. Тогда (да и сейчас) полагалось полностью загрузить файлы, прежде чем просматривать их. Это приемлемо для файлов небольшого размера, но не для мультимедиа — они, чаще всего, огромны. В потоковых технологиях используется следующий алгоритм: перед воспроизведением создается буфер данных, клиент загружает в него часть файла, распаковывает полученный фрагмент

и начинает проигрывать его содержимое (звук или видео), не дожидаясь, пока придет остаток. В это время загружается последующий фрагмент. Воспроизведенные же порции файла периодически удаляются. Таким образом, ни в один момент времени мультимедиа-файл не доступен клиенту целиком.

Звук в Интернете

Несмотря на перегрузку и ограниченную пропускную способность каналов связи, звуковые средства в Интернете, по сравнению со средствами видео, процветают. В этом нет ничего удивительного, поскольку требования к пропускной способности для передачи звука значительно ниже, чем для передачи видеоизображения.

Звук хорошо сжимается. Речь человека содержит изрядное количество пауз, которые просто пропускаются. К тому же можно отсечь верхнюю и нижнюю части звукового спектра, без заметного снижения качества звука. Другими словами, для сжатия аудиофайла используют алгоритм, который «теряет» часть исходных данных.

Сжатые данные затем передаются через Интернет по протоколам TCP или UDP. Каждый протокол имеет свои «плюсы» и, соответственно, поклонников. Продукт Internet Wave компании VocalTec использует TCP. Преимущество TCP в том, что он снабжен встроенным механизмом управления потоком данных, который гарантирует, что сеть не будет переполнена мультимедиа-данными. Недостаток же таков: воспроизведение мультимедиа иногда прерывается из-за утери или задержки TCP-пакета. Временами лучше не дожидаться пропавшего пакета, а попробовать проиграть один звук (без видео), используя полученные пакеты, поскольку данные в потерянном пакете могут быть незначительны. Продукт RealAudio компании RealNetworks, Inc. (раньше — Progressive Networks) использует для передачи сжатых мультимедиа-данных протокол UDP. Преимущество UDP в том, что он проще и несколько быстрее, чем TCP. Тот факт, что доставка пакетов не гарантирована, не имеет значения, поскольку соотношение утерянных и полученных данных достаточно мало.

Для компенсации потерянных во время передачи пакетов применяют различные методы. VocalTec использует алгоритм прогнозирующего кэширования (predictive caching), когда пытаются «угадать» содержимое потерянных пакетов. RealAudio применяет чередование (interleaving), или «нарезку» (mincing). Например, фрагмент данных длиною 48 миллисекунд поделен на части по 1 миллисекунде. Вместо того чтобы поме-

стить четыре первые миллисекунды в первый пакет, четыре следующие во второй и т. д., в первый пакет помещают 1-ю, 13-ю, 25-ю и 37-ю миллисекунды. Второй пакет содержит данные 2-й, 14-й, 26-й и 38-й миллисекунды. Этот процесс показан на рис. 15-1.

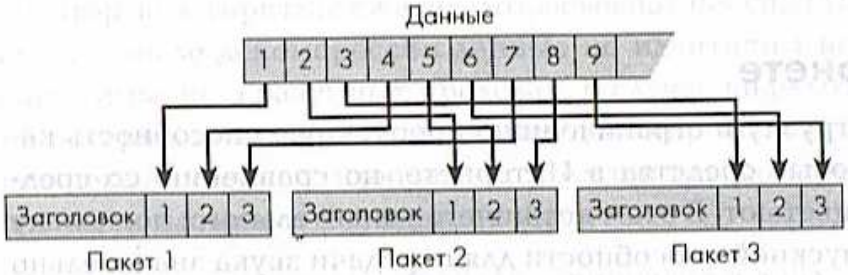


Рис. 15-1. *Метод чередования*

В результате при потере пакета выпадает по 1 миллисекунде на участке в 4 миллисекунды, вместо непрерывного фрагмента длиной 4 миллисекунды.

ПРЕДУПРЕЖДЕНИЕ

Числа, взятые для примера, могут не соответствовать используемым в RealAudio. Эти значения даны исключительно для иллюстрации методики.

VocalTec

Технология Internet Wave (I-Wave), разработанная VocalTec, использует в качестве транспортного механизма не UDP, а TCP. Преимущество TCP в наличии управления потоком данных и контроля перегрузок канала связи, гарантирующего, что сеть не будет «затоплена» мультимедиаданными.

RealAudio

RealAudio — это потоковая технология передачи голосовых данных через Интернет. В качестве транспортного механизма она использует UDP, а не TCP. Очевидно, UDP — более «легковесный» протокол, нежели TCP, с меньшими вычислительными затратами. RealAudio обладает собственным алгоритмом обработки утерянных UDP-пакетов. Недостаток TCP в том, что он ожидает доставку одного потерянного или опаздывающего пакета, чтобы представить данные в порядке следования. Если получены четыре пакета голосовых данных, а пятый утерян или задерживается, лучшее, что может сделать транспортный механизм, — подать данные «как есть», а не тратить время на ожидание, особенно если аудиоданных в одном пакете немного.

Технология RealAudio определена как патентованная клиент-серверная архитектура с патентованным протоколом. Позже этот протокол был представлен на рассмотрение IETF как черновой стандарт. Он описан далее в этой главе в разделе, посвященном протоколу RTSP. Протокол RTSP допускает двунаправленную связь между клиентом и сервером. (Не забудьте, что HTTP этого не позволяет.) Он обеспечивает такие возможности, как «быстрая перемотка» вперед или назад. Реализация RealAudio фактически состоит из двух отдельных серверов (рис. 15-2).

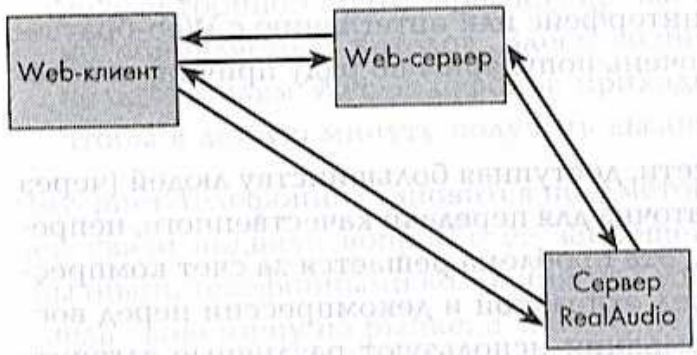


Рис. 15-2. *Клиент-серверное взаимодействие в RealAudio*

Когда файлы загружаются для просмотра в автономном режиме, клиент посылает запрос Web-серверу и получает от него ответ. Это при условии, что файлы самодостаточны и не содержат ссылок на другие документы. Когда пользователь просматривает данные интерактивно (например, в случае прямой трансляции бейсбольного матча), Web-сервер возвращает информацию о их местонахождении. Затем клиент посылает запрос мультимедиа-серверу RealAudio, получает данные, декодирует и воспроизводит их.

Между Web-сервером и сервером RealAudio возможен обмен сообщениями. Например, когда клиент посылает запрос Web-серверу, сервер иногда спрашивает, где найти данные, а в это время сервер RealAudio уже начнет выполнять требуемое действие. Однако подробности этого обмена сообщениями являются собственностью соответствующих фирм и не описаны ни в одном стандарте.

Средства RealAudio позволяют разрабатывать разные интересные программы, например аналог демонстрации слайдов с фоновым звуком. Все более популярными становятся прямые трансляции спортивных событий с помощью RealAudio. Просто посетите домашнюю страницу любой бейсбольной команды (например *http://www.mariners.org*), и Вы найдете гиперссылку на RealAudio, а также расписание трансляций матчей через Интернет.

Интернет-телефония

Программы, реализующие в Интернете аналог телефонной связи, разрабатываются рядом компаний, включая DigiPhone, VocalTec, NetSpeak, и Quarterdeck (существуют и бесплатные версии). Эти программы — *Web-телефоны* (Web phones) — позволяют людям разговаривать через Интернет. Основное преимущество такого общения — цена, равная стоимости подключения к локальному поставщику услуг Интернета. Web-телефоны предоставляют дополнительные возможности, например графический пользовательский интерфейс или интеграцию с Web-браузером. Пока Web-телефоны не очень популярны по ряду причин, некоторые из них описаны ниже.

- Пропускная способность сети, доступная большинству людей (через модем 28,8 кбит/с), недостаточна для передачи качественного, непрерывного звукового потока. Эта проблема решается за счет компрессии голосовых данных перед отправкой и декомпрессии перед воспроизведением. Разные компании используют различные алгоритмы компрессии/декомпрессии (кодеки). Это приводит к тому, что общаться могут только владельцы одинаковых Web-телефонов, произведенных одной компанией. Продукт WebPhone (от NetSpeak Corporation) на компьютерах с процессором Pentium использует для компрессии/декомпрессии алгоритм TrueSpeech. Он достигает коэффициента сжатия до 18:1. На компьютерах с 486-м процессором WebPhone реализует стандарт GSM (Global System for Mobile Communications), применяемый в сотовой телефонии. GSM обеспечивает коэффициент сжатия до 5:1. Разные компании работают по различным протоколам передачи данных. Кроме того, алгоритмы сжатия и пропускная способность линий связи могут снизить четкость звука, он в отличие от слышимого по обычному телефону становится не так отчетлив.
- Сегодня многие компьютеры оснащены полудуплексной звуковой платой. Это означает, что разговор происходит по правилам, напоминающим CB-радио (переходя в режим слушателя, вам придется говорить «прием»). Это неудобно и непривычно для людей, привыкших к обычным телефонам.
- Чтобы начать разговор через компьютер, необходимо знать IP-адрес вызываемой стороны. Многие пользователи при подключении к Интернету получают IP-адрес динамически от сервера поставщика услуг Интернета. Очевидно, адрес может меняться от сеанса к сеансу. Разработчики и поставщики Web-телефоов обходят эту проблему,

создав центральный сервер. Когда Web-телефон запущен на пользовательском ПК, он отправляет IP-адрес (временно присвоенный) центральному серверу, который в свою очередь постоянно обновляет список людей, соединенных с сетью и готовых принимать вызовы на Web-телефон. Такую схему реализует компания VocalTec для Internet Phone. Quarterdeck действует аналогично — поддерживает целую сеть серверов, которые предоставляют программе WebTalk службу каталогов. DigiPhone применяет другое решение: рассылает по электронной почте зарегистрированным пользователям ежедневно обновляемый каталог. Здесь возникает еще одно неудобство: пользователям Web-телефонов приходится не выключать свой ПК, чтобы в любую минуту получить вызов.

Интернет-телефония становится предметом дискуссии: операторы дальней связи подняли вопрос о ее запрещении или контроле наравне с обычными телефонными компаниями. Однако Интернет-телефония уже нашла свою нишу на рынке, а значит, уже сформировался круг людей, не согласных с операторами дальней связи.

Основы сжатия видео

Мультимедийная трансляция заключается в передаче и показе последовательности изображений. Если изображение не «автономно», а является частью видеоклипа, его обычно называют *кадром* (frame). Сжатие необходимо, чтобы просто уменьшить объем видеоданных до управляемого размера. Некоторые методы предназначены для сжатия одиночного кадра, безотносительно какого-либо другого. Такие методы называются *внутрикадровым кодированием* (intra-frame coding). Сжатие видеоданных основано на устранении избыточности в видеоизображениях. Существует три варианта избыточности. Они перечислены ниже.

- *Пространственная избыточность* (spatial redundancy). Пикселы внутри изображения имеют пространственную связь друг с другом, например один пикселы граничит с другим. Пространственная избыточность имеет место, если граничащие пикселы окрашены в одинаковый цвет или имеют одинаковую интенсивность. Это случается довольно часто. Пространственная избыточность устраняется путем внутрикадрового кодирования. Один из методов внутрикадрового кодирования — дискретное косинусоидальное преобразование (Discrete Cosine Transform, DCT) — описан в следующем разделе.

- *Спектральная избыточность* (spectral redundancy). Спектральная информация состоит из световой интенсивности и цвета. Спектраль-

ная избыточность в кадрах кинофильма появляется из-за того, что яркие пикселы обычно ярки во всех цветах, а не в одном или двух. Кроме того, глаз человека более чувствителен к яркости, чем к цвету; то есть глаз скорее замечает необычную яркость пиксела, чем правильный цвет.

- *Временная избыточность* (temporal redundancy). В кино кадры сменяются со скоростью примерно 30 раз в секунду. Даже в боевике кадры одной сцены, сменяющие друг друга при такой частоте, отличаются лишь некоторыми деталями. Один из методов удаления временной избыточности — кодирование вектора перемещения (Motion Vector Encoding). Идея в том, чтобы поделить кадр на несколько блоков. Фактически, алгоритм кодирования сообщает: «Все, как и в предыдущем кадре, за исключением того, что в этом блоке произошло движение». (Например, моргнул глаз.)

Обычного кадра, содержащего полное изображение, недостаточно для использования временной избыточности. Чтобы работать с различными типами избыточности используются три вида кадров. Первый — обычный кадр, содержащий полное изображение. Это *I-кадр* (I-frame, intracode frame), названный так, дабы подчеркнуть то, что он независим от любого другого кадра. Второй тип — *P-кадр* (P-frame), или *предсказуемый кадр* (predicted frame) — создается на основе предыдущего I- или P-кадра, вектора перемещения и изображения ошибки, которое корректирует ошибки в векторе перемещения. Последний тип — *B-кадр* (B-frame), или *двунаправленный предсказуемый кадр* (bi-directional predicted frame) — состоит из вектора прямого перемещения, вектора обратного перемещения и ошибки. Эти кадры используют, когда в P-кадре появляется новый объект. (Типов векторов перемещения множество, но их описание выходит за рамки этой книги.)

Дискретное косинусоидальное преобразование

Для описания и передачи данных алгоритм сжатия может использовать уравнение вместо многократной выборки фактических значений. Предположим, Вы хотите нарисовать прямую линию. Элементарная алгебра учит нас, что прямую задает уравнение $y = mx + C$. Это означает, что линию можно «сжать» до всего лишь двух значений — m и C.

Разумеется, компрессия вовсе не так проста. Сложность составляет динамическое построение уравнения, описывающего вызвавшие интерес данные. Это уравнение обычно значительно сложнее, чем уравнение прямой линии. Одна из методик сжатия видеоданных — *дискретное косинусоидальное преобразование* (discrete cosine transform, DCT).

Здесь данные представлены в виде ряда косинусоид. Сохраняются лишь некоторые из них (соответствующие основным частотам), но при этом происходит потеря части данных.

DCT используется в стандартах MPEG (Moving Picture Experts Group), описанных в следующем разделе. DCT относится к методам внутрикадрового кодирования, поскольку в конкретный момент времени применяется к одному кадру.

Стандарты видео

Для кодирования видеоданных и видеоконференций существуют различные стандарты. В этом разделе кратко описаны некоторые наиболее распространенные из них.

Группа экспертов по движущимся изображениям

MPEG (Moving Pictures Experts Group) — Группа экспертов по движущимся изображениям — неофициальное название промышленной отрасли, разрабатывающей стандарты для мультимедиа. Официальное ее обозначение — ISO/IEC JTC1 SC29 WG11, что расшифровывается как «Международная организация по стандартизации/Международная комиссия по электротехнике, Совместный технический комитет 1, Подкомиссия 29, Рабочая группа 11». На собраниях MPEG, проводимых примерно четыре раза в год, планируется дальнейшая деятельность и принимаются резолюции.

До настоящего времени MPEG одобрила только три стандарта — MPEG-1, MPEG-2 и MPEG-3. Ведется работа над MPEG-4 и MPEG-7.

MPEG-1

MPEG-1, прежде всего, используется для сжатия видеоизображения на CD-ROM и подробно описан в стандарте ISO/IEC 11172-2.

MPEG-1 основан на принципах пространственной, спектральной и временной избыточности — о них говорилось в предыдущем разделе. Каждый кадр делится на макроблоки размером 16 на 16 пикселов. Несжатые данные каждого макроблока состоят из 16 блоков: 4 блока представляют данные зеленого цвета, 4 — красного, 4 — синего и еще 4 — цветность (или яркость) каждого пиксела. MPEG-1 сокращает число блоков до четырех: 2 представляют цветность и по 1 отведено для красного и синего цветов. Для сжатия MPEG-1 применяет алгоритм DCT, а также векторы перемещения и три типа кадров — I-, P-, и B- кадры, описанные ранее.

MPEG-1 требует больше вычислительных ресурсов для компрессии, чем для декомпрессии. Коэффициент сжатия MPEG-1 обычно не превосходит 50:1, хотя теоретически возможно соотношение 200:1. Забавно, что MPEG-1 завоевывает популярность только сейчас, после того как комитет MPEG принял стандарты MPEG-2 и MPEG-3 и активно разрабатывает новые стандарты. Отчасти причина в том, что MPEG-2 не стал так популярен, как ожидалось. Другая причина, вероятно, такова: MPEG-1 немного опередил свое время — вычислительная мощь компьютеров лишь недавно догнала его требования.

MPEG-2

MPEG-2 — это дополненная версия MEPG-1, пригодная для работы в области цифрового телевидения. Отличительные возможности MPEG-2 — поддержка кадров с чередованием строк (один кадр описывает половину изображения), поддержка уплотнения видео потоков, а также наличие нескольких алгоритмов сжатия сигналов яркости и цветности.

Пока реализаций MPEG-2 немного, но положение может измениться с наращиванием вычислительной мощности процессоров, например с распространением технологии Intel MMX.

MPEG-3

MPEG-3 предназначался для расширения возможностей MPEG-2 от стандарта сжатия для цифрового телевидения до стандарта, пригодного для телевидения высокой четкости (HDTV). Впоследствии стало очевидно, что все функции MPEG-3 можно было реализовать и в MPEG-2. Таким образом, MPEG-3 остался лишь вехой в истории технологий мультимедиа.

MPEG-4

Продолжается работа над MPEG-4, который может стать международным стандартом к концу 1998 года. В настоящее время анонсированные возможности MPEG-4 включают поддержку двунаправленного видеопотока, линий связи с низкой пропускной способностью и интерактивного взаимодействия с пользователем, то есть возможность выбирать одни фрагменты видеоданных, а другие пропускать.

MPEG-7

Не существует внятного объяснения, что же случилось с MPEG-5 и MPEG-6, но работа продолжается уже над MPEG-7; одна из задач — создание стандартов для мультимедийной поисковой машины.

Международный союз электросвязи

Международный союз электросвязи (International Telecommunications Union, ITU) помимо прочего определяет стандарты для звуковой и видеосвязи, а также для телеконференций.

ITU H.320

В 1990 году ITU принял стандарт H.320, состоящий из ряда спецификаций, определяющих стандарты для видеоконференций, включая групповые конференции (когда участвуют более двух абонентов). В качестве основной цифровой системы передачи H.320 рассматривает ISDN. (Описание ISDN см. в главе 3.)

ITU H.323

Стандарт ITU H.323 представляет собой целый комплекс рекомендаций, охватывающих несколько областей, включая кодеки и управление вызовами. H.323 — не что иное, как расширение H.320, допускающее работу через TCP/IP, Ethernet и аналоговые телефонные линии, помимо ISDN. Таким образом, H.323 позволяет проводить недорогие телеконференции. Примеры коммерческих программ, реализующих H.323, — Microsoft NetMeeting и CU-SeeMe от White Pine Software (см. раздел «Видео в Интернете»).

Для одного сеанса конференц-связи H.323 использует два TCP-соединения и одно UDP. Реально за время сеанса используется до пяти портов UDP. Настроить межсетевой экран так, чтобы он позволял выполнять подобные соединения, — непростая задача, иногда даже требуется помощь системного администратора.

ITU H.261

ITU H.261 — стандарт для приложений, обеспечивающих видеоконференции. Для таких сеансов характерно лишь незначительное перемещение объектов (в сравнении с кинофильмами), что и использует стандарт H.261, ограничивая длину векторов перемещения 15 пикселами. В отличие от MPEG-стандартов, H.261 не использует B-кадры. Нельзя назвать H.261 наиболее эффективным для видеоконференций. Компания PictureTel, оккупировавшая большую часть рынка связи для видеоконференций, имеет патентованный алгоритм, превосходящий H.261. Причем этот алгоритм применяется, только когда все участники конференции пользуются оборудованием, совместимым с PictureTel; иначе применяется H.261.

ITU H.260

Стандарт ITU H.260 предназначен для совместного использования документов и проведения групповых конференций. В нем описаны стандарты для реализации телеконференций, совместного доступа к «доскам» и передачи файлов. Работа над ITU H.260 еще продолжается, ожидается появление очередных протоколов для телеконференций.

Видео в Интернете

Ниже описаны различные программы, обеспечивающие использование видео в Интернете. Одна из серьезных проблем подобных программ — недостаточная совместимость. Бывает, что версия программы для IBM-совместимого компьютера не работает с аналогом для Macintosh. А если программы разные, то их совместимость еще хуже.

CU-SeeMe

CU-SeeMe — это программа для видеоконференций и видеотелефонии через Интернет, разработанная в Корнелльском университете (Cornell University). Она бесплатна как для IBM PC, так и для Macintosh. Компания White Pine Software продает расширенную версию CU-SeeMe. Наиболее значимое дополнение в этой версии — электронная «доска» для телеконференций.

Для передачи данных CU-SeeMe использует протокол UDP, а для устранения временной избыточности — патентованный метод. (Подробнее о временной избыточности см. раздел «Основы сжатия видео» ранее в этой главе.) Этот метод чем-то напоминает стандарт MPEG-1. Кадр делится на макроблоки размером 8 на 8 пикселов, которые сравниваются с соответствующими блоками предыдущего кадра. Передаются только значительно измененные блоки. Преимущество — необходимая степень сжатия достигается при гораздо меньших вычислительных затратах по сравнению с методами кодирования вектора перемещения. Недостаток же в том, что, если при передаче некоторые блоки потеряются, то изображение на экране превратится в беспорядочную мешанину старых и обновленных блоков.

В CU-SeeMe реализован механизм управления потоком данных. Получатель периодически передает информацию об утерянных пакетах, а отправитель соответственно корректирует объем отправляемых данных.

CU-SeeMe реализует групповые конференции посредством *рефлекторов* — серверов, которые представляют собой центральный ресурс для всех клиентов, желающих участвовать в групповой конференции. Из-

начально этот сервер функционирует как служба каталогов, выводя список доступных конференций и подконференций. Это позволяет клиентам обнаружить то, что представляет для них интерес. Как только сеанс установлен, рефлектор начинает работать на экономию пропускной способности сети. Обычно если в групповой конференции участвуют n клиентов, каждый клиент должен разослать $n-1$ копию звуковых и видеоданных. Причем некоторым, вероятно, придется делать это на скорости 28 кбит/с. Однако, при наличии рефлекторов, каждый клиент отправляет только одну копию данных, а уже рефлектор рассылает $n-1$ копию остальным клиентам. К тому же рефлектор располагает каналами с большей пропускной способностью, чем у клиента. Недостаток в том, что рефлектор должен разослать $n \times (n-1)$ копий данных. Для UNIX существует бесплатная версия ПО рефлектора.

В Интернете работают независимые рефлекторы, которые можно «брать взаймы», если владелец не возражает. Компания White Pine использует Four11 (*http://www.four11.com*) как сервер службы каталогов, дабы клиенты, желающие провести конференцию, могли найти друг друга.

MeetingPoint

Компания White Pine продает коммерческую версию рефлектора для Microsoft Windows NT под названием MeetingPoint. Он действует как сервер службы каталогов, позволяя клиентам находить и инициировать конференции, к которым они хотят присоединиться. MeetingPoint поддерживает групповое вещание протокола IP, что сокращает сетевой трафик. Также он поддерживает протокол H.323. Таким образом, MeetingPoint может выполнять роль сервера для любого H.323-клиента, в том числе для браузеров от Microsoft и Netscape (для версий, поддерживающих H.323).

VideoPhone и QuickTime

VideoPhone — продукт компании Connectix Corporation. Это ПО аналогично продукту QuickTime компании Apple и поставляется в комплекте с QuickCam — устройством ввода видеоданных (также от Connectix). VideoPhone и QuickTime, оба, поддерживают групповые конференции и групповое вещание протокола IP. Также они обеспечивают дополнительные возможности, например конференции с электронной «доской». Однако, версии для Macintosh (QuickTime) и для Windows (VideoPhone) несовместимы. Компания Connectix обязалась обеспечить совместимость Macintosh- и Windows-версий.

Microsoft NetShow 3.0

Компания Microsoft в своих печатных изданиях представляет NetShow как платформу для доставки мультимедийных данных через сети. Это может быть звук, видео, озвученные неподвижные изображения, текст, URL и т. д. Программа NetShow состоит из нескольких компонентов. NetShow-проигрыватель — клиентский компонент для воспроизведения мультимедиа-файлов. NetShow-сервер выполняется на платформе Windows NT и направляет клиентам потоки данных, содержащие звук, видео и др. Существует обширный набор служб администрирования сервера, а также ряд инструментальных средств для создания мультимедиа-файлов. Также компания Microsoft поставляет ряд SDK для управления NetShow-серверами, кодирования и редактирования данных.

Наиболее важное свойство NetShow — поддержка разнообразных кодеков. Это позволяет использовать NetShow в различных ситуациях, а также обеспечивает совместимость с широким кругом приложений. В табл. 15-3 перечислены кодеки, поддерживаемые программой NetShow.

Таблица 15-3. *Кодеки, поддерживаемые Microsoft NetShow 3.0*

Кодек	Звук/видео	Комментарий
Lernout & Hauspie CELP	Звук	Сжимает звук до фиксированной скорости битового потока 4,8 кбит/с; применяется в аудиоприложениях для звука низкого качества
Microsoft G.723.1	Звук	Универсальный звуковой кодек
Voxware MetaSound	Звук	Универсальный звуковой кодек; сжимает до скорости 8—28 кбит/с
Fraunhofer Institut Integrierte	Звук	Кодек для звука CD-качества Schaltungen MPEG-3
Voxware MetaVoice	Звук	Сжимает звук до фиксированной скорости битового потока 2,4 кбит/с
Sipro Lab Telecom ACELP.net	Звук	Универсальный аудиокодек; сжимает для передачи на низких скоростях
MSN Audio	Звук	Универсальный аудиокодек; сжимает для передачи на низких скоростях
Microsoft MPEG-4	Видео	Сжимает видео для передачи на скоростях 28,8—300 кбит/с
VDOnet	Видео	Кодек для низких и средних скоростей (28.8—150 кбит/с)
Intel H.263	Видео	Кодек потокового видео, для линий с низкой пропускной способностью (порядка 28,8 кбит/с)
Duck TrueMotion RT	Видео	Кодек, требующий высокой скорости передачи данных, порядка 1-2 Мбит/с

NetShow работает данными, представленными в файлах усовершенствованного потокового формата ASF (Advanced Streaming Format). Это открытый (незапатентованный, основанный на стандартах) формат файлов, допускающий объединение и доставку в едином потоке различных типов данных, таких, как звук, видео, неподвижные изображения, текст и HTML-страницы. ASF-файлы можно размещать на разных серверах, включая HTTP-серверы или обычные файловые серверы. Поток ASF-данных передается посредством различных транспортных протоколов, например RTP (Real-time Transport Protocol), UDP, TCP и т. д.

Протоколы мультимедиа

Для передачи мультимедиа через Интернет создано множество протоколов. Все они находятся на различных стадиях утверждения. В следующих разделах подробно описан каждый из них.

MBone

Магистраль группового вещания (Multicast backbone, MBone) — сеть, которая существует внутри Интернета с начала 1992 года. MBone состоит из IP-адресов класса D и набора маршрутизаторов, поддерживающих групповое вещание протокола IP. Многие маршрутизаторы в MBone выполняют протоколы IP-маршрутизации, обеспечивающие групповое вещание. Некоторые из них — на самом деле обычные рабочие станции, выполняющие роль маршрутизаторов. Другие — коммерческие маршрутизаторы с модифицированным ПО. Когда IP-пакеты, адресованные группе получателей, необходимо переправить через те участки Интернета, которые не поддерживают групповое вещание протокола IP, то используется «туннелинг» (см. главу 3, раздел «Виртуальная частная сеть»). То есть IP-пакеты, адресованные группе получателей, вкладываются в обычные IP-пакеты, адресованные маршрутизатору на другом конце туннеля (рис. 15-3).

MBone применяют для тестирования новых протоколов маршрутизации, ПО группового IP-вещания, а также протоколов мультимедиа, например RTP. Временами в MBone транслируются аудио и видео. Программа трансляции таких передач распространяется через список рассылки IETF. Подписаться на него можно, направив запрос по адресу *ietfrequest@ietf.org*. Бесплатное ПО для приема трансляций из MBone, в частности звуковые приложения, доступно на ряде Web-серверов. Дополнительную информацию Вы найдете в MBone FAQ (Frequently Asked Questions) по адресу *ftp.isi.edu/mbone/faq.txt*.

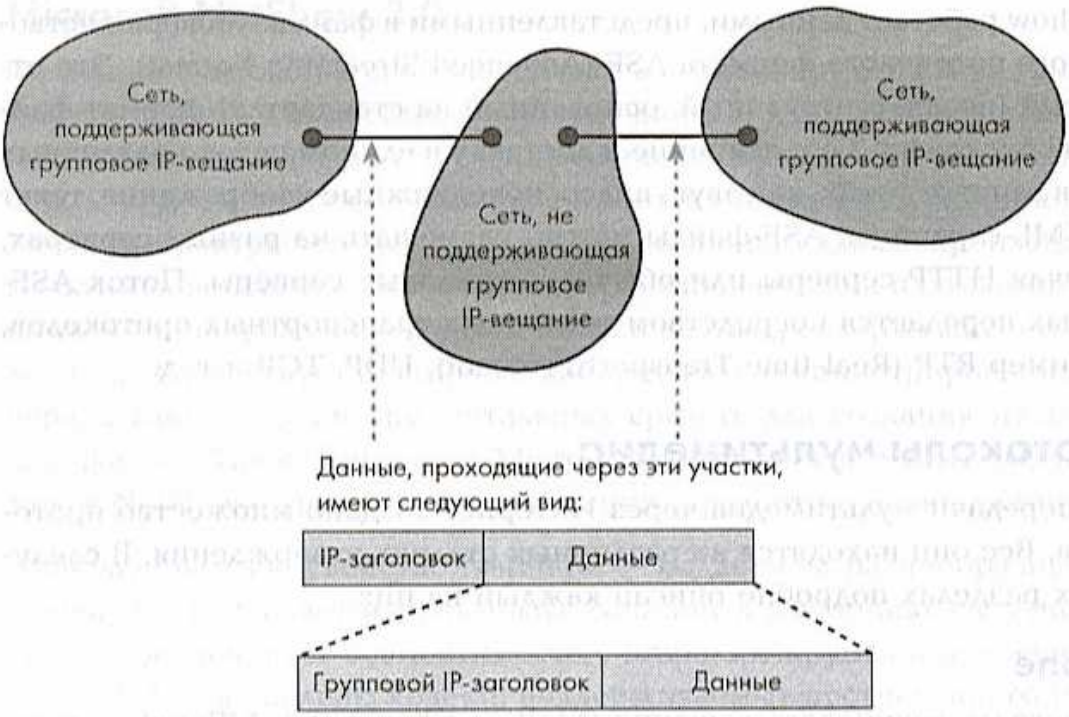


Рис. 15-3. *Туннелинг IP-пакетов группового вещания*

RTP

Протокол RTP (Real-time Transport Protocol) предоставляет транспортные услуги мультимедийным приложениям и позволяет им работать в сетях. RTP не гарантирует доставку и правильный порядок пакетов. Он просто присваивает каждому пакету номер, в результате мультимедийные приложения способны обнаружить утерю пакетов или нарушение порядка. Протокол RTP предназначен для работы в режимах передачи «один-одному» или «один-многим».

Обычно RTP передает данные, используя в качестве транспортного механизма протокол UDP, тем не менее он не зависит от транспортных протоколов и может применить другие, например TCP или ATM. Предполагается, что каждый RTP-пакет вложен в один элемент данных нижележащего транспортного протокола, например в один UDP-пакет. Когда RTP применяется совместно с транспортным механизмом без сообщений, например TCP, то используется псевдозаголовок, состоящий из RTP-заголовка, расположенного за полем длины.

Когда приложение начинает RTP-сеанс, оно использует два порта: один для протокола RTP, а другой для RTCP (Real-time Control Protocol), описанного ниже. При передаче мультимедийных данных задействованы два сеанса — для звука и для видео. Далее описаны некоторые поля RTP-заголовка.

- **Тип полезной нагрузки** (Payload Type) — указывает тип данных, которые несет RTP-пакет. Это позволяет приложениям применить правильный кодек для воспроизведения содержимого пакета. Приложения могут динамически (даже посреди RTP-сеанса) изменять тип данных исходя из текущего состояния сети. Один из допустимых типов описан в RFC 1889. В RFC 1890 перечислены значения поля **Тип полезной нагрузки** и их смысл. Например, число 3 обозначает данные формата GSM, число 26 — формата JPEG (Joint Photographic Experts Group), число 31 — формата H.261.
- **Порядковый номер** (Sequence Number) — 16-разрядное поле, позволяет приложениям выявлять утерю или нарушение последовательности пакетов.
- **Метка времени** (Timestamp) — 32-разрядное поле для синхронизации звуковых и видеопотоков. В этом поле устанавливается произвольное начальное значение, которое постепенно увеличивается. Два и более пакета могут иметь одинаковое значение поля **Метка времени**, например два звуковых пакета, соответствующие одному видеокадру.

Протокол RTP — не надежный транспортный механизм доставки пакетов в заданной последовательности. Также RTP не обеспечивает управление потоком данных и контроль перегрузки канала — это функции протокола RTCP.

Текущая версия RTP — вторая. Первая была экспериментальной и несовместима со второй. Протокол RTP подробно описан в RFC 1889 и 1890.

RTCP

Протокол RTCP (Real-Time Control Protocol) работает совместно с RTP, обеспечивая управление потоком данных и контроль перегрузки канала. Участники RTP-сеанса периодически посылают RTCP-пакеты, которые содержат статистические данные: количество отправленных пакетов, число утерянных и т. д. Отправитель мультимедиа-данных может использовать эту информацию для динамической корректировки скорости передачи и даже изменения типа полезной нагрузки. При применении RTCP получатель вправе уведомить отправителя о необходимости изменения скорости передачи, если, например, один из потоков поступает слишком быстро по сравнению с другим.

Также RTCP-пакеты несут идентификатор — *каноническое имя* (canonical name, CNAME). Он используется вкупе с транспортным адресом и но-

мером порта (например, IP-адрес и UDP-порт), чтобы уникально идентифицировать поток информации.

RTSP

Находящиеся в стадии утверждения протокол RTSP (Real-Time Streaming Protocol) — результат совместных усилий компаний RealNetworks и Netscape Communications Corporation. В нем оговаривается, как эффективно осуществлять доставку мультимедиа-данных по IP-сетям для приложений типа «один-многим». Архитектурно RTSP расположен над RTP и RTCP. Подобно тому как HTTP осуществляет передачу HTML, RTSP транспортирует мультимедиа-данные. Различие в том, что HTTP-клиент посылает запросы, а сервер обслуживает их, тогда как при применении RTSP запросы формирует как сервер, так и клиент; то есть RTSP — двунаправленный протокол. RTSP реализует передачу данных с помощью TCP или RTP. Сам же RTP способен использовать TCP или UDP.

Протокол RTSP содержит методы установки соединения, управления соединением и запроса объектов. Например, сообщение SET_TRANSPORT позволяет выбрать адреса портов, а также групповой IP-адрес (при групповом вещании). Сообщение PLAY_RANGE запрашивает у сервера передачу определенного количества мультимедиа-данных (в миллисекундах). Клиент может передать сообщение STOP, аналогичное нажатию кнопки «Пауза» на видеомагнитофоне. Сообщение SEND_REPORT иллюстрирует двунаправленность протокола RTSP. Его посылает сервер клиенту, чтобы получить отчет о качестве приема данных.

Протокол представлен на рассмотрение IETF как черновой стандарт. RTSP предназначен для осуществления взаимодействия клиентов и серверов от разных производителей. RTSP содержит механизмы управления, отсутствующие в RTP. Например, он позволяет выбирать способ доставки — UDP, TCP и т. д. Ожидается, что RTSP поможет постепенно перейти к групповому IP-вещанию. Работа над RTSP еще продолжается, и сейчас рассматриваются механизмы установления сеанса и управления им.

RSVP

RSVP (Resource Reservation Setup Protocol) — это сигнальный протокол, который используется в Интернете для резервирования ресурсов сети и маршрутизаторов. RSVP инициализируется получателями, а не отправителями и хорошо масштабируем в сетях со множеством узлов. RSVP определяет путь от отправителя до получателя, основываясь на протоколах маршрутизации (групповой или одиночной). RSVP-пакеты пере-

сылаются по этому пути от получателя к отправителю. Маршрутизаторы сначала объединяют запросы от нескольких получателей, а затем передают запрос на выделение ресурсов «против течения» — отправителю. Маршрутизаторы применяют «гибкий режим», то есть получатели должны периодически обновлять запрос на выделение ресурсов.

Архитектурно RSVP работает поверх IP, однако идейно он ближе к протоколам ICMP (Internet Control Message Protocol) и IGMP (Internet Group Membership Protocol), чем к транспортным. RSVP-сообщения инициируются хостами от имени приложений. Каждый узел, получивший RSVP-сообщение, выполняет два действия. Первое направлено на оценку RSVP-сообщения и решение, можно ли выделить запрошенный ресурс. При оценке проверяется наличие ресурсов (так называемая «проверка доступности»), а также наличие у пользователя соответствующих прав (так называемая «проверка стратегии»). Если запрос допустим, то узел выполняет второе действие — обновление информации о состоянии. Далее запрос передается следующему узлу по обратному пути — от отправителя получателю. Состояние каждого узла считается временным — то есть, запрашивающий узел должен периодически обновлять запрос. В противном случае состояние считается просроченным (и выделенный ресурс освобождается). Сообщения-обновления могут проходить по различным путям. Таким образом, если маршрут изменится, то состояния узлов старого маршрута считаются просроченными, и новые ресурсы будут выделяться по новому маршруту.

Ссылки

http://www.digiphone.com (DigiPhone)

http://www.vocaltec.com (VocalTec Inc., производитель Internet Phone)

http://www.itelco.com (NetSpeak, производитель WebPhone)

http://www.quarterdeck.com (Quarterdeck Corp., производитель WebTalk)

http://www.imtc.org (International Multimedia Teleconferencing Consortium)

http://www.connectix.com (Connectix, производитель VideoPhone)

http://www.wpine.com (White Pine, производитель Enhanced CU-SeeMe)

http://www.apple.com (Apple)

http://www.cornell.edu (Cornell University)

http://www.four11.com (Internet White Pages от Yahoo!, сервер службы каталогов для CU-SeeMe от White Pine)

http://www.ietf.cnri.reston.va.us/html.charters/rsvp-charter.html (RSVP)

http://huntleyl.prognet.com/prognet/rt/ (RTSP)

http://www.mediadesign.co.at/newmedia/more/mbone-faq.html

http://www.mbone.com

http://www.mpeg.org (Ресурсы MPEG — НЕ официальная страница MPEG)

http://drogo.cselt.stet.it/mpeg/ (Официальная страница MPEG)

http://www.itu.ch/

http://web.ansi.org

RFC 1890, «RTP Profile for Audio and Video Conferences with Minimal Control»

RFC 1889, «RTP: A Transport Protocol for Real-Time Applications»

RFC 1112, «Host Extensions for IP Multicasting»

ГЛАВА 16

Язык VRML

По мере роста популярности Web-протоколов появилась возможность замены двумерного пользовательского интерфейса, применяющего для описания текста и изображений язык HTML, на трехмерный, основанный на VRML (Virtual Reality Modeling Language) — языке моделирования виртуальной реальности.

VRML — это особый формат файлов для описания интерактивных трехмерных объектов и целых виртуальных «миров» в Интернете. Он использует существующие Web-протоколы для работы в инфраструктуре Web. Пользователь просматривает VRML-мир с помощью VRML-браузеров, или обозревателей, — некоторые из них представляют собой самостоятельные программы, другие же — расширения существующих Web-браузеров.

Обычно в VRML-миры попадают через HTML-страницы, указывая соответствующий URL и обращаясь по протоколу HTTP к обычным Web-серверам. (Иногда VRML-миры находятся на локальном диске.) В VRML-мирах зачастую используются стандартные форматы представления мультимедийных данных, например JPEG, MPEG, PNG (Portable Network Graphics) и MIDI (Musical Instrument Digital Interface), которые «вклеиваются» в VRML при помощи MIME. Действия с этими форматами описывают программно — как в самом VRML-мире при помощи сценариев на Java и JavaScript, так и внешне средствами отдельного Java-приложения.

Язык VRML предназначен и для дизайнеров (разработчиков содержания), и для программистов. VRML-миры можно создавать инструментальными средствами, не требующими собственно программирования. Однако возможности программирования позволяют сделать их более интерактивными, чем большинство CAD-объектов и другие трехмерные объекты.

Версии VRML

Язык VRML версии 1, разработанный в 1994 году, предназначен для создания статических VRML-миров с весьма ограниченной интерактивностью. В этой версии использовались файлы формата ASCII (с расширением .wrl), своим происхождением обязанные программе компьютерного дизайна OpenInventor компании Silicon Graphics. Эти wrl-файлы могли описываться в формате Unicode UTF-8. VRML версии 1 поддерживает лишь статические трехмерные объекты и гиперссылки на объекты других типов.

Язык VRML версии 2 (разработанный в 1996 году) допускает более сложное поведение объектов. Он поддерживает динамические объекты и позволяет встраивать сценарии на Java и JavaScript.

Спецификация VRML 97, будучи DIS-стандартом (Draft International Standard — проект международного стандарта), основана на черновом варианте VRML версии 2, предложенном в августе 1996 года. Это — результат сотрудничества ISO и IEC (International Electrotechnical Commission). Разработку выполняла Подкомиссия 24 (SC24) из Совместного технического комитета 1 (JTC 1), которая специализируется на компьютерной графике и обработке изображений. Спецификация VRML 97 от ISO/IEC JTC1/SC24 описана в документе ISO/IEC DIS 14772-1.

Основные понятия

VRML-мир — это сцена, представленная в виде иерархического направленного графа без циклов, содержащего один или более объектов. Сцена состоит из некоторого числа *узлов* (nodes), которые описывают целый ряд характеристик — форму, цвет, текстуры, освещение и т. д. Узел имеет одно или несколько полей, содержащих информацию о параметрах.

В VRML-мирах есть генераторы и приемники событий. Большинство генераторов событий представляют собой *сенсорные узлы* (sensor nodes), которые обеспечивают взаимодействие внутри VRML-мира, описывая время, столкновения, близость, касание, видимость и т. д. Язык VRML поддерживает событийную архитектуру, чтобы узлы могли взаимодействовать, посылая и получая события.

Другой тип узла, способный генерировать события, — *узел сценария* (script node). Привязки, задаваемые таким узлом, используются в программном коде (например, на языках Java и JavaScript) для вызова событий, воздействующих на VRML-мир.

Узлы VRML-мира могут храниться в одном или в нескольких файлах, на которые указывают URL. Узлы VRML описаны в табл. 16-1.

Таблица 16-1. *Узлы VRML*

Узел	Описание
Anchor	Вызывает выбранный пользователем URL
Appearance	Определяет визуальные характеристики объекта
AudioClip	Задает звуковой объект
Background	Определяет декорации (земля, небо и т. д.)
Billboard	Один из видов узла Transform, где точка, с которой ведется обзор, вращается вместе с пользователем
Box	Определяет объект-прямоугольник
Collision	Параметры обработки столкновений для дочерних узлов
Color	Цвета (RGB), используемые в полях узла
ColorInterpolator	Формирует цвет на основе смены событий
Cone	Определяет объект-конус
Coordinate	Задает трехмерную систему координат
CoordinateInterpolator	Формирует трехмерную систему координат на основе смены событий
Cylinder	Определяет цилиндрический объект
CylinderSensor	Преобразует движение мыши во вращение цилиндра
DirectionalLight	Задает источник направленного света
ElevationGrid	Определяет прямоугольную сетку
Extrusion	«Выдавливает» плоские фигуры вдоль трехмерных «хребтов»
Fog	Имитирует дымку
FontStyle	Задает информацию о шрифте в текстовых узлах
Group	Объединяет несколько дочерних узлов в логическую группу
ImageTexture	Определяет текстурную карту изображения
IndexedFaceSet	Представляет трехмерную фигуру в виде множества многоугольников
IndexedLineSet	Представляет трехмерную фигуру в виде множества линий
Inline	Узел группировки, который считывает из URL данные дочернего объекта
LOD	Задает уровень детализации объекта при разной удаленности
Material	Определяет свойства материала поверхности геометрических узлов
MovieTexture	Определяет данные синхронизации для видеофайла и текстуры

(см. след. стр.)

Таблица 16-1. *Узлы VRML* (продолжение)

Узел	Описание
NavigationInfo	Описывает характеристики пользовательской модели визуализации
Normal	Определяет набор нормальных векторов к трехмерным поверхностям для использования с геометрическими узлами
NormalInterpolator	Интерполирует список наборов нормальных векторов
OrientationInterpolator	Интерполирует набор параметров вращения
PixelTexture	Задает плоскую текстуру
PlaneSensor	Преобразует движение указателя мыши в перемещение объекта на плоскости
PointLight	Задает положение источника света в трехмерном пространстве
PointSet	Задает набор трехмерных точек
PositionInterpolator	Линейно интерполирует набор трехмерных векторов
ProximitySensor	Сенсор для перемещения объекта внутри области
ScalarInterpolator	Линейно интерполирует набор значений с плавающей точкой
Script	Узел интерфейса языка сценария
Shape	Определяет внешний вид и геометрию объекта
Sound	Определяет пространственное представление звука в сцене
Sphere	Задает сферический объект
SphereSensor	Преобразует движение мыши во вращение сферы
SpotLight	Задает источник света из определенной точки
Switch	Узел условной группировки, который ссылается на другой узел
Text	Задает текст
TextureCoordinate	Определяет набор координат двумерной текстуры
TextureTransform	Задает двумерное преобразование для координат текстуры
TimeSensor	Генерирует события через заданные промежутки времени
TouchSensor	Отслеживает состояние мыши, когда она указывает на объект
Transform	Узел группировки, задающий координатную систему для своих потомков
Viewpoint	Задает точку, с которой пользователь наблюдает за сценой
VisibilitySensor	Выявляет изменения видимости объекта
WorldInfo	Задает общую информацию о VRML-мире

VRML-сценарии на Java и JavaScript

Интерфейс сценариев браузера (Browser Scripting Interface), благодаря узлам типа Script, позволяет запрограммировать поведение VRML-мира. Сценарии имеют доступ ко многим узлам, полям и событиям VRML-файла. Кроме того, Интерфейс сценариев браузера предоставляет интерфейсы для взаимодействия с узлами, полями и событиями VRML-сцены, а также для взаимодействия с самим VRML-браузером. Спецификация VRML версии 2 определяет интерфейсы для языков Java и JavaScript.

Внешний программный интерфейс VRML

VRML версии 2 обеспечивает интерфейс для программ, внешних по отношению к VRML-браузеру, что позволяет им взаимодействовать с VRML-браузером и VRML-миром. Внешний программный интерфейс VRML (External Authoring Interface, EAI) обеспечивает программный доступ к VRML-браузеру. EAI функционирует подобно внутреннему интерфейсу VRML-сценариев, обеспечивая доступ к узлам, событиям, и Интерфейсу сценариев браузера.

Существующая версия EAI не окончательная. Интерфейс, предложенный рабочей группой EAI, предназначен для языка Java и реализован в пакете vrml.external.

Перспективные разработки

Целый ряд рабочих групп занимается расширением возможностей языка VRML. Сегодня можно говорить о следующих достижениях:

- рабочая группа VRML object-oriented extensions (объектно-ориентированные расширения) разрабатывает объектно-ориентированные расширения для VRML;
- рабочая группа VRML database (базы данных) работает над интеграцией баз данных с VRML;
- рабочая группа VRML compressed binary format (сжатый двоичный формат) создает метод кодирования VRML-файлов, который позволит сократить их размер и увеличить скорость передачи по сети, а также разрабатывает упрощенный формат файлов для обеспечения эффективного анализа;
- рабочая группа VRML color fidelity (точность воспроизведения цветов) разрабатывает методы обеспечения качественной цветопередачи для VRML-браузеров;

- рабочая группа VRML keyboard input (клавиатурный ввод) работает над модификацией VRML, чтобы расширить возможности применения клавиатуры в VRML-мирах, которые сейчас в основном управляются мышью;
- рабочая группа VRML widgets создает специальные графические элементы, из которых как из кубиков можно создать пользовательский интерфейс;
- рабочая группа VRML universal media libraries (универсальные библиотеки) разрабатывает набор элементов из изображений, звуков и некоторых VRML-объектов для локальных VRML-браузеров, что позволит уменьшить сетевой трафик и более насытить мультимедийное наполнение миров;
- рабочая группа VRML humanoid animation (человекоподобная анимация) создает методы воспроизведения человекоподобных форм в VRML-мирах;
- рабочая группа VRML biota (флора и фауна) разрабатывает инструменты для представления живых систем в VRML-мирах;
- рабочая группа VRML metaforms создает методологию описания формальных грамматик, задающих «законы природы» в VRML-мирах, что позволит изображать структуру и развитие компьютерных форм жизни;
- рабочая группа VRML living worlds (живые миры) конструирует набор интерфейсов, придающих VRML-мирам многопользовательские возможности;
- рабочая группа VRML distributed interactive simulation (распределенное интерактивное моделирование) работает над интеграцией VRML, Java и технологии DIS (Distributed Interactive Simulation — Распределенное интерактивное моделирование) для создания крупномасштабных виртуальных сред. Технология DIS известна, вероятно, лучше остальных благодаря применению в военных многопользовательских имитаторах.

Ссылки

http://www.vrml.org

http://www.bsi.org.uk/sc24

Приложения

Приложение А

Стандарты Интернета

В этом приложении перечислены стандарты, которым соответствуют документы RFC (Request for Comments), одобренные IAB (Internet Architecture Board). Каждому стандарту присвоен номер, который зачастую отличается от номера соответствующего RFC. Некоторые стандарты описываются несколькими RFC. Целесообразность такой схемы объясняется тем, что конкретный протокол может описываться и развиваться путем публикации новых RFC, тогда как номер стандарта остается неизменным. Например, соответствующий Стандарту 1 «Internet Official Protocol Standards» («Официальные стандарты протоколов Интернета») RFC изменялся более десяти раз. В таблице перечислены номера стандартов Интернета, их названия и соответствующие RFC.

Стандарты Интернета

Номер стандарта	Название	Номера RFC
0001	Internet Official Protocol Standards	2200
0002	Assigned Numbers	1700
0003	Host Requirements	1123, 1122
0004	Gateway Requirements	1009
0005	Internet Protocol (IP)	1112, 950, 922, 919, 792, 791
0006	User Datagram Protocol (UDP)	768
0007	Transmission Control Protocol (TCP)	793
0008	Telnet Protocol (TP)	855, 854
0009	File Transfer Protocol (FTP)	959
0010	Simple Mail Transfer Protocol (SMTP) Service Extensions	1869, 821
0011	Standard for the Format of Advanced Research Project Agency (ARPA) Internet Text Messages	822
0012	Network Time Protocol (NTP)	1119
0013	Domain Name System (DNS)	1035, 1034
0014	Mail Routing and the Domain System	974

(см. след. стр.)

Стандарты Интернета (продолжение)

Номер стандарта	Название	Номера RFC
0015	Simple Network Management Protocol (SNMP)	1157
0016	Structure of Management Information	1212, 1155
0017	Management Information Base	1213
0018	External Gateway Protocol (EGP)	904
0019	NetBIOS Service Protocols; NetBIOS Working Group	1002, 1001
0020	Echo Protocol	862
0021	Discard Protocol	863
0022	Character Generator Protocol	864
0023	Quote of the Day Protocol	865
0024	Active Users Protocol	866
0025	Daytime Protocol	867
0026	Time Server Protocol	868
0027	Binary Transmission Telnet Option	856
0028	Echo Telnet Option	857
0029	Suppress Go Ahead Telnet Option	858
0030	Status Telnet Option	859
0031	Timing Mark Telnet Option	860
0032	Extended Options List Telnet Option	861
0033	Trivial File Transfer Protocol	1350
0034	Routing Information Protocol	1058
0035	ISO Transport Service on Top of the TCP (Version3)	1006
0036	Transmission of IP and Address Resolution Protocol (ARP) over Fiber Distributed Data Interface (FDDI) Networks	1390
0037	An Ethernet Address Resolution Protocol	826
0038	A Reverse Address Resolution Protocol	903
0039	Interface Message Processor: Specifications for the Interconnection of a Host and an IMP (Revised)	Нет
0040	Host Access Protocol Specification	1221
0041	Standard for the Transmission of IP Datagrams over Ethernet Networks	894
0042	Standard for the Transmission of IP Datagrams over Experimental Ethernet Networks	895
0043	Standard for the Transmission of IP Datagrams over Institute of Electrical and Electronics Engineers (IEEE) 802 Networks	1042
0044	DCN Local-Network Protocols	891

Стандарты Интернета (продолжение)

Номер стандарта	Название	Номера RFC
0045	Internet Protocol on Network System's HYPER-channel: Protocol Specification	1044
0046	Transmitting IP Traffic over ARCNET Networks	1201
0047	Nonstandard for Transmission of IP Datagrams over Serial Lines: Serial Line Internet Protocol (SLIP)	1055
0048	Standard for the Transmission of IP Datagrams over NetBIOS Networks	1008
0049	Standard for the Transmission of 802.2 Packets over Internetwork Packet Exchange (IPX) Networks	1132
0050	Definitions of Managed Objects for the Ethernet-like Interface Types	1643
0051	Point-to-Point Protocol (PPP)	1662
0052	The Transmission of IP Datagrams over the Switched Multimegabit Data Services (SMDS) Service	1209
0053	Post Office Protocol (POP) Version 3	1939
0054	Open Shortest Path First (OSPF) Version 2	2328

Приложение Б

Черновые стандарты

Черновой стандарт — это очередная (после предложенного стандарта) степень «зрелости» документа RFC (см. приложение В), за которой следует внесение этого RFC в число официальных стандартов Интернета (см. приложение А).

RFC 2289, «A One-Time Password System»

RFC 2197, «SMTP Service Extension for Command Pipelining»

RFC 2178, «OSPF Version 2»

RFC 2132, «DHCP Options and BOOTP Vendor Extensions»

RFC 2131, «Dynamic Host Configuration Protocol»

RFC 2115, «Management Information Base for Frame Relay DTEs Using SMIv2»

RFC 2067, «IP over HIPPI»

RFC 2049, «Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples»

RFC 2047, «MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text»

RFC 2046, «Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types»

RFC 2045, «Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies»

RFC 1994, «PPP Challenge Handshake Authentication Protocol (CHAP)»

RFC 1990, «The PPP Multilink Protocol (MP)»

RFC 1989, «PPP Link Quality Monitoring»

RFC 1908, «Coexistence between Version 1 and Version 2 of the Internet-Standard Network Management Framework»

RFC 1907, «Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2)»

RFC 1906, «Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)»

RFC 1905, «Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)»

RFC 1904, «Conformance Statements for Version 2 of the Simple Network Management Protocol (SNMPv2)»

RFC 1903, «Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2)»

RFC 1902, «Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)»

RFC 1864, «The Content-MD5 Header Field»

RFC 1850, «OSPF Version 2 Management Information Base»

RFC 1832, «XDR: External Data Representation Standard»

RFC 1779, «A String Representation of Distinguished Names»

RFC 1778, «The String Representation of Standard Attribute Syntaxes»

RFC 1777, «Lightweight Directory Access Protocol»

RFC 1772, «Application of the Border Gateway Protocol in the Internet»

RFC 1771, «A Border Gateway Protocol 4 (BGP-4)»

RFC 1762, «The PPP DECnet Phase IV Control Protocol (DNCP)»

RFC 1757, «Remote Network Monitoring Management Information Base»

RFC 1748, «IEEE 802.5 MIB Using SMIv2»

RFC 1743, «IEEE 802.5 MIB Using SMIv2»

RFC 1725, «Post Office Protocol — Version 3»

RFC 1724, «RIP Version 2 MIB Extension»

RFC 1723, «RIP Version 2 — Carrying Additional Information»

RFC 1722, «RIP Version 2 Protocol Applicability Statement»

RFC 1694, «Definitions of Managed Objects for SMDS Interfaces Using SMIv2»

RFC 1660, «Definitions of Managed Objects for Parallel-Printer-Like Hardware Devices Using SMIv2»

RFC 1659, «Definitions of Managed Objects for RS-232-Like Hardware Devices Using SMIv2»

RFC 1658, «Definitions of Managed Objects for Character Stream Devices Using SMIv2»

RFC 1657, «Definitions of Managed Objects for the Fourth Version of the Border Gateway Protocol (BGP-4) Using SMIv2»

RFC 1653, «SMTP Service Extension for Message Size Declaration»

RFC 1652, «SMTP Service Extension for 8bit-MIME transport»

RFC 1651, «SMTP Service Extensions»

RFC 1629, «Guidelines for OSI NSAP Allocation in the Internet»

RFC 1583, «OSPF Version 2»

RFC 1575, «An Echo Function for CLNP (ISO 8473)»

RFC 1559, «DECnet Phase IV MIB Extensions»

RFC 1549, «PPP in HDLC Framing»

RFC 1548, «The Point-to-Point Protocol (PPP)»

RFC 1542, «Clarifications and Extensions for the Bootstrap Protocol»

RFC 1534, «Interoperation Between DHCP and BOOTP»

RFC 1522, «MIME (Multipurpose Internet Mail Extensions) Part Two: Message Header Extensions for Non-ASCII Text»

RFC 1521, «MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies»

RFC 1516, «Definitions of Managed Objects for IEEE 802.3 Repeater Devices»

RFC 1497, «BOOTP Vendor Information Extensions»

RFC 1493, «Definitions of Managed Objects for Bridges»

RFC 1490, «Multiprotocol Interconnect over Frame Relay»

RFC 1460, «Post Office Protocol — Version 3»

RFC 1398, «Definitions of Managed Objects for the Ethernet-Like Interface Types»

RFC 1395, «BOOTP Vendor Information Extensions»

RFC 1356, «Multiprotocol Interconnect on X.25 and ISDN in the Packet Mode»

RFC 1305, «Network Time Protocol (Version 3) Specification, Implementation»

RFC 1288, «The Finger User Information Protocol»

RFC 1247, «OSPF Version 2»

RFC 1225, «Post Office Protocol: Version 3»

RFC 1196, «Finger User Information Protocol. December 1, 1990 (Format: TXT= 24799) (Obsoletes RFC1194, 742)»

RFC 1194, «Finger User Information Protocol. November 1, 1990 (Format: TXT= 24626 bytes) (Obsoletes RFC0742)»

RFC 1191, «Path MTU Discovery»

RFC 1188, «Proposed Standard for the Transmission of IP Datagrams over FDDI Networks»

RFC 1184, «Telnet Linemode Option»

RFC 1171, «Point-to-Point Protocol for the Transmission of Multiprotocol Datagrams over Point-to-Point Links»

RFC 1158, «Management Information Base for Network Management of TCP/IP-Based Internets: MIB-II»

RFC 954, «NICNAME/WHOIS»

Приложение B

Предложенные стандарты

Здесь перечислены документы RFC, предложенные к принятию в качестве стандартов Интернета (см. приложение A). Время покажет, станут ли эти RFC частью стандартов.

RFC 2327, «SDP: Session Description Protocol»

RFC 2320, «Definitions of Managed Objects for Classical IP and ARP over ATM Using SMIv2 (IPOA-MIB)»

RFC 2308, «Negative Caching of DNS Queries (DNS NCACHE)»

RFC 2305, «A Simple Mode of Facsimile Using Internet Mail»

RFC 2304, «Minimal FAX Address Format in Internet Mail»

RFC 2303, «Minimal PSTN Address Format in Internet Mail»

RFC 2302, «Tag Image File Format (TIFF)—Image/TIFF MIME Sub-Type Registration»

RFC 2301, «File Format for Internet Fax»

RFC 2298, «An Extensible Message Format for Message Disposition Notifications»

RFC 2294, «Representing the O/R Address Hierarchy in the X.500 Directory Information Tree»

RFC 2293, «Representing Tables and Subtrees in the X.500 Direc»

RFC 2290, «Mobile-IPv4 Configuration Option for PPP IPCP»

RFC 2287, «Definitions of System-Level Managed Objects for Applications»

RFC 2284, «PPP Extensible Authentication Protocol (EAP)»

RFC 2283, «Multiprotocol Extensions for BGP-4»

RFC 2280, «Routing Policy Specification Language (RPSL)»

RFC 2279, «UTF-8, a Transformation Format of ISO 10646»

RFC 2275, «View-Based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)»

RFC 2274, «User-Based Security Model (USM) for Version 3 of the Simple Network Management Protocol (SNMPv3)»

RFC 2273, «SNMPv3 Applications»

RFC 2272, «Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)»

RFC 2271, «An Architecture for Describing SNMP Management Frameworks»

RFC 2266, «Definitions of Managed Objects for IEEE 802.12 Repeater Devices»

RFC 2265, «View-Based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)»

RFC 2264, «User-Based Security Model (USM) for Version 3 of the Simple Network Management Protocol (SNMPv3)»

RFC 2263, «SNMPv3 Applications»

RFC 2262, «Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)»

RFC 2261, «An Architecture for Describing SNMP Management Frameworks»

RFC 2257, «Agent Extensibility (AgentX) Protocol Version 1»

RFC 2256, «A Summary of the X.500(96) User Schema for Use with LDAPv3»

RFC 2255, «The LDAP URL Format»

RFC 2254, «The String Representation of LDAP Search Filters»

RFC 2253, «Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names»

RFC 2252, «Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions»

RFC 2251, «Lightweight Directory Access Protocol (v3)»

RFC 2250, «RTP Payload Format for MPEG1/MPEG2 Video»

RFC 2249, «Mail Monitoring MIB»

RFC 2248, «Network Services Monitoring MIB»

RFC 2247, «Using Domains in LDAP/X.500 Distinguished Names»

RFC 2245, «Anonymous SASL Mechanism»

RFC 2244, «ACAP — Application Configuration Access Protocol»

RFC 2243, «OTP Extended Responses»

RFC 2242, «NetWare/IP Domain Name and Information»

RFC 2241, «DHCP Options for Novell Directory Services»

RFC 2239, «Definitions of Managed Objects for IEEE 802.3 Medium Attachment Units (MAUs) Using SMIv2»

RFC 2238, «Definitions of Managed Objects for HPR Using SMIv2»

RFC 2236, «Internet Group Management Protocol, Version 2»

RFC 2234, «Augmented BNF for Syntax Specifications: ABNF»

RFC 2233, «The Interfaces Group MIB Using SMIv2»

RFC 2232, «Definitions of Managed Objects for DLUR Using SMIv2»

RFC 2231, «MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations»

RFC 2228, «FTP Security Extensions»

RFC 2227, «Simple Hit-Metering and Usage-Limiting for HTTP»

RFC 2226, «IP Broadcast over ATM Networks»

RFC 2225, «Classical IP and ARP over ATM»

RFC 2222, «Simple Authentication and Security Layer (SASL)»

RFC 2221, «IMAP4 Login Referrals»

RFC 2218, «A Common Schema for the Internet White Pages Service»

RFC 2215, «General Characterization Parameters for Integrated Service Network Elements»

RFC 2214, «Integrated Services Management Information Base Guaranteed Service Extensions using SMIv2»

RFC 2213, «Integrated Services Management Information Base Using SMIv2»

RFC 2212, «Specification of Guaranteed Quality of Service»

RFC 2211, «Specification of the Controlled-Load Network Element Service»

RFC 2210, «The Use of RSVP with IETF Integrated Services»

RFC 2207, «RSVP Extensions for IPSEC Data Flows»

RFC 2206, «RSVP Management Information Base using SMIv2»

RFC 2205, «Resource ReSerVation Protocol (RSVP) — Version 1 Functional Specification»

RFC 2203, «RPCSEC_GSS Protocol Specification»

RFC 2198, «RTP Payload for Redundant Audio Data»

RFC 2195, «IMAP/POP AUTHorize Extension for Simple Challenge/Response»

RFC 2193, «IMAP4 Mailbox Referrals»

RFC 2192, «IMAP URL Scheme»

RFC 2190, «RTP Payload Format for H.263 Video Streams»

RFC 2184, «MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations»

RFC 2183, «Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field»

RFC 2181, «Clarifications to the DNS Specification»

RFC 2177, «IMAP4 IDLE command»

RFC 2165, «Service Location Protocol»

RFC 2164, «Use of an X.500/LDAP Directory to Support MIXER Address Mapping»

RFC 2163, «Using the Internet DNS to Distribute MIXER Conformant Global Address Mapping (MCGAM)»

RFC 2160, «Carrying PostScript in X.400 and MIME»

RFC 2159, «A MIME Body Part for FAX»

RFC 2158, «X.400 Image Body Parts»

RFC 2157, «Mapping Between X.400 and RFC-822/MIME Message Bodies»

RFC 2156, «MIXER (Mime Internet X.400 Enhanced Relay): Mapping Between X.400 and RFC 822/MIME»

RFC 2155, «Definitions of Managed Objects for APPN Using SMIv2»

RFC 2147, «TCP and UDP over IPv6 Jumbograms»

RFC 2142, «Mailbox Names for Common Services, Roles, and Functions»

RFC 2141, «URN Syntax»

RFC 2138, «Remote Authentication Dial-In User Service (RADIUS)»

RFC 2137, «Secure Domain Name System Dynamic Update»

RFC 2136, «Dynamic Updates in the Domain Name System (DNS UPDATE)»

RFC 2128, «Dial Control Management Information Base Using SMIv2»

RFC 2127, «ISDN Management Information Base Using SMIv2»

RFC 2126, «ISO Transport Service on Top of TCP (ITOT)»

RFC 2125, «The PPP Bandwidth Allocation Protocol (BAP)/The PPP Bandwidth Allocation Control Protocol (BACP)»

RFC 2122, «VEMMI URL Specification»

RFC 2113, «IP Router Alert Option»

RFC 2112, «The MIME Multipart/Related Content-Type»

RFC 2111, «Content-ID and Message-ID Uniform Resource Locators»

RFC 2110, «MIME E-mail Encapsulation of Aggregate Documents, Such as HTML (MHTML)»

RFC 2109, «HTTP State Management Mechanism»
RFC 2108, «Definitions of Managed Objects for IEEE 802.3 Repeater Devices Using SMIv2»
RFC 2097, «The PPP NetBIOS Frames Control Protocol (NBFCP)»
RFC 2096, «IP Forwarding Table MIB»
RFC 2095, «IMAP/POP AUTHorize Extension for Simple Challenge/Response»
RFC 2091, «Triggered Extensions to RIP to Support Demand Circuits»
RFC 2088, «IMAP4 Nonsynchronizing Literals»
RFC 2087, «IMAP4 QUOTA Extension»
RFC 2086, «IMAP4 ACL Extension»
RFC 2085, «HMAC-MD5 IP Authentication with Replay Prevention»
RFC 2082, «RIP-2 MD5 Authentication»
RFC 2080, «RIPng for IPv6»
RFC 2079, «Definition of an X.500 Attribute Type and an Object Class to Hold Uniform Resource Identifiers (URIs)»
RFC 2078, «Generic Security Service Application Program Interface, Version 2»
RFC 2077, «The Model Primary Content Type for Multipurpose Internet Mail Extensions»
RFC 2074, «Remote Network Monitoring MIB Protocol Identifiers»
RFC 2073, «An IPv6 Provider-Based Unicast Address Format»
RFC 2070, «Internationalization of the Hypertext Markup Language»
RFC 2069, «An Extension to HTTP: Digest Access Authentication»
RFC 2068, «Hypertext Transfer Protocol — HTTP/1.1»
RFC 2065, «Domain Name System Security Extensions»
RFC 2062, «Internet Message Access Protocol — Obsolete Syntax»
RFC 2060, «Internet Message Access Protocol — Version 4rev1»
RFC 2058, «Remote Authentication Dial-In User Service (RADIUS)»
RFC 2056, «Uniform Resource Locators for Z39.50»
RFC 2051, «Definitions of Managed Objects for APPC Using SMIv2»
RFC 2038, «RTP Payload Format for MPEG1/MPEG2 Video»
RFC 2037, «Entity MIB Using SMIv2»
RFC 2035, «RTP Payload Format for JPEG-Compressed Video»
RFC 2034, «SMTP Service Extension for Returning Enhanced Error Codes»

RFC 2032, «RTP Payload Format for H.261 Video Streams»

RFC 2029, «RTP Payload Format of Sun's CellB Video Encoding»

RFC 2025, «The Simple Public-Key GSS-API Mechanism (SPKM)»

RFC 2024, «Definitions of Managed Objects for Data Link Switching Using SMIv2»

RFC 2023, «IP Version 6 over PPP»

RFC 2022, «Support for Multicast over UNI 3.0/3.1 – based ATM Networks»

RFC 2021, «Remote Network Monitoring Management Information Base Version 2 Using SMIv2»

RFC 2020, «IEEE 802.12 Interface MIB»

RFC 2019, «Transmission of IPv6 Packets Over FDDI»

RFC 2018, «TCP Selective Acknowledgment Options»

RFC 2017, «Definition of the URL MIME External-Body AccessType»

RFC 2015, «MIME Security with Pretty Good Privacy (PGP)»

RFC 2013, «SNMPv2 Management Information Base for the User Datagram Protocol using SMIv2»

RFC 2012, «SNMPv2 Management Information Base for the Transmission Control Protocol Using SMIv2»

RFC 2011, «SNMPv2 Management Information Base for the Internet Protocol Using SMIv2»

RFC 2006, «The Definitions of Managed Objects for IP Mobility Support Using SMIv2»

RFC 2005, «Applicability Statement for IP Mobility Support»

RFC 2004, «Minimal Encapsulation Within IP»

RFC 2003, «IP Encapsulation Within IP»

RFC 2002, «IP Mobility Support»

RFC 2001, «TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms»

RFC 1997, «BGP Communities Attribute»

RFC 1996, «A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)»

RFC 1995, «Incremental Zone Transfer in DNS»

RFC 1985, «SMTP Service Extension for Remote Message Queue Starting»

RFC 1982, «Serial Number Arithmetic»

RFC 1981, «Path MTU Discovery for IP Version 6»

RFC 1973, «PPP in Frame Relay»

RFC 1972, «A Method for the Transmission of IPv6 Packets over Ethernet Networks»

RFC 1971, «IPv6 Stateless Address Autoconfiguration»

RFC 1970, «Neighbor Discovery for IP Version 6 (IPv6)»

RFC 1968, «The PPP Encryption Control Protocol (ECP)»

RFC 1964, «The Kerberos Version 5 GSS-API Mechanism»

RFC 1962, «The PPP Compression Control Protocol (CCP)»

RFC 1961, «GSS-API Authentication Method for SOCKS Version 5»

RFC 1960, «A String Representation of LDAP Search Filters»

RFC 1959, «An LDAP URL Format»

RFC 1938, «A One-Time Password System»

RFC 1933, «Transition Mechanisms for IPv6 Hosts and Routers»

RFC 1929, «Username/Password Authentication for SOCKS V5»

RFC 1928, «SOCKS Protocol Version 5»

RFC 1914, «How to Interact with a Whois++ Mesh»

RFC 1913, «Architecture of the Whois++ Index Service»

RFC 1894, «An Extensible Message Format for Delivery Status Notifications»

RFC 1893, «Enhanced Mail System Status Codes»

RFC 1892, «The Multipart/Report Content Type for the Reporting of Mail System Administrative Messages»

RFC 1891, «SMTP Service Extension for Delivery Status Notifications»

RFC 1890, «RTP Profile for Audio and Video Conferences with Minimal Control»

RFC 1889, «RTP: A Transport Protocol for Real-Time Applications»

RFC 1886, «DNS Extensions to Support IP Version 6»

RFC 1885, «Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6)»

RFC 1884, «IP Version 6 Addressing Architecture»

RFC 1883, «Internet Protocol, Version 6 (IPv6) Specification»

RFC 1866, «Hypertext Markup Language — 2.0»

RFC 1854, «SMTP Service Extension for Command Pipelining»

RFC 1848, «MIME Object Security Services»

RFC 1847, «Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted»

RFC 1835, «Architecture of the Whois++ Service»

RFC 1833, «Binding Protocols for ONC RPC Version 2»

RFC 1831, «RPC: Remote Procedure Call Protocol Specification Version 2»

RFC 1829, «The ESP DES-CBC Transform»

RFC 1828, «IP Authentication Using Keyed MD5»

RFC 1827, «IP Encapsulating Security Payload (ESP)»

RFC 1826, «IP Authentication Header»

RFC 1825, «Security Architecture for the Internet Protocol»

RFC 1812, «Requirements for IP Version 4 Routers»

RFC 1808, «Relative Uniform Resource Locators»

RFC 1798, «Connectionless Lightweight X.500 Directory Access Protocol»

RFC 1793, «Extending OSPF to Support Demand Circuits»

RFC 1784, «TFTP Timeout Interval and Transfer Size Options»

RFC 1783, «TFTP Block-Size Option»

RFC 1782, «TFTP Option Extension»

RFC 1781, «Using the OSI Directory to Achieve User-Friendly Naming»

RFC 1767, «MIME Encapsulation of EDI Objects»

RFC 1766, «Tags for the Identification of Languages»

RFC 1764, «The PPP XNS IDP Control Protocol (XNSCP)»

RFC 1763, «The PPP Banyan Vines Control Protocol (BVCP)»

RFC 1759, «Printer MIB»

RFC 1755, «ATM Signaling Support for IP over ATM»

RFC 1752, «The Recommendation for the IP Next Generation Protocol»

RFC 1749, «IEEE 802.5 Station Source Routing MIB Using SMIv2»

RFC 1747, «Definitions of Managed Objects for SNA Data Link Control (SDLC) Using SMIv2»

RFC 1745, «BGP4/IDRP for IP — OSPF Interaction»

RFC 1742, «AppleTalk Management Information Base II»

RFC 1740, «MIME Encapsulation of Macintosh Files — MacMIME»

RFC 1738, «Uniform Resource Locators (URLs)»

RFC 1734, «POP3 AUTHentication Command»

RFC 1731, «IMAP4 Authentication Mechanisms»

RFC 1730, «Internet Message Access Protocol — Version 4»

RFC 1717, «The PPP Multilink Protocol (MP)»

RFC 1697, «Relational Database Management System (RDBMS) Management Information Base (MIB) using SMIv2»

RFC 1696, «Modem Management Information Base (MIB) using SMIv2»

RFC 1695, «Definitions of Managed Objects for ATM Management Version 8.0 Using SMIv2»

RFC 1692, «Transport Multiplexing Protocol (TMux)»

RFC 1666, «Definitions of Managed Objects for SNA NAUs Using SMIv2. August 1994. (Format: TXT=134385 Bytes) (Obsoletes RFC1665)»

RFC 1665, «Definitions of Managed Objects for SNA NAUs Using SMIv2. July 1994. (Format: TXT=133381 Bytes) (Obsoleted by RFC1666)»

RFC 1663, «PPP Reliable Transmission»

RFC 1656, «BGP-4 Protocol Document Roadmap and Implementation Experience»

RFC 1655, «Application of the Border Gateway Protocol in the Internet»

RFC 1654, «A Border Gateway Protocol 4 (BGP-4)»

RFC 1650, «Definitions of Managed Objects for the Ethernet-Like Interface Types using SMIv2»

RFC 1648, «Postmaster Convention for X.400 Operations»

RFC 1647, «TN3270 Enhancements»

RFC 1638, «PPP Bridging Control Protocol (BCP)»

RFC 1628, «UPS Management Information Base»

RFC 1626, «Default IP MTU for Use over ATM AAL5»

RFC 1619, «PPP over SONET/SDH»

RFC 1618, «PPP over ISDN»

RFC 1612, «DNS Resolver MIB Extensions»

RFC 1611, «DNS Server MIB Extensions»

RFC 1604, «Definitions of Managed Objects for Frame Relay Service»

RFC 1598, «PPP in X.25»

RFC 1596, «Definitions of Managed Objects for Frame Relay Service»

RFC 1595, «Definitions of Managed Objects for the SONET/SDH Interface Type»

RFC 1587, «The OSPF NSSA Option»

RFC 1584, «Multicast Extensions to OSPF»

RFC 1582, «Extensions to RIP to Support Demand Circuits»

RFC 1577, «Classical IP and ARP over ATM»

RFC 1573, «Evolution of the Interfaces Group of MIB-II»

RFC 1572, «Telnet Environment Option»

RFC 1570, «PPP LCP Extensions»

RFC 1567, «X.500 Directory Monitoring MIB»

RFC 1566, «Mail Monitoring MIB»

RFC 1565, «Network Services Monitoring MIB»

RFC 1553, «Compressing IPX Headers over WAN Media (CIPX)»

RFC 1552, «The PPP Internetworking Packet Exchange Control Protocol (IPXCP)»

RFC 1544, «The Content-MD5 Header Field»

RFC 1541, «Dynamic Host Configuration Protocol»

RFC 1533, «DHCP Options and BOOTP Vendor Extensions»

RFC 1532, «Clarifications and Extensions for the Bootstrap Protocol»

RFC 1531, «Dynamic Host Configuration Protocol»

RFC 1525, «Definitions of Managed Objects for Source Routing Bridges»

RFC 1519, «Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy»

RFC 1518, «An Architecture for IP Address Allocation with CIDR»

RFC 1517, «Applicability Statement for the Implementation of Classless Inter-Domain Routing (CIDR)»

RFC 1515, «Definitions of Managed Objects for IEEE 802.3 Medium Attachment Units (MAUs)»

RFC 1514, «Host Resources MIB»

RFC 1513, «Token Ring Extensions to the Remote Network Monitoring MIB»

RFC 1512, «FDDI Management Information Base»

RFC 1510, «The Kerberos Network Authentication Service (V5)»

RFC 1509, «Generic Security Service API: C-Bindings»

RFC 1508, «Generic Security Service Application Program Interface»

RFC 1507, «DASS — Distributed Authentication Security Service»

RFC 1502, «X.400 Use of Extended Character Sets»

RFC 1496, «Rules for Downgrading Messages from X.400/88 to X.400/84 When MIME Content Types are Present in the Messages»

RFC 1495, «Mapping Between X.400 and RFC-822 Message Bodies»

RFC 1494, «Equivalences Between 1988 X.400 and RFC-822 Message Bodies»

RFC 1488, «The X.500 String Representation of Standard Attribute Syntaxes»

RFC 1487, «X.500 Lightweight Directory Access Protocol»

RFC 1485, «A String Representation of Distinguished Names (OSI-DS 23 v5)»

RFC 1483, «Multiprotocol Encapsulation over ATM Adaptation Layer 5»

RFC 1479, «Inter-Domain Policy Routing Protocol Specification: Version 1»

RFC 1478, «An Architecture for Inter-Domain Policy Routing»

RFC 1477, «IDPR as a Proposed Standard»

RFC 1474, «The Definitions of Managed Objects for the Bridge Network Control Protocol of the Point-to-Point Protocol»

RFC 1473, «The Definitions of Managed Objects for the IP Network Control Protocol of the Point-to-Point Protocol»

RFC 1472, «The Definitions of Managed Objects for the Security Protocols of the Point-to-Point Protocol»

RFC 1471, «The Definitions of Managed Objects for the Link Control Protocol of the Point-to-Point Protocol»

RFC 1469, «IP Multicast over Token-Ring Local Area Networks»

RFC 1461, «SNMP MIB Extension for Multiprotocol Interconnect over X.25»

RFC 1452, «Coexistence between Version 1 and Version 2 of the Internet-Standard Network Management Framework»

RFC 1450, «Management Information Base for Version 2 of the Simple Network Management Protocol (SNMP-V2)»

RFC 1449, «Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMP-V2)»

RFC 1448, «Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMP-V2)»

RFC 1444, «Conformance Statements for Version 2 of the Simple Network Management Protocol (SNMP-V2)»

RFC 1443, «Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMP-V2)»

RFC 1442, «Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMP-V2)»

RFC 1441, «Introduction to Version 2 of the Internet-Standard Network Management Framework»

RFC 1427, «SMTP Service Extension for Message Size Declaration»

RFC 1426, «SMTP Service Extension for 8bit-MIMEtransport»

RFC 1425, «SMTP Service Extensions»

RFC 1424, «Privacy Enhancement for Internet Electronic Mail: Part IV: Key Certification and Related Services»

RFC 1423, «Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifiers»

RFC 1422, «Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management»

RFC 1421, «Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures»

RFC 1420, «SNMP over IPX»

RFC 1419, «SNMP over AppleTalk»

RFC 1418, «SNMP over OSI»

RFC 1415, «FTP-FTAM Gateway Specification»

RFC 1414, «Identification MIB»

RFC 1413, «Identification Protocol»

RFC 1407, «Definitions of Managed Objects for the DS3/E3 Interface Type»

RFC 1406, «Definitions of Managed Objects for the DS1 and E1 Interface Types»

RFC 1403, «BGP OSPF Interaction»

RFC 1397, «Default Route Advertisement in BGP2 and BGP3 Version of the Border Gateway Protocol»

RFC 1389, «RIP Version 2 MIB Extensions»

RFC 1388, «RIP Version 2 Carrying Additional Information»

RFC 1382, «SNMP MIB Extension for the X.25 Packet Layer»

RFC 1381, «SNMP MIB Extension for X.25 LAPB»

RFC 1378, «The PPP AppleTalk Control Protocol (ATCP)»

RFC 1377, «The PPP OSI Network Layer Control Protocol (OSINLCP)»

RFC 1376, «The PPP DECnet Phase IV Control Protocol (DNCP)»

RFC 1374, «IP and ARP on HIPPI»

RFC 1372, «Telnet Remote Flow Control Option»

RFC 1370, «Applicability Statement for OSPF»

RFC 1369, «Implementation Notes and Experience for the Internet Ethernet MIB»

RFC 1368, «Definition of Managed Objects for IEEE 802.3 Repeater Devices»

RFC 1364, «BGP OSPF Interaction»

RFC 1354, «IP Forwarding Table MIB»

RFC 1353, «Definitions of Managed Objects for Administration of SNMP Parties»

RFC 1352, «SNMP Security Protocols»

RFC 1351, «SNMP Administrative Model»

RFC 1349, «Type of Service in the Internet Protocol Suite»

RFC 1341, «MIME (Multipurpose Internet Mail Extensions): Mechanisms for Specifying and Describing the Format of Internet Message Bodies»

RFC 1334, «PPP Authentication Protocols»

RFC 1333, «PPP Link Quality Monitoring»

RFC 1332, «The PPP Internet Protocol Control Protocol (IPCP)»

RFC 1331, «Point-to-Point Protocol (PPP) for the Transmission of Multi-protocol Datagrams over Point-to-Point Links»

RFC 1328, «X.400 1988 to 1984 Downgrading»

RFC 1327, «Mapping Between X.400 (1988)/ISO 10021 and RFC 822»

RFC 1323, «TCP Extensions for High Performance»

RFC 1318, «Definitions of Managed Objects for Parallel-Printer-Like Hardware Devices»

RFC 1317, «Definitions of Managed Objects for RS-232-Like Hardware Devices»

RFC 1316, «Definitions of Managed Objects for Character Stream Devices»

RFC 1315, «Management Information Base for Frame Relay DTEs»

RFC 1314, «A File Format for the Exchange of Images in the Internet»

RFC 1304, «Definitions of Managed Objects for the SIP Interface Type»

RFC 1294, «Multiprotocol Interconnect over Frame Relay»

RFC 1293, «Inverse Address Resolution Protocol»

RFC 1289, «DECnet Phase IV MIB Extensions»

RFC 1286, «Definitions of Managed Objects for Bridges»

RFC 1285, «FDDI Management Information Base»

RFC 1284, «Definitions of Managed Objects for the Ethernet-Like Interface Types»

RFC 1277, «Encoding Network Addresses to Support Operation over Non-OSI Lower Layers»

RFC 1276, «Replication and Distributed Operations Extensions to Provide an Internet Directory Using X.500»

RFC 1274, «The COSINE and Internet X.500 Schema»

RFC 1271, «Remote Network Monitoring Management Information Base»

RFC 1269, «Definitions of Managed Objects for the Border Gateway Protocol: Version 3»

RFC 1256, «ICMP Router Discovery Messages»

RFC 1253, «OSPF Version 2 Management Information Base»

RFC 1252, «OSPF Version 2 Management Information Base»

RFC 1248, «OSPF Version 2 Management Information Base»

RFC 1243, «AppleTalk Management Information Base»

RFC 1240, «OSI Connectionless Transport Services on Top of UDP: Version 1»

RFC 1239, «Reassignment of Experimental MIBs to Standard MIBs»

RFC 1237, «Guidelines for OSI NSAP Allocation in the Internet»

RFC 1234, «Tunneling IPX Traffic through IP Networks»

RFC 1233, «Definitions of Managed Objects for the DS3 Interface Type»

RFC 1232, «Definitions of Managed Objects for the DS1 Interface Type»

RFC 1231, «IEEE 802.5 Token Ring MIB»

RFC 1229, «Extensions to the Generic-Interface MIB»

RFC 1220, «Point-to-Point Protocol Extensions for Bridging»

RFC 1195, «Use of OSI IS-IS for Routing in TCP/IP and Dual Environments»

RFC 1172, «Point-to-Point Protocol (PPP) Initial Configuration Options»

RFC 1144, «Compressing TCP/IP Headers for Low-Speed Serial Links»

RFC 1139, «Echo Function for ISO 8473»

RFC 1134, «Point-to-Point Protocol: Proposal for Multiprotocol Transmission of Datagrams over Point-to-Point Links»

RFC 1131, «OSPF Specification»

RFC 1116, «Telnet Linemode Option»

RFC 1096, «Telnet X Display Location Option»

RFC 1091, «Telnet Terminal Type Option»

RFC 1079, «Telnet Terminal Speed Option»
RFC 1073, «Telnet Window Size Option»
RFC 1053, «Telnet X.3 PAD Option»
RFC 1043, «Telnet Data Entry Terminal Option: DODIIS Implementation»
RFC 1041, «Telnet 3270 Regime Option»
RFC 977, «Network News Transfer Protocol»
RFC 946, «Telnet Terminal Location Number Option»
RFC 933, «Output-Marking Telnet Option»
RFC 927, «TACACS User-Identification Telnet Option»
RFC 885, «Telnet End-of-Record Option»
RFC 779, «Telnet Send-Location Option»
RFC 749, «Telnet SUPDUP-Output Option»
RFC 736, «Telnet SUPDUP Option»
RFC 735, «Revised Telnet Byte Macro Option»
RFC 727, «Telnet Logout Option»
RFC 726, «Remote-Controlled Transmission and Echoing Telnet Option»
RFC 698, «Telnet Extended ASCII Option»
RFC 658, «Telnet Output Linefeed Disposition»
RFC 657, «Telnet Output Vertical Tab Disposition Option»
RFC 656, «Telnet Output Vertical Tab Stops Option»
RFC 655, «Telnet Output Form-Feed Disposition Option»
RFC 654, «Telnet Output Horizontal Tab Disposition Option»
RFC 653, «Telnet Output Horizontal Tab Stops Option»
RFC 652, «Telnet Output Carriage-Return Disposition Option»

Предметный указатель

Н

О

П

C

Дилип Найк

Стандарты и протоколы Интернета

Перевод с английского под общей редакцией *Ю. С. Конова*

Переводчики *Н. К. Козловская, Ю. С. Конов, А. Н. Некрасов, А. А. Савельев*

Технический консультант *О. О. Михальский*

Редактор *Ю. П. Леонова*

Компьютерная верстка *В. Б. Хильченко*

Технический редактор *Н. Г. Тимченко*

Дизайн обложки *Е. В. Козлова*

Оригинал-макет выполнен с использованием издательской системы Adobe PageMaker 6.0

Главный редактор *А. И. Козлов*

Главный менеджер *М. И. Царейкин*

РУССКАЯ РЕДАКЦИЯ

Лицензия ЛР № 066422 от 19.03.99 г.