В.А. Погонин, С.Б. Путин,
А.А. Третьяков, В.А. Шиганцов

СЕТИ И СИСТЕМЫ ТЕЛЕКОММУНИКАЦИЙ

Москва
"Издательство Машиностроение-1"
2005

В.А. Погонин, С.Б. Путин, А.А. Третьяков, В.А. Шиганцов

СЕТИ И СИСТЕМЫ ТЕЛЕКОММУНИКАЦИЙ

Допущено Учебно-методическим объединением вузов по образованию в области автоматизированного машиностроения (*УМО АМ*) *в качестве учебного пособия для студентов высших учебных заведений*, *обучающихся по направлению подготовки дипломированных специалистов* "*Автоматизированные технологии и производства*"

Москва
"Издательство Машиностроение-1"
2005

УДК 681.32 (075.8)
ББК ←973.202-04я73
С33

Р е ц е н з е н т ы :

Руководитель направления института системного анализа РАН
доктор физико-математических наук, профессор
А.П. Афанасьев

Директор ТамбовЦНИТ
кандидат технических наук
В.Е. Подольский

С33 **Сети и системы телекоммуникаций**: Учебное пособие / В.А. Погонин, С.Б. Путин, А.А. Третьяков, В.А. Шиганцов. М.: "Издательство Машиностроение-1", 2005. 172 с.

Описываются основные аспекты архитектуры и технологии современных компьютерных сетей и систем телекоммуникаций. Рассматривается их функционирование, эффективность и перспективы развития.

Рекомендуется в качестве учебного пособия студентам, обучающихся по специальностям "Автоматизация технологических процессов и производств", "Прикладная информатика в экономике".

УДК 681.32 (075.8)
ББК ←973.202-04я73

ISBN 5-94275-231-1

Учебное издание

ПОГОНИН Василий Александрович,
ПУТИН Сергей Борисович,
ТРЕТЬЯКОВ Александр Александрович,
ШИГАНЦОВ Владислав Александрович

СЕТИ И СИСТЕМЫ
ТЕЛЕКОММУНИКАЦИЙ

Учебное пособие

Редактор З.Г. Чернова

Инженер по компьютерному макетированию М.Н. Рыжкова

Подписано в печать 08.12.05
Формат 60 × 84/16. Бумага офсетная. Печать офсетная.
Гарнитура Times New Roman. Объем: 10,0 усл. печ. л.; 10,0 уч.-изд. л.
Тираж 400 экз. С. 863^{М}

"Издательство Машиностроение-1", 107076, Москва, Стромынский пер., 4

Подготовлено к печати и отпечатано в издательско-полиграфическом центре
Тамбовского государственного технического университета,
392000, Тамбов, Советская 106, к. 14

Введение

В соответствии с тенденцией быстрого продвижения общества к широкому и всестороннему использованию достижений компьютерной техники и бурно развивающихся информационных технологий студенты, как будущие специалисты в области управления технологическими процессами и производствами, должны знать аппаратные и программные средства современных компьютерных сетей, возможности доступа к удаленным информационным ресурсам и их использования, знать проблемы разработки новых высокотехнологичных компьютерных сетей.

Учебное пособие предназначено для студентов, которые хотят получить базовые знания о принципах построения компьютерных сетей и систем телекоммуникаций, понять особенности традиционных и перспективных технологий локальных и глобальных сетей и управления такими сетями.

1. История развития

вычислительных сетей

1.1. ЭВОЛЮЦИЯ КОМПЬЮТЕРНЫХ СЕТЕЙ

Сети передачи данных, называемые также вычислительными или компьютерными сетями, являются результатом эволюции двух важнейших научно-технических отраслей современной цивилизации – компьютерных и телекоммуникационных технологий (рис. 1.1). С одной стороны, сети передачи данных представляют собой частный случай распределенных вычислительных систем, в которых группа компьютеров согласованно выполняет набор взаимосвязанных задач, обмениваясь данными в автоматическом режиме; с другой – компьютерные сети могут рассматриваться как средство передачи информации на большие расстояния, для чего в них применяются методы кодирования и мультиплексирования данных, получившие развитие в различных телекоммуникационных системах. Итак,

- *компьютерная* сеть – это набор компьютеров, связанных коммуникационной системой и снабженных соответствующим программным обеспечением, которое предоставляет пользователям сети доступ к ресурсам этого набора компьютеров;
- сеть могут образовывать компьютеры разных типов – небольшие микропроцессоры, рабочие станции, мини-компьютеры, персональные компьютеры или суперкомпьютеры;

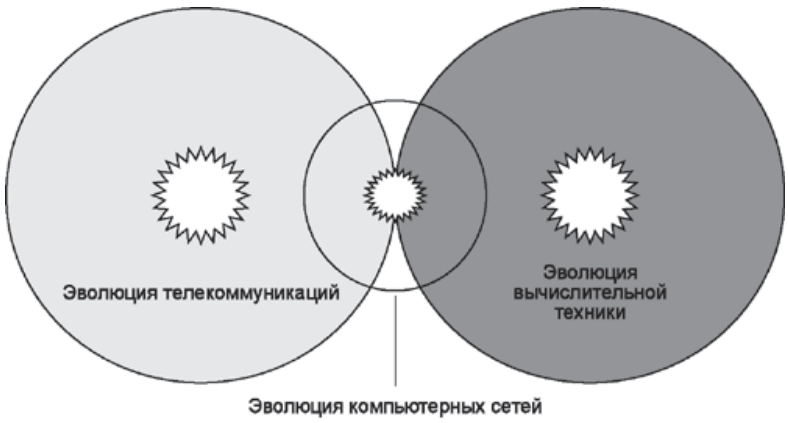

Рис. 1.1. Эволюция компьютерных сетей на стыке вычислительной техники и телекоммуникационных технологий

- передачу сообщений между любой парой компьютеров сети обеспечивает коммуникационная система, которая может включать кабели, повторители, коммутаторы, маршрутизаторы и другие устройства;
- компьютерная сеть позволяет пользователю работать со своим компьютером, как с автономным, и добавляет к этому возможность доступа к информационным и аппаратным ресурсам других компьютеров сети.

Идея компьютера была предложена английским математиком Чарльзом Бэбиджем (Charles Babig) в середине девятнадцатого века. Однако его механическая "аналитическая машина" по-настоящему так и не заработала.

Подлинное рождение цифровых вычислительных машин произошло вскоре после окончания второй мировой войны. В середине 40-х годов XX века были созданы первые ламповые вычислительные устройства. Для этого периода характерно следующее:

– компьютер представлял собой скорее предмет исследования, а не инструмент для решения каких-либо практических задач из других областей;

– программирование осуществлялось исключительно на машинном языке;

– не было никакого системного программного обеспечения, кроме библиотек математических и служебных подпрограмм;

– операционные системы еще не появились, все задачи организации вычислительного процесса решались вручную каждым программистом с пульта управления.

С середины 50-х годов XX века начался следующий период в развитии вычислительной техники, связанный с появлением новой технической базы – полупроводниковых элементов. В этот период:

– выросло быстродействие процессоров, увеличились объемы оперативной и внешней памяти;

– появились первые алгоритмические языки, и, таким образом, к библиотекам математических и служебных подпрограмм добавился новый тип системного программного обеспечения – трансляторы;

– разработаны первые системные управляющие программы – мониторы, которые автоматизировали всю последовательность действий оператора по организации вычислительного процесса.

Программные мониторы явились прообразом современных операционных систем, они стали первыми системными программами, предназначенными не для обработки данных, а для управления вычислительным процессом.

В ходе реализации мониторов был разработан формализованный язык управления заданиями, с помощью которого программист сообщал системе и оператору, какие действия и в какой последовательности он хотел бы выполнить на вычислительной машине. Типовой набор директив обычно включал признак начала отдельной работы, вызов транслятора, вызов загрузчика, признаки начала и конца исходных данных.

Оператор составлял пакет заданий, которые в дальнейшем без его участия последовательно запускались на выполнение монитором. Кроме того, монитор был способен самостоятельно обрабатывать наиболее распространенные аварийные ситуации, возникающие при работе пользовательских программ, такие, как отсутствие исходных данных, переполнение регистров, деление на ноль, обращение к несуществующей области памяти и т.д.

1.2. МУЛЬТИПРОГРАММИРОВАНИЕ

Следующий важный период развития операционных систем относится к 1965 – 1975 годам. В это время в технической базе вычислительных машин произошел переход от отдельных полупроводниковых элементов типа транзисторов к интегральным микросхемам, что открыло путь к появлению следующего поколения компьютеров, представителем которого является, например, IBM/360.

В этот период были реализованы практически все основные механизмы, присущие современным операционным системам (ОС): мультипрограммирование, мультипроцессирование, поддержка многотерминального многопользовательского режима, виртуальная память, файловые системы, разграничение доступа и сетевая работа. В эти годы начинается расцвет системного программирования. Из направления прикладной математики, представляющего интерес для узкого круга специалистов, системное программирование превращается в отрасль индустрии, оказывающую непосредственное влияние на практическую деятельность миллионов людей.

В условиях резко возросших возможностей компьютера, связанных с обработкой и хранением данных, выполнение только одной программы в каждый момент времени оказалось крайне неэффективным.

Начались разработки в области мультипрограммирования.

Мультипрограммирование – способ организации вычислительного процесса, при котором в памяти компьютера находится одновременно несколько программ, попеременно выполняющихся на одном процессоре.

Мультипрограммирование было реализовано в двух вариантах:

- пакетная обработка;

- разделение времени.

Системы пакетной обработки предназначались для решения задач в основном вычислительного характера, не требующих быстрого получения результатов. Главной целью и критерием эффективности систем пакетной обработки является максимальная пропускная способность, то есть решение максимального числа задач в единицу времени.

Для достижения этой цели в системах пакетной обработки используется следующая схема функционирования (рис. 1.2): в начале работы формируется пакет заданий, каждое задание содержит требование к системным ресурсам; из этого пакета заданий формируется мультипрограммный набор, то есть множество одновременно выполняемых задач. Для одновременного выполнения выбираются задачи, предъявляющие к ресурсам различные требования, так, чтобы обеспечивалась сбалансированная загрузка всех устройств вычислительной машины. Например, в мультипрограммном наборе желательно присутствие и вычислительных задач, и задач с интенсивным вводом-выводом. Таким образом, выбор нового задания из пакета заданий зависит от внутренней ситуации, складывающейся в системе, то есть выбирается "выгодное" задание. Следовательно, в вычислительных системах, работающих под управлением пакетных ОС, невозможно гарантировать выполнение того или иного задания в течение определенного периода времени.

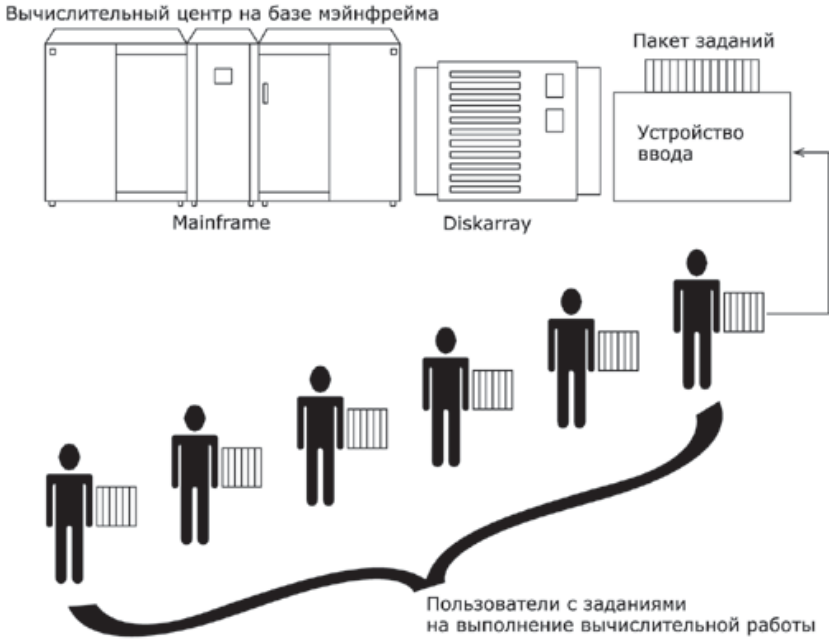

Рис. 1.2. Централизованный характер вычислений в системах пакетной обработки

В системах пакетной обработки переключение процессора с одной задачи на другую происходит по инициативе самой активной задачи, например, когда она "отказывается" от процессора из-за необходимости выполнить операцию ввода-вывода. Поэтому существует высокая вероятность того, что одна задача может надолго занять процессор, и выполнение интерактивных задач станет невозможным. Взаимодействие пользователя с вычислительной машиной, на которой установлена система пакетной обработки, сводится к тому, что пользователь приносит задание, отдает его диспетчеру-оператору, а в конце дня после выполнения всего пакета заданий получает результат. Очевидно, что такой порядок повышает эффективность функционирования аппаратуры, но снижает эффективность работы пользователя.

В *системах разделения времени* пользователям (или одному пользователю) предоставляется возможность интерактивной работы сразу с несколькими приложениями. Для этого каждое приложение должно регулярно взаимодействовать с пользователем. Понятно, что в пакетных системах возможности диалога пользователя с приложением ограничены.

В системах разделения времени эта проблема решается за счет того, что ОС принудительно периодически приостанавливает приложения, не дожидаясь, когда они сами освободят процессор. Всем приложениям попеременно выделяется квант процессорного времени. Таким образом, пользователи, запустившие программы на выполнение, получают возможность поддерживать с ними диалог.

Системы разделения времени призваны исправить основной недостаток систем пакетной обработки – изоляцию пользователя-программиста от процесса выполнения задач. Каждому пользователю в этом случае предоставляется терминал, с которого он может вести диалог со своей программой. Так как в

системах разделения времени каждой задаче выделяется только квант процессорного времени, ни одна задача не занимает процессор надолго, и время ответа оказывается приемлемым. Если квант небольшой, то у всех пользователей, одновременно работающих на одной и той же машине, складывается впечатление, что каждый из них использует машину единолично.

Ясно, что системы разделения времени обладают меньшей пропускной способностью, чем системы пакетной обработки, так как на выполнение принимается каждая запущенная пользователем задача, а не та, которая "выгодна" системе. Кроме того, производительность системы снижается из-за дополнительного расходования вычислительной мощности на более частое переключение процессора с задачи на задачу. Это вполне соответствует тому, что критерием эффективности систем разделения времени является не максимальная пропускная способность, а удобство и эффективность работы пользователя. Вместе с тем, мультипрограммное выполнение интерактивных приложений повышает и пропускную способность компьютера (пусть и не в такой степени, как пакетные системы). Аппаратура загружается лучше, поскольку пока одно приложение ждет сообщения пользователя, другие приложения могут обрабатываться процессором.

1.3. МНОГОТЕРМИНАЛЬНЫЕ СИСТЕМЫ – ПРООБРАЗ СЕТИ

Терминалы, выйдя за пределы вычислительного центра, рассредоточились по всему предприятию. Многотерминальный режим использовался не только в системах разделения времени, но и в системах пакетной обработки. При этом не только оператор, но и все пользователи получали возможность формировать свои задания и управлять их выполнением со своего терминала. Такие операционные системы получили название систем удаленного ввода заданий.

Терминальные комплексы могли располагаться на большом расстоянии от процессорных стоек, соединяясь с ними с помощью различных глобальных связей – модемных соединений телефонных сетей или выделенных каналов. Для поддержки удаленной работы терминалов в операционных системах появились специальные программные модули, реализующие различные (в то время, как правило, нестандартные) протоколы связи. Такие вычислительные системы с удаленными терминалами, сохраняя централизованный характер обработки данных, в какой-то степени являлись прообразом современных компьютерных сетей (рис. 1.3), а соответствующее системное программное обеспечение – прообразом сетевых операционных систем.

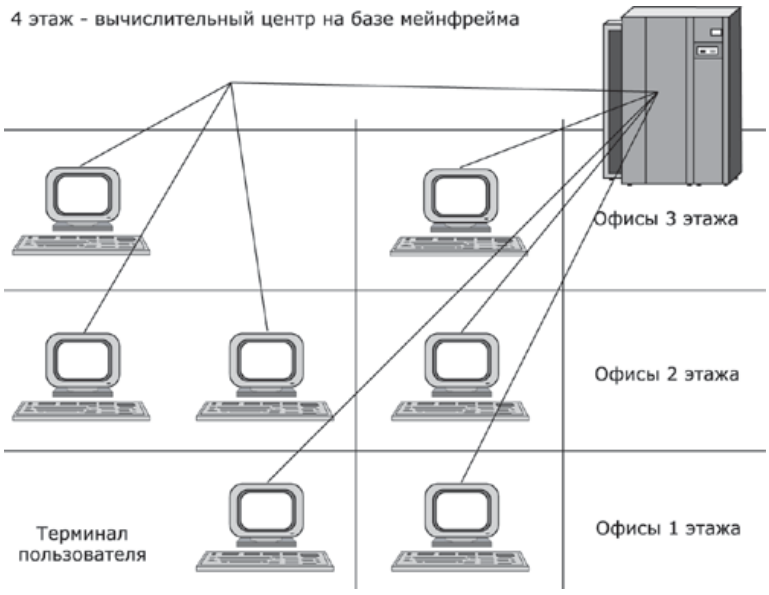

Рис. 1.3. Многотерминальная система – прообраз вычислительной сети

Многотерминальные централизованные системы уже имели все внешние признаки локальных вычислительных сетей, однако по существу ими не являлись, так как сохраняли сущность централизованной обработки данных автономно работающего компьютера.

Действительно, рядовой пользователь работу за терминалом мэйнфрейма воспринимал примерно так же, как сейчас воспринимает работу за подключенным к сети персональным компьютером. Пользователь мог получить доступ к общим файлам и периферийным устройствам, при этом у него создавалась полная иллюзия единоличного владения компьютером, так как он мог запустить нужную ему программу в любой момент и почти сразу же получить результат. (Некоторые далекие от вычислительной техники пользователи даже были уверены, что все вычисления выполняются внутри их дисплея.)

1.4. ПЕРВЫЕ СЕТИ – ГЛОБАЛЬНЫЕ

Хотя теоретические работы по созданию концепций сетевого взаимодействия велись почти с момента появления вычислительных машин, значимые практические результаты по объединению компьютеров в сети были получены лишь в конце 60-х, когда с помощью глобальных связей и техники коммутации пакетов удалось реализовать взаимодействие машин класса мэйнфреймов и суперкомпьютеров (рис. 1.4). Эти дорогостоящие компьютеры хранили уникальные данные и программы, обмен которыми позволил повысить эффективность их использования.

Но еще до реализации связей "компьютер-компьютер" была решена более простая задача – организация связи "удаленный терминал-компьютер". Терминалы, находящиеся от компьютера на расстоянии многих сотен, а то и тысяч километров, соединялись с компьютерами

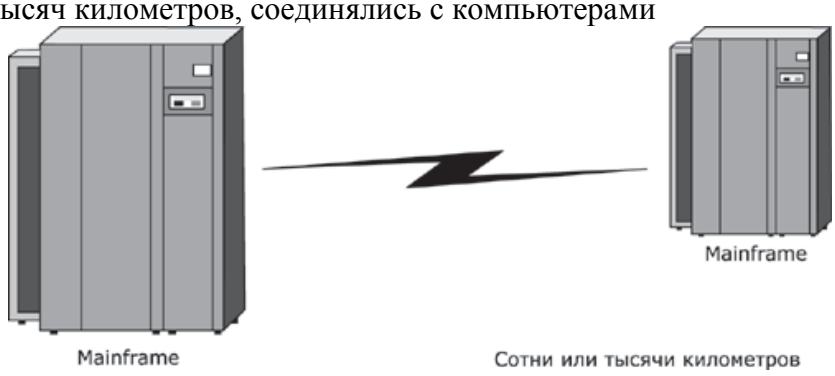

Рис. 1.4. Объединение удаленных супер-ЭВМ глобальными связями

через телефонные сети с помощью модемов. Такие сети позволяли многочисленным пользователям получать удаленный доступ к разделяемым ресурсам нескольких мощных компьютеров класса супер-ЭВМ.

И только потом были разработаны средства обмена данными между компьютерами в автоматическом режиме. На основе этого механизма в первых сетях были реализованы службы обмена файлами, синхронизации баз данных, электронной почты и другие, ставшие теперь традиционными, сетевые службы.

В 1969 году министерство обороны США инициировало работы по объединению в общую сеть суперкомпьютеров оборонных и научно-исследовательских центров. Эта сеть, получившая название ARPANET послужила отправной точкой для создания первой и самой известной ныне глобальной сети – Internet. Сеть ARPANET объединяла компьютеры разных типов, работавшие под управлением различных ОС с дополнительными модулями, реализующими коммуникационные протоколы, общие для всех компьютеров сети. Такие ОС можно считать первыми сетевыми операционными системами.

Сетевые ОС в отличие от многотерминальных позволяли не только рассредоточить пользователей, но и организовать распределенное хранение и обработку данных между несколькими компьютерами, связанными электрическими связями. Любая сетевая операционная система, с одной стороны, выполняет все функции локальной операционной системы, а с другой – обладает некоторыми дополнительными средствами, позволяющими ей взаимодействовать по сети с операционными системами других компьютеров. Программные модули, реализующие сетевые функции, появлялись в операционных системах постепенно, по мере развития сетевых технологий, аппаратной базы компьютеров и возникновения новых задач, требующих сетевой обработки.

В 1974 году компания IBM объявила о создании собственной сетевой архитектуры для своих мэйнфреймов, получившей название SNA (System Network Architecture, системная сетевая архитектура). В это же время в Европе активно велись работы по созданию и стандартизации сетей X.25.

Таким образом, хронологически первыми появились глобальные сети (Wide Area Networks**,** WAN), то есть сети, объединяющие территориально рассредоточенные компьютеры, возможно, находящиеся в различных городах и странах. Именно при построении глобальных сетей были впервые предложены и отработаны многие основные идеи и концепции современных вычислительных сетей, такие, например, как многоуровневое построение коммуникационных протоколов, технология коммутации пакетов и маршрутизация пакетов в составных сетях.

Глобальные компьютерные сети очень многое унаследовали от других, гораздо более старых и глобальных сетей – телефонных.

Главным результатом создания первых глобальных компьютерных сетей был отказ от принципа коммутации каналов, на протяжении многих десятков лет успешно использовавшегося в телефонных сетях.

Выделяемый на все время сеанса связи составной канал с постоянной скоростью не мог эффективно использоваться пульсирующим трафиком компьютерных данных, у которого периоды интенсивного обмена чередуются с продолжительными паузами. Эксперименты и математическое моделирование показали, что пульсирующий и в значительной степени не чувствительный к задержкам компьютерный трафик гораздо эффективней передается по сетям, использующим принцип коммутации пакетов, когда данные разделяются на небольшие порции, которые самостоятельно перемещаются по сети за счет встраивания адреса конечного узла в заголовок пакета.

Так как прокладка высококачественных линий связи на большие расстояния обходится очень дорого, в первых глобальных сетях часто использовались уже существующие каналы связи, изначально предназначенные совсем для других целей. Например, в течение многих лет глобальные сети строились на основе телефонных каналов тональной частоты, способных в каждый момент времени вести передачу только одного разговора в аналоговой форме. Поскольку скорость передачи дискретных компьютерных данных по таким каналам была очень низкой (десятки килобит в секунду), набор предоставляемых услуг в глобальных сетях такого типа обычно ограничивался передачей файлов, преимущественно в фоновом режиме, и электронной почтой.

Помимо низкой скорости такие каналы имеют и другой недостаток – они вносят значительные искажения в передаваемые сигналы. Поэтому протоколы глобальных сетей, построенных с использованием каналов связи низкого качества, отличаются сложными процедурами контроля и восстановления данных. Типичным примером таких сетей являются сети X.25, разработанные еще в начале 70-х годов XX столетия, когда низкоскоростные аналоговые каналы, арендуемые у телефонных компаний, были преобладающим типом каналов, соединяющих компьютеры и коммутаторы глобальной вычислительной сети.

Развитие технологии глобальных компьютерных сетей во многом определялся прогрессом телефонных сетей. С конца 60-х годов в телефонных сетях все чаще стала применяться передача голоса в цифровой форме, что привело к появлению высокоскоростных цифровых каналов, соединяющих АТС и позволяющих одновременно передавать десятки и сотни разговоров. Была разработана специальная технология плезиохронной цифровой иерархии (Plesiochronous Digital Hierarchy, PDH), предназначенная для создания так называемых первичных, или опорных, сетей. Такие сети не предоставляют услуг конечным пользователям, они являются фундаментом, на котором строятся скоростные цифровые каналы "точка-точка", соединяющие оборудование другой (так называемой наложенной) сети, которая уже работает на конечного пользователя.

Первоначально технология PDH, поддерживающая скорости до 140 Мбит/с, была внутренней технологией телефонных компаний. Однако со временем эти компании стали сдавать часть своих каналов PDH в аренду предприятиям, которые использовали их для создания собственных телефонных и глобальных компьютерных сетей.

Появившаяся в конце 80-х годов технология синхронной цифровой иерархии (Synchronous Digital Hierarchy, SDH) расширила диапазон скоростей цифровых каналов до 10 Гбит/с, а технология спектрального мультиплексирования DWDM (Dense Wave Division Multiplexing) – до сотен гигабит и даже нескольких терабит в секунду.

Сегодня глобальные сети по разнообразию и качеству предоставляемых услуг догнали локальные сети, которые долгое время лидировали в этом отношении, хотя и появились на свет значительно позже.

1.5. МИНИ-КОМПЬЮТЕРЫ – ПРЕДВЕСТНИКИ ЛОКАЛЬНЫХ СЕТЕЙ

В начале 70-х годов произошло важное событие, непосредственно повлиявшее на эволюцию компьютерных сетей.

В результате технологического прорыва в области производства компьютерных компонентов появились большие интегральные схемы (БИС). Их сравнительно невысокая стоимость и богатые функциональные возможности привели к созданию мини-компьютеров, которые стали реальными конкурентами мэйнфреймов.

Даже небольшие подразделения предприятий получили возможность иметь собственные компьютеры. К середине 70-х годов стали широко использоваться мини-компьютеры PDP-11, Nova, HP.

С помощью мини-компьютеров осуществлялось управление технологическим оборудованием и выполнялись другие задачи уровня отдела предприятия. Таким образом, появилась концепция распределения компьютерных ресурсов по всему предприятию. Однако при этом все компьютеры одной организации по-прежнему продолжали работать автономно (рис. 1.5).

Архитектура мини-компьютеров была значительно упрощена по сравнению с мэйнфреймами, что нашло отражение и в их операционных системах. Многие функции мультипрограммных многопользовательских ОС мэйнфреймов были усечены, с учетом ограниченности ресурсов мини-компьютеров. Операционные системы мини-компьютеров часто стали делать специализированными, например, только для управления в реальном времени (ОС RT-11 для мини-компьютеров PDP-11) или только

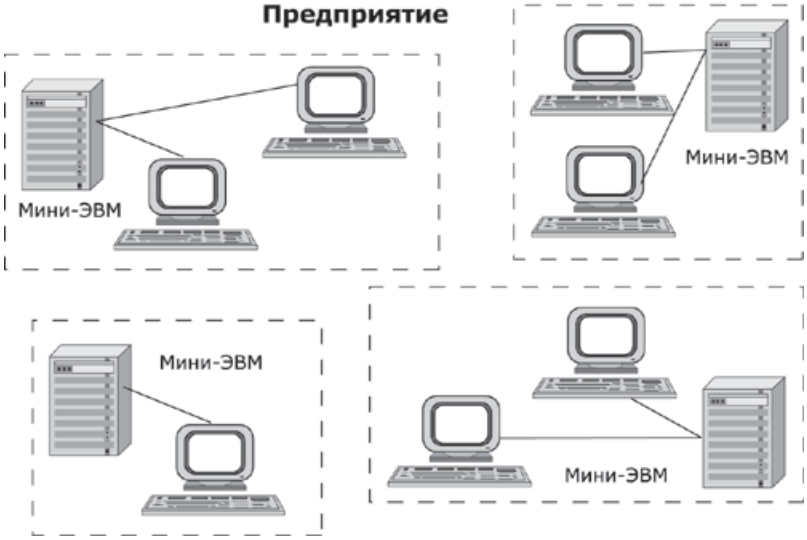

Рис. 1.5. Автономное использование нескольких мини-компьютеров на одном предприятии

для поддержания режима разделения времени (RSX-11M для тех же компьютеров). Эти операционные системы не всегда были многопользовательскими, что во многих случаях оправдывалось невысокой стоимостью машин. Важной вехой в истории мини-компьютеров и вообще в истории операционных систем стало создание ОС Unix.

1.6. ПОЯВЛЕНИЕ СТАНДАРТНЫХ ТЕХНОЛОГИЙ ЛОКАЛЬНЫХ СЕТЕЙ

В середине 80-х годов XX века положение дел в локальных сетях стало меняться. Утвердились стандартные технологии объединения компьютеров в сеть – Ethernet, Arcnet, Token Ring, Token Bus, несколько позже – FDDI.

Все *стандартные технологии локальных сетей* опирались на тот же принцип коммутации, который был с успехом опробован и доказал свои преимущества при передаче трафика данных в глобальных компьютерных сетях – принцип коммутации пакетов.

Стандартные сетевые технологии сделали задачу построения локальной сети почти тривиальной. Для создания сети достаточно было приобрести сетевые адаптеры соответствующего стандарта, например Ethernet, стандартный кабель, присоединить адаптеры к кабелю стандартными разъемами и установить на компьютер одну из популярных сетевых операционных систем, например Novell NetWare. После этого сеть начинала работать, и последующее присоединение каждого нового компьютера не вызывало никаких проблем – естественно, если на нем был установлен сетевой адаптер той же технологии.

В 80-е годы были приняты основные стандарты на коммуникационные технологии для локальных сетей: в 1980 году – Ethernet, в 1985 – Token Ring, в конце 80-х – FDDI. Это позволило обеспечить совместимость сетевых операционных систем на нижних уровнях, а также стандартизировать интерфейс ОС с драйверами сетевых адаптеров.

Конец 90-х выявил явного лидера среди технологий локальных сетей – семейство Ethernet, в которое вошли классическая технология Ethernet 10 Мбит/с, а также Fast Ethernet 100 Мбит/с и Gigabit Ethernet 1000 Мбит/с. Простые алгоритмы работы предопределили низкую стоимость оборудования Ethernet. Широкий диапазон иерархии скоростей позволяет рационально строить локальную сеть, применяя ту технологию, которая в наибольшей степени отвечает задачам предприятия и потребностям

пользователей. Важно также, что все технологии Ethernet очень близки друг другу по принципам работы, что упрощает обслуживание и интеграцию построенных на их основе сетей.

1.7. РОЛЬ ПЕРСОНАЛЬНЫХ КОМПЬЮТЕРОВ В ЭВОЛЮЦИИ КОМПЬЮТЕРНЫХ СЕТЕЙ

Начало 80-х годов XX века связано с еще одним знаменательным для истории сетей событием – появлением персональных компьютеров.

Эти устройства стали идеальными элементами для построения сетей: с одной стороны, они были достаточно мощными для работы сетевого программного обеспечения, а с другой – явно нуждались в объединении вычислительной мощности для решения сложных задач, а также разделения дорогих периферийных устройств и дисковых массивов. Поэтому персональные компьютеры стали активно использоваться в локальных сетях, причем не только в качестве клиентских компьютеров, но и в качестве центров хранения и обработки данных, то есть сетевых серверов, потеснив с этих ролей мини-компьютеры и мэйнфреймы.

С точки зрения архитектуры персональные компьютеры ничем не отличались от мини-компьютеров типа PDP-11, но их стоимость была существенно ниже. Если с появлением мини-компьютера возможность иметь собственную вычислительную машину получили отделы предприятий или университеты, то создание персонального компьютера дало такую возможность отдельному человеку.

Создание персональных компьютеров послужило мощным катализатором для бурного роста локальных сетей, поскольку появилась отличная материальная основа в виде десятков и сотен машин, принадлежащих одному предприятию и расположенных в пределах одного здания.

Компьютеры стали использоваться не только специалистами, что потребовало разработки "дружественного" программного обеспечения, и предоставление соответствующих функций стало прямой обязанностью операционных систем. В результате поддержка сетевых функций стала для ОС персональных компьютеров необходимым условием.

Разработчики локальных сетей привнесли в организацию работы пользователей много нового. Так, стало намного проще, чем в глобальных сетях, получать доступ к сетевым ресурсам – в локальной сети пользователю не приходится запоминать сложные идентификаторы разделяемых ресурсов. Для этих целей система предоставляет список ресурсов в удобной для восприятия форме, например в виде древовидной графической структуры ("дерева" ресурсов). Еще один прием, позволяющий оптимизировать работу в локальной сети, состоит в том, что после соединения с удаленным ресурсом пользователь получает возможность обращаться к нему с помощью тех же команд, которые он применял при работе с локальными ресурсами. Следствием (и в то же время движущей силой) такого прогресса стало появление огромного количества непрофессиональных пользователей, освобожденных от необходимости изучать специальные (и достаточно сложные) команды для сетевой работы.

Может возникнуть вопрос – почему все эти преимущества пользователи получили только с появлением локальных сетей? Главным образом, это связано с использованием в локальных сетях качественных кабельных линий связи, на которых даже сетевые адаптеры первого поколения обеспечивали скорость передачи данных до 10 Мбит/с. При небольшой протяженности, свойственной локальным сетям, стоимость таких линий связи была вполне приемлемой. Поэтому экономное расходование пропускной способности каналов – одна из основных задач, возложенных на технологии первых глобальных сетей – никогда не выходило на первый план при разработке протоколов локальных сетей. В таких условиях основным механизмом прозрачного доступа к ресурсам локальных сетей стали периодические широковещательные объявления серверов о своих ресурсах и услугах. На основании таких объявлений клиентские компьютеры составляли списки имеющихся в сети ресурсов и предоставляли их пользователю.

1.8. ЭВОЛЮЦИЯ СЕТЕВЫХ ОПЕРАЦИОННЫХ СИСТЕМ

Однако и "дружественный" интерфейс, и сетевые функции появились у операционных систем персональных компьютеров не сразу. Первая версия наиболее популярной операционной системы раннего этапа развития персональных компьютеров – MS-DOS компании Microsoft – не предоставляла таких возможностей. Недостающие функции для MS-DOS и подобных ей ОС компенсировались внешними программами, предоставлявшими пользователю удобный графический интерфейс (например, Norton Commander) или средства тонкого управления дисками (например, PC Tools). Наибольшее влияние на развитие программного обеспечения для персональных компьютеров оказала операционная среда Windows компании Microsoft, представлявшая собой надстройку над MS-DOS.

Вместе с версией MS-DOS 3.1 в 1984 году компания Microsoft выпустила продукт Microsoft Networks, который обычно называют MS-NET. Некоторые концепции, заложенные в MS-NET, такие как введение в структуру базовых сетевых компонентов – редиректора и сетевого сервера, успешно перешли в более поздние сетевые продукты Microsoft: LAN Manager, Windows for Workgroups, а затем и в Windows NT.

Иной путь выбрали разработчики Novell. Они изначально сделали ставку на создание операционной системы со встроенными сетевыми функциями и добились на этом пути больших успехов. Сетевые операционные системы NetWare производства Novell на долгое время стали эталоном производительности, надежности и защищенности для локальных сетей.

В 1987 году в результате совместных усилий Microsoft и IBM появилась первая многозадачная операционная система для персональных компьютеров с процессором Intel 80286, в полной мере использующая возможности защищенного режима – OS/2.

Сетевые разработки компаний Microsoft и IBM привели к появлению NetBIOS – очень популярного транспортного протокола и одновременно интерфейса прикладного программирования для локальных сетей, нашедшего применение практически во всех сетевых операционных системах для персональных компьютеров. Этот протокол и сегодня применяется для создания небольших локальных сетей.

Не очень удачная судьба OS/2 не позволила системам LAN Manager и LAN Server захватить заметную долю рынка, но принципы работы этих сетевых систем во многом нашли отражение в более успешной Microsoft Windows NT, содержащей встроенные сетевые компоненты (некоторые из них имеют приставку LM – от LAN Manager).

На персональные компьютеры устанавливались специально для них разработанные операционные системы, подобные MS-DOS, NetWare и OS/2, а также адаптировались существующие ОС. Появление процессоров Intel 80286 и особенно 80386 с поддержкой мультипрограммирования позволило перенести на платформу персональных компьютеров ОС Unix. Наиболее известной системой этого типа была версия Unix компании Santa Cruz Operation (SCO Unix).

В 90-е годы практически все операционные системы, занимающие заметное место на рынке, стали сетевыми. Сетевые функции сегодня встраиваются в ядро ОС и являются его неотъемлемой частью. Операционные системы получили средства для работы со всеми основными технологиями локальных (Ethernet, Fast Ethernet, Gigabit Ethernet, Token Ring, FDDI, ATM) и глобальных (X.25, frame relay, ISDN, ATM) сетей, а также средства для создания составных сетей (IP, IPX, AppleTalk, RIP, OSPF, NLSP). В операционных системах используются средства мультиплексирования нескольких стеков протоколов, что позволяет компьютерам поддерживать сетевую работу с разнородными клиентами и серверами. Появились специализированные ОС, предназначенные исключительно для выполнения коммуникационных задач. Например, сетевая операционная система IOS компании Cisco Systems, работающая в маршрутизаторах, организует в мультипрограммном режиме выполнение набора программ, каждая из которых реализует один из коммуникационных протоколов.

Во второй половине 90-х годов все производители операционных систем резко усилили поддержку средств работы с Internet (кроме производителей Unix-систем, в которых эта поддержка всегда была существенной). Кроме самого стека TCP/IP в комплект поставки начали включать утилиты, реализующие такие популярные сервисы Internet как telnet, ftp, DNS и Web. Влияние Internet проявилось и в том, что компьютер превратился из вычислительного устройства в средство коммуникаций с развитыми вычислительными возможностями.

На современном этапе развития операционных систем на передний план вышли средства обеспечения безопасности. Это обусловлено возросшей ценностью информации, обрабатываемой компьютерами, а также повышенным уровнем риска, связанного с передачей данных по сетям, особенно по общедоступным, таким, как Internet. Многие операционные системы обладают сегодня развитыми средствами защиты информации, основанными на шифровании данных, аутентификации и авторизации.

Современным операционным системам присуща *многоплатформенность*, то есть способность работать на компьютерах различного типа. Многие операционные системы имеют специальные версии для поддержки кластерных архитектур, обеспечивающих высокую производительность и отказоустойчивость. Исключение пока составляет ОС NetWare, все версии которой разработаны для платформы Intel, а реализация функций NetWare в виде оболочки для других ОС, например NetWare for AIX, успеха не имела.

В последние годы получила дальнейшее развитие тенденция повышения удобства работы с компьютером. Эффективность работы пользователя становится основным фактором, определяющим эффективность вычислительной системы в целом. Усилия человека не должны тратиться на настройку параметров вычислительного процесса, как это происходило в ОС предыдущих поколений. Например, в

системах пакетной обработки для мэйнфреймов каждый пользователь должен был с помощью языка управления заданиями определить большое количество параметров, относящихся к организации вычислительных процессов в компьютере. Так, для системы OS/360 язык управления заданиями JCL предусматривал возможность определения пользователем более 40 параметров, среди которых были приоритет задания, требования к основной памяти, предельное время выполнения задания, перечень используемых устройств ввода-вывода и режимы их работы.

Современная операционная система берет на себя выбор параметров операционной среды, с помощью различных адаптивных алгоритмов. Например, тайм-ауты в коммуникационных протоколах часто определяются в зависимости от условий работы сети. Распределение оперативной памяти между процессами осуществляется автоматически с помощью механизмов виртуальной памяти в зависимости от активности этих процессов и информации о частоте использования ими той или иной страницы. Мгновенные приоритеты процессов определяются динамически в зависимости от предыстории, включающей, например, время нахождения процесса в очереди, процент использования выделенного кванта времени, интенсивность ввода-вывода и т.п. Даже в процессе установки большинство ОС предлагают режим выбора параметров по умолчанию, который гарантирует пусть не оптимальное, но всегда приемлемое качество работы систем.

Постоянно повышается удобство интерактивной работы с компьютером путем включения в операционную систему развитых графических интерфейсов, использующих наряду с графикой звук и видео. Это особенно важно для превращения компьютера в терминал новой общедоступной сети, которой постепенно становится Internet, так как для массового пользователя терминал должен быть по простоте использования подобен телефонному аппарату. Пользовательский интерфейс операционной системы становится все более интеллектуальным, он направляет действия человека в типовых ситуациях и выполняет многие задачи автоматически.

Уровень удобства в работе с ресурсами, которые сегодня предоставляют пользователям, администраторам и разработчикам приложений операционные системы изолированных компьютеров, для сетевых операционных систем является только заманчивой перспективой. Пока же пользователи и администраторы сети тратят значительное время на попытки выяснить, где находится тот или иной ресурс, а разработчики сетевых приложений прилагают много усилий для определения местоположения данных и программных модулей в сети. Операционные системы будущего должны обеспечить высокий уровень прозрачности сетевых ресурсов, взяв на себя задачу организации распределенных вычислений, превратив сеть в виртуальный компьютер.

2. ОСНОВНЫЕ ЗАДАЧИ ПОСТРОЕНИЯ СЕТЕЙ

При создании вычислительных сетей разработчикам пришлось решать множество самых разных задач, связанных с кодированием и синхронизацией электрических (оптических) сигналов, выбором конфигурации физических и логических связей, разработкой схем адресации устройств, созданием различных способов коммутации, мультиплексированием и демультиплексированием потоков данных, совместным использованием передающей среды. В данной разделе сформулируем все эти задачи, причем в той последовательности, в которой они возникали в процессе развития и совершенствования сетевых технологий.

Начнем с наиболее простого случая непосредственного соединения двух устройств физическим каналом, такое соединение называется связью **"**точка-точка**" (**point-to-point**)**.

2.1. СВЯЗЬ КОМПЬЮТЕРА С ПЕРИФЕРИЙНЫМИ УСТРОЙСТВАМИ

Частным случаем связи "точка-точка" является соединение компьютера с периферийным устройством. Поскольку механизмы взаимодействия компьютеров в сети многое позаимствовали у схемы взаимодействия компьютера с периферийными устройствами, начнем рассматривать принципы работы сети с этого "досетевого" случая.

Для обмена данными компьютер и периферийное устройство (ПУ) оснащены внешними интерфейсами или портами (рис. 2.1). В данном случае к понятию "интерфейс" относятся:

- электрический разъем;
- набор проводов, соединяющих устройства;
- совокупность правил обмена информацией по этим проводам.

Со стороны компьютера логикой передачи сигналов на внешний интерфейс управляют:

- *контроллер ПУ* – аппаратный блок, часто реализуемый в виде отдельной платы;
- *драйвер ПУ* – программа, управляющая контроллером периферийного устройства.

Со стороны ПУ интерфейс чаще всего реализуется аппаратным устройством управления ПУ, хотя встречаются и программно-управляемые периферийные устройства.

Обмен данными между ПУ и компьютером, как правило, является двунаправленным. Так, например, даже принтер, который представляет собой устройство вывода информации, возвращает в компьютер данные о своем состоянии.

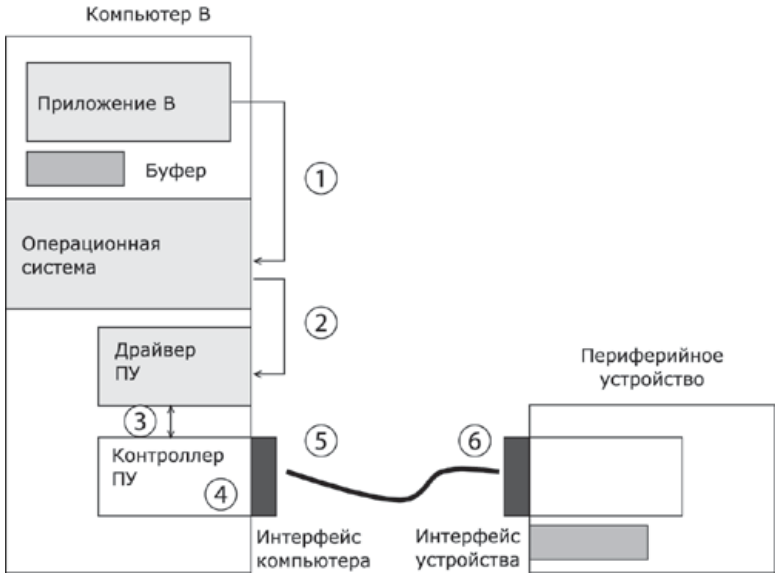

Рис. 2.1. Связь компьютера с периферийным устройством

Таким образом, по каналу, связывающему внешние интерфейсы, передается следующая информация:

– данные, поступающие от контроллера на ПУ, например байты текста, который нужно распечатать на бумаге;

– команды управления, которые контроллер передает на устройство управления ПУ; в ответ на них оно выполняет специальные действия, например переводит головку диска на соответствующую дорожку или же выталкивает из принтера лист бумаги;

– данные, возвращаемые устройством управления ПУ в ответ на запрос от контроллера, например данные о готовности к выполнению операции.

Рассмотрим последовательность действий, которые выполняются в том случае, когда некоторому приложению требуется напечатать текст на принтере. Со стороны компьютера в выполнении этой операции принимает участие, кроме уже названных контроллера, драйвера и приложения, еще один важнейший компонент – операционная система. Поскольку все операции ввода-вывода являются привилегированными, все приложения при выполнении операций с периферийными устройствами используют ОС как арбитра. Итак, последовательность действий такова:

1. Приложение обращается с запросом на выполнение операции печати к операционной системе. В запросе указываются: адрес данных в оперативной памяти, идентифицирующая информация принтера и операция, которую требуется выполнить (например, чтение или запись).

2. Получив запрос, операционная система анализирует его, решает, может ли он быть выполнен, и если решение положительное, то запускает соответствующий драйвер, передавая ему в качестве параметров адрес выводимых данных. Дальнейшие действия, относящиеся к операции ввода-вывода, со стороны компьютера реализуются совместно драйвером и контроллером принтера.

3. Драйвер передает команды и данные контроллеру, который помещает их в свой внутренний буфер. Пусть, например, драйвер загружает значение некоторого байта в буфер контроллера ПУ.

4. Контроллер перемещает данные из внутреннего буфера во внешний порт.

5. Контроллер начинает последовательно передавать биты в линию связи, представляя каждый бит соответствующим электрическим сигналом. Чтобы сообщить устройству управления принтера о том, что начинается передача байта, перед передачей первого бита данных контроллер формирует стартовый сигнал специфической формы, а после передачи последнего информационного бита – стоповый сигнал. Эти сигналы синхронизируют передачу байта. Кроме информационных бит, контроллер может передавать бит контроля четности для повышения достоверности обмена.

6. Устройство управления принтера, обнаружив на соответствующей линии стартовый бит, выполняет подготовительные действия и начинает принимать информационные биты, формируя из них байт в своем приемном буфере. Если передача сопровождается битом четности, то выполняется проверка корректности передачи: при правильно выполненной передаче в соответствующем регистре устройства управления принтера устанавливается признак завершения приема информации. Наконец, принятый байт обрабатывается принтером – выполняется соответствующая команда или печатается символ.

Обязанности между драйвером и контроллером могут распределяться по-разному, но чаще всего контроллер поддерживает набор простых команд, служащих для управления периферийным устройством, а на драйвер обычно возлагаются наиболее сложные функции реализации обмена. Например, контроллер принтера может поддерживать такие элементарные команды, как "Печать символа", "Перевод строки", "Возврат каретки" и т.п.

Драйвер же принтера с помощью этих команд реализует печать строк символов, разделение документа на страницы и другие более высокоуровневые операции (например, подсчет контрольной суммы последовательности передаваемых байтов, анализ состояния периферийного устройства, проверка правильности выполнения команды). Драйвер, задавая ту или иную последовательность команд, определяет тем самым логику работы периферийного устройства. Для одного и того же контроллера можно разработать различные драйверы, которые с помощью одного и того же набора доступных команд будут реализовывать разные алгоритмы управления одним и тем же ПУ.

Возможно распределение функций между драйвером и контроллером (УУ).

Функции, выполняемые драйвером:

- ведение очередей запросов;
- буферизация данных;
- подсчет контрольной суммы последовательности байтов;
- анализ состояния ПУ;
- загрузка очередного байта данных (или команды) в регистр контроллера;
- считывание байта данных или байта состояния ПУ из регистра контроллера.

Функции, выполняемые контроллером:

- преобразование байта из регистра (порта) в последовательность бит;
- передача каждого бита в линию связи;
- обрамление байта стартовым и стоповым битами – синхронизация;
- формирование бита четности;
- установка признака завершения приема/передачи байта.

2.2. СВЯЗЬ ДВУХ КОМПЬЮТЕРОВ

Предположим, что пользователь другого компьютера хотел бы распечатать текст. Сложность состоит в том, что к его компьютеру не подсоединен принтер и требуется воспользоваться тем принтером, который связан с другим компьютером (рис. 2.2).

Программа, работающая на одном компьютере, не может получить непосредственный доступ к ресурсам другого компьютера – его дискам, файлам, принтеру. Она может только "попросить" об этом другую программу, выполняемую на том компьютере, которому принадлежат эти ресурсы. Эти "просьбы" выражаются в виде сообщений, передаваемых по каналам связи между компьютерами. Такая организация печати называется удаленной.

Предположим, что мы связали компьютеры по кабелю через COM-порты, которые, как известно, реализуют интерфейс RS-232C (такое соединение часто называют нуль-модемным). Связь между компьютерами осуществляется аналогично связи компьютера с ПУ. Только теперь контроллеры и драйверы портов действуют с двух сторон. Вмес-

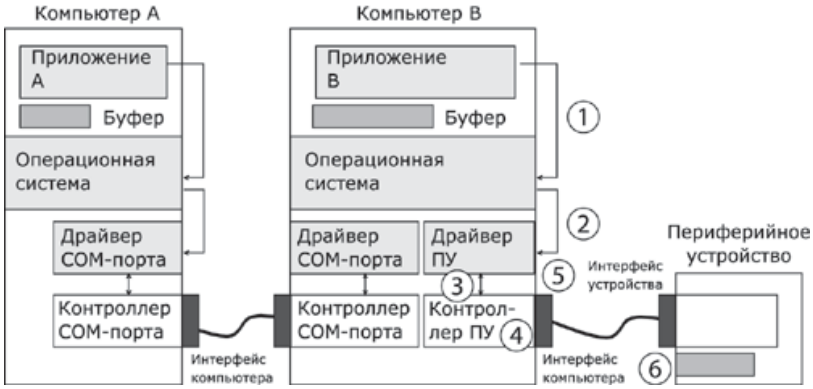

Рис. 2.2. Взаимодействие двух компьютеров

те они обеспечивают передачу по кабелю между компьютерами одного байта информации. (В "настоящих" локальных сетях подобные функции передачи данных в линию связи выполняются сетевыми адаптерами и их драйверами.)

Итак, механизм обмена байтами между двумя компьютерами определен. Теперь нужно договориться о правилах обмена сообщениями между приложениями А и В. Приложение В должно "уметь" расшифровать получаемую от приложения А информацию. Для этого программисты, разрабатывавшие приложения А и В, строго оговаривают форматы сообщений, которыми будут обмениваться приложения, и их семантику. Например, они могут договориться о том, что любое выполнение удаленной операции печати начинается с передачи сообщения, запрашивающего информацию о готовности приложения В; что в начале сообщения идет число, определяющее длину данных, предназначенных для печати; что признаком срочного завершения печати является определенная кодовая комбинация и т.п. Тем самым, как будет показано дальше, определяется протокол взаимодействия приложений.

Вернемся к последовательности действий, которые необходимо выполнить для распечатки текста на принтере "чужого" компьютера.

1. Приложение А формирует очередное сообщение (содержащее, например, строку, которую необходимо вывести на принтер) приложению В, помещает его в буфер оперативной памяти и обращается к ОС с запросом на передачу содержимого буфера на компьютер В.
2. ОС компьютера А обращается к драйверу COM-порта, который инициирует работу контроллера.
3. Действующие с обеих сторон пары драйверов и контроллеров COM-порта последовательно, байт за байтом, передают сообщение на компьютер В.
4. Драйвер компьютера В периодически выполняет проверку на наличие признака завершения приема, устанавливаемого контроллером при правильно выполненной передаче данных, и при его появлении считывает принятый байт из буфера контроллера в оперативную память, тем самым делая его доступным для программ компьютера В. В некоторых случаях драйвер вызывается асинхронно, по прерываниям от контроллера. Аналогично реализуется и передача байта в другую сторону – от компьютера В к компьютеру А.
5. Приложение В принимает сообщение, интерпретирует его, и в зависимости от того, что в нем содержится, формирует запрос к своей ОС на выполнение тех или иных действий с принтером. В нашем примере сообщение содержит указание на печать текста, поэтому ОС передает драйверу принтера запрос на печать строки.

Далее выполняются все действия 1–5, описывающие выполнение запроса приложения к ПУ в соответствии с рассмотренной ранее схемой "локальная ОС – драйвер ПУ – контроллер ПУ – устройство управления ПУ" (см. предыдущий раздел). В результате строка будет напечатана.

Рассмотрели последовательность работы системы при передаче только одного сообщения от приложения А к приложению В. Однако порядок взаимодействия этих двух приложений может предполагать неоднократный обмен сообщениями разного типа. Например, после успешной печати строки (в предыдущем примере) согласно правилам, приложение В должно послать сообщение-подтверждение. Это ответное сообщение приложение В помещает в буферную область оперативной памяти, а далее с помощью драйвера COM-порта передает его по каналу связи в компьютер А, где оно и попадает к приложению А.

2.3. КЛИЕНТ, РЕДИРЕКТОР И СЕРВЕР

Можно представить, что любая программа, которой потребуется печать на "чужом" принтере, должна включать в себя функции, подобные тем, которые выполняет приложение А. Но нагружать этими стандартными действиями каждое приложение – текстовые и графические редакторы, системы управления базами данных и другие приложения – не очень рационально (хотя существует большое количество программ, которые действительно самостоятельно решают все задачи по обмену данными между компьютерами, например Kermit – программа обмена файлами через COM-порты, реализованная для различных ОС, Norton Commander 3.0 с его функцией Link). Гораздо выгоднее создать специальный программный модуль, который (вместо приложения А) будет выполнять формирование сообщений-запросов к удаленной машине и прием результатов для всех приложений. Такой служебный модуль называется *клиентом*.

На стороне же компьютера В (на месте приложения В) должна работать другая специализированная программа – *сервер*, постоянно ожидающий прихода запросов на удаленный доступ к принтеру (или файлам, расположенным на диске) этого компьютера. Сервер, приняв запрос из сети, обращается к локальному ПУ, возможно, с участием локальной ОС.

Очень удобной и полезной функцией клиентской программы является способность отличить запрос к удаленному файлу от запроса к локальному файлу. Если клиентская программа умеет это делать, она сама распознает и перенаправляет (redirect) запрос к удаленной машине. Отсюда и название, часто используемое для клиентской части – *редиректор*. Иногда функции распознавания выделяются в особый программный модуль, в этом случае редиректором называют не всю клиентскую часть, а только этот модуль.

Программные клиент и сервер выполняют системные функции по обслуживанию запросов всех приложений компьютера А на удаленный доступ к файлам компьютера В. Чтобы приложения компьютера В могли пользоваться файлами компьютера А, описанную схему нужно симметрично дополнить клиентом для компьютера В и сервером для компьютера А. Схема взаимодействия клиента и сервера с приложениями и локальной операционной системой приведена на рис. 2.3.

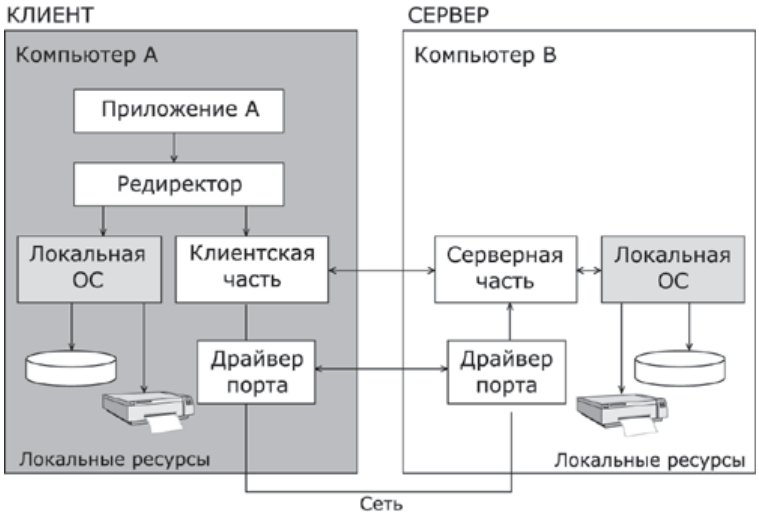

Рис. 2.3. Взаимодействие программных компонентов при связи двух компьютеров

Для того, чтобы компьютер мог работать в сети, его операционная система должна быть дополнена клиентским и/или серверным модулем, а также средствами передачи данных между компьютерами. В результате такого добавления операционная система компьютера становится *сетевой ОС*.

2.4. Задача физической передачи данных по линиям связи

Даже при рассмотрении простейшей сети, состоящей всего из двух машин, можно увидеть многие проблемы, присущие любой вычислительной сети, в том числе, связанные с *физической передачей* сигналов по линиям связи.

При соединении "точка-точка" на первый план выходит задача физической передачи данных по линиям связи. Эта задача среди прочего включает:

- *кодирование и модуляцию* данных;
- взаимную *синхронизацию* передатчика одного компьютера с приемником другого;

- подсчет *контрольной суммы* и передачу ее по линиям связи после каждого байта или после некоторого блока байтов.

В вычислительной технике для представления данных используется двоичный код. Представление данных в виде электрических или оптических сигналов называется кодированием. Существуют различные способы кодирования двоичных цифр 1 и 0, например потенциальный способ, при котором единице соответствует один уровень напряжения, а нулю – другой, или импульсный способ, когда для представления цифр используются импульсы различной или одной полярности.

Аналогичные подходы могут использоваться для кодирования данных и при их передаче между двумя компьютерами по линиям связи. Однако эти линии связи отличаются по своим электрическим характеристикам от тех, которые существуют внутри компьютера. Главное отличие внешних линий связи от внутренних состоит в их гораздо большей протяженности, а также в том, что они проходят вне экранированного корпуса по пространствам, зачастую подверженным воздействию сильных электромагнитных помех. Все это приводит к существенно большим искажениям прямоугольных импульсов (например, "заваливанию" фронтов), чем внутри компьютера. Поэтому при передаче данных внутри и вне компьютера не всегда можно использовать одни и те же скорости и способы кодирования.

В вычислительных сетях применяют как потенциальное, так и импульсное кодирование дискретных данных, а также специфический способ представления данных, который никогда не используется внутри компьютера, – модуляцию (рис. 2.4). При модуляции дискретная инфор-

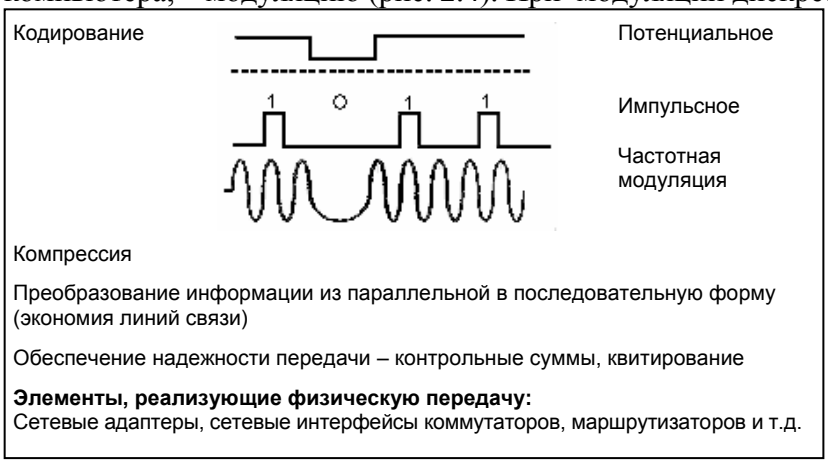

Рис. 2.4. Задачи физической передачи данных по линии связи

мация представляется синусоидальным сигналом той частоты, которую хорошо передает имеющаяся линия связи.

Потенциальное или импульсное кодирование применяется на каналах высокого качества, а модуляция на основе синусоидальных сигналов предпочтительнее в том случае, когда канал вносит сильные искажения в передаваемые сигналы. Обычно модуляция используется в глобальных сетях при передаче данных через аналоговые телефонные линии, которые были разработаны для передачи голоса в аналоговой форме и поэтому не очень подходят для непосредственной передачи импульсов.

На способ передачи сигналов влияет и количество проводов в линиях связи между компьютерами. Чтобы снизить стоимость линий связи в сетях, разработчики стараются сократить количество проводов и из-за этого используют не параллельную передачу всех бит одного байта или даже нескольких байт, как это делается внутри компьютера, а последовательную, побитную передачу, требующую всего одной пары проводов.

При передаче сигналов приходится еще решать проблему взаимной синхронизации передатчика одного компьютера с приемником другого. При организации взаимодействия модулей внутри компьютера она решается очень просто, так как в этом случае все модули синхронизируются от общего тактового генератора. Проблема синхронизации при связи компьютеров может решаться разными способами, как с помощью обмена специальными тактовыми синхроимпульсами по отдельной линии, так и посредством периодической синхронизации заранее обусловленными кодами или импульсами характерной формы, отличной от формы импульсов данных.

Несмотря на принятые меры (выбор соответствующей скорости обмена данными, линий связи с определенными характеристиками, способа синхронизации приемника и передатчика), существует вероятность искажения некоторых бит передаваемых данных. Для более надежной передачи данных часто используется стандартный прием – подсчет контрольной суммы и передача ее по линиям связи после каждого байта или после некоторого блока байтов. Часто в протокол обмена данными включается как

обязательный элемент сигнал-квитанция, которая подтверждает правильность приема данных и посылается от получателя отправителю.

В каждый сетевой интерфейс, будь то порт маршрутизатора, концентратора или коммутатора, встроены средства, в той или иной мере решающие задачу надежного обмена двоичными сигналами, представленными соответствующими электромагнитными сигналами. Некоторые сетевые устройства, такие, как *модемы* и *сетевые адаптеры*, специализируются на физической передаче данных. Модемы выполняют в глобальных сетях модуляцию и демодуляцию дискретных сигналов, синхронизируют передачу электромагнитных сигналов по линиям связи, проверяют правильность передачи по контрольной сумме и могут выполнять некоторые другие операции. Сетевые адаптеры рассчитаны, как правило, на работу с определенной передающей средой – коаксиальным кабелем, витой парой, оптоволокном и т.п. Каждый тип передающей среды обладает определенными электрическими характеристиками, влияющими на способ использования данной среды, и определяет скорость передачи сигналов, способ их кодирования и некоторые другие параметры.

До сих пор рассматривали сеть, состоящую всего из двух машин. При объединении в сеть большего количества компьютеров возникает целый комплекс новых проблем.

3. ТОПОЛОГИЯ ФИЗИЧЕСКИХ СВЯЗЕЙ

3.1. ТИПЫ КОНФИГУРАЦИЙ СВЯЗИ КОМПЬЮТЕРОВ

Как только компьютеров становится больше двух, возникает проблема выбора *конфигурации физических связей* или *топологии*. Под топологией сети понимается конфигурация графа, вершинам которого соответствуют конечные узлы сети (например, компьютеры) и коммуникационное оборудование (например, маршрутизаторы), а ребрам – электрические и информационные связи между ними.

Число возможных конфигураций резко возрастает при увеличении числа связываемых устройств. Так, если три компьютера мы можем связать двумя способами, то для четырех компьютеров (рис. 3.1) можно предложить уже шесть топологически различных конфигураций (при условии неразличимости компьютеров).

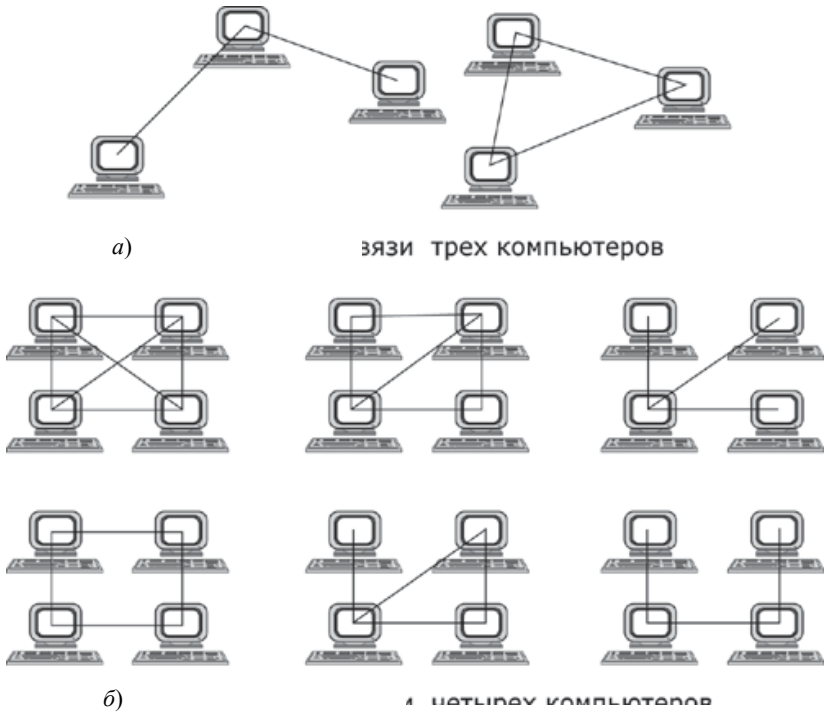

Рис. 3.1. Варианты связи компьютеров:
а – трех компьютеров; *б* – четырех компьютеров

Можно соединять каждый компьютер с каждым или же связывать их последовательно, предполагая, что они будут общаться, передавая друг другу сообщения "транзитом". При этом транзитные узлы должны быть оснащены специальными средствами, позволяющими выполнять эту специфическую посредническую операцию. В роли транзитного узла может выступать как универсальный компьютер, так и специализированное устройство.

От выбора топологии связей зависят многие характеристики сети. Например, наличие между узлами нескольких путей повышает надежность сети и делает возможной балансировку загрузки отдельных каналов. Простота присоединения новых узлов, свойственная некоторым топологиям, делает сеть легко расширяемой. Экономические соображения часто приводят к выбору топологий, для которых характерна минимальная суммарная длина линий связи.

Среди множества возможных конфигураций различают *полносвязные* и *неполносвязные* (см. рис. 3.2):

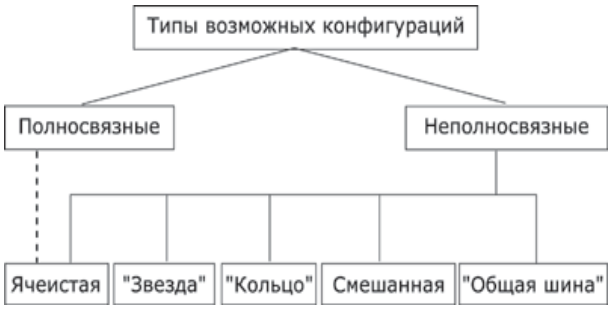

Рис. 3.2. Типы конфигураций

Полносвязная топология (рис. 3.3) соответствует сети, в которой каждый компьютер непосредственно связан со всеми остальными. Несмотря на логическую простоту, этот вариант громоздкий и неэффективный. Действительно, каждый компьютер в сети должен иметь большое количество коммуникационных портов, достаточное для связи с каждым из остальных компьютеров. Для каждой пары компьютеров должна быть выделена отдельная физическая линия связи. (В некоторых случаях даже две, если невозможно использование этой линии для двусторонней передачи.) Полносвязные топологии в крупных сетях применяются редко, так как для связи N узлов требуется N(N-1)/2 физических дуплексных линий связи, то есть имеет место квадратическая зависимость. Чаще этот вид топологии используется в многомашинных комплексах или в сетях, объединяющих небольшое количество компьютеров.

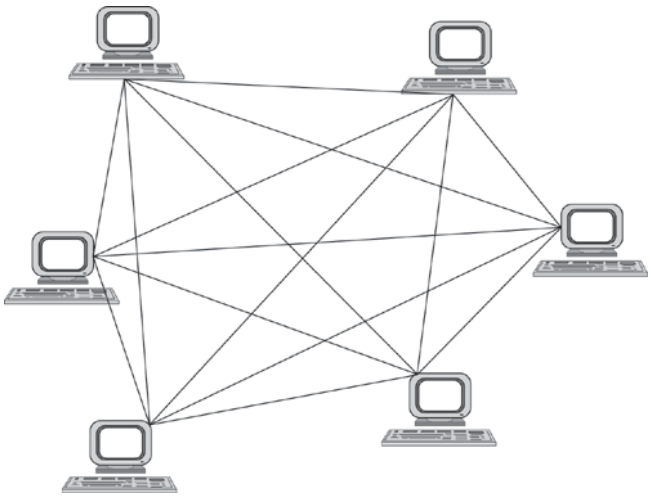

Рис. 3.3. Полносвязная конфигурация

Все другие варианты основаны на неполносвязных топологиях, когда для обмена данными между двумя компьютерами может потребоваться промежуточная передача данных через другие узлы сети.

Ячеистая топология (mesh) получается из полносвязной путем удаления некоторых возможных связей. Ячеистая топология допускает соединение большого количества компьютеров и характерна для крупных сетей (рис. 3.4).

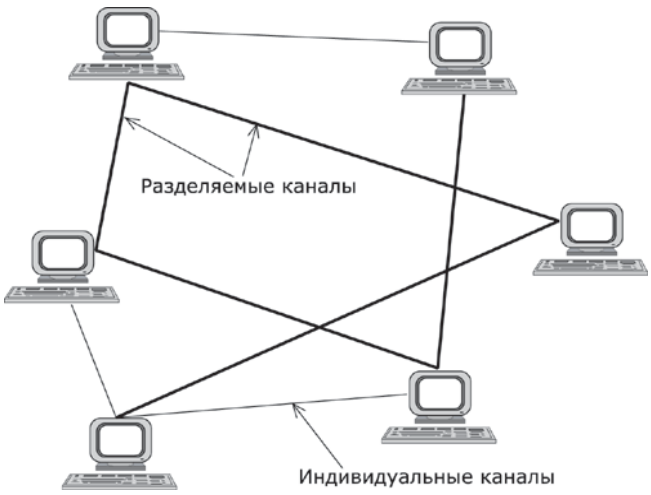

Рис. 3.4. Ячеистая топология

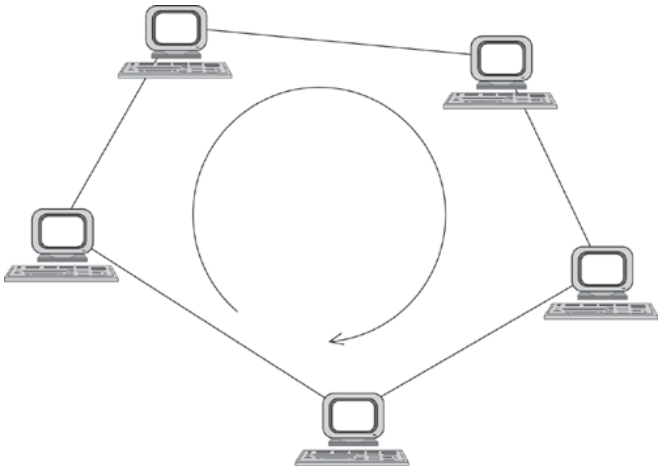

Рис. 3.5. Топология "кольцо"

В сетях с кольцевой конфигурацией (рис. 3.5) данные передаются по кольцу от одного компьютера к другому. Главное достоинство "кольца" в том, что оно по своей природе обладает свойством резервирования связей. Действительно, любая пара узлов соединена здесь двумя путями – по часовой стрелке и против. "Кольцо" представляет собой очень удобную конфигурацию и для организации обратной связи

– данные, сделав полный оборот, возвращаются к узлу-источнику. Поэтому отправитель в указанном случае может контролировать процесс доставки данных адресату. Часто это свойство "кольца" используется для тестирования связности сети и поиска узла, работающего некорректно. В то же время в сетях с кольцевой топологией необходимо принимать специальные меры, чтобы в случае выхода из строя или отключения какой-либо станции не прерывался канал связи между остальными станциями "кольца".

Топология "звезда" (рис. 3.6) образуется в том случае, когда каждый компьютер с помощью отдельного кабеля подключается к общему центральному устройству, называемому концентратором. В функции концентратора входит направление передаваемой компьютером информации одному или всем остальным компьютерам сети. В роли концентратора может выступать как компьютер, так и специализированное устройство, такое как многовходовый повторитель, коммутатор или маршрутизатор. К недостаткам топологии типа "звезда" относится более высокая стоимость сетевого оборудования, связанная с необходимостью приобретения специализированного центрального устройства. Кроме того, возможности наращивания количества узлов в сети ограничиваются количеством портов концентратора.

Рис. 3.6. Топология "звезда"

Иногда имеет смысл строить сеть с использованием нескольких концентраторов, иерархически соединенных между собой связями типа "звезда" (рис. 3.7). Получаемую в результате структуру называют также *деревом*. В настоящее время дерево является самым распространенным типом топологии связей, как в локальных, так и в глобальных сетях.

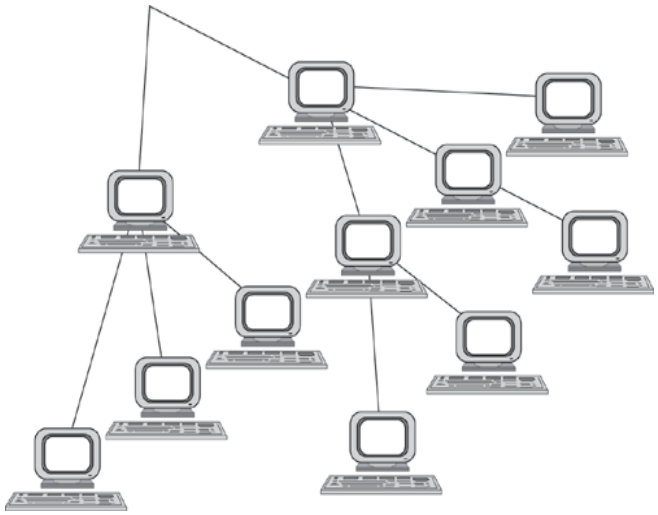

Рис. 3.7. Топология "иерархическая звезда" или "дерево"

Рис. 3.8. Топология "общая шина"

Особым частным случаем конфигурации "звезда" является конфигурация "общая шина" (рис. 3.8). Здесь в роли центрального элемента выступает пассивный кабель, к которому по схеме "монтажного ИЛИ" подключается несколько компьютеров (такую же топологию имеют многие сети, использующие беспроводную связь – роль общей шины здесь играет общая радиосреда). Передаваемая информация распространяется по кабелю и доступна одновременно всем присоединенным к нему компьютерам.

Основными преимуществами такой схемы являются низкая стоимость и простота наращивания, то есть присоединения новых узлов к сети.

Самым серьезным недостатком "общей шины" является ее недостаточная надежность: любой дефект кабеля или какого-нибудь из многочисленных разъемов полностью парализует всю сеть. Другой недостаток "общей шины" – невысокая производительность, так как при таком способе подключения в каждый момент времени только один компьютер может передавать данные по сети, поэтому пропускная способность канала связи всегда делится между всеми узлами сети. До недавнего времени "общая шина" являлась одной из самых популярных топологий для локальных сетей.

В то время как небольшие сети, как правило, имеют типовую топологию – "звезда", "кольцо" или "общая шина", для крупных сетей характерно наличие произвольных связей между компьютерами. В таких сетях можно выделить отдельные произвольно связанные фрагменты (подсети), имеющие типовую топологию, поэтому их называют сетями со *смешанной* топологией (рис. 3.9).

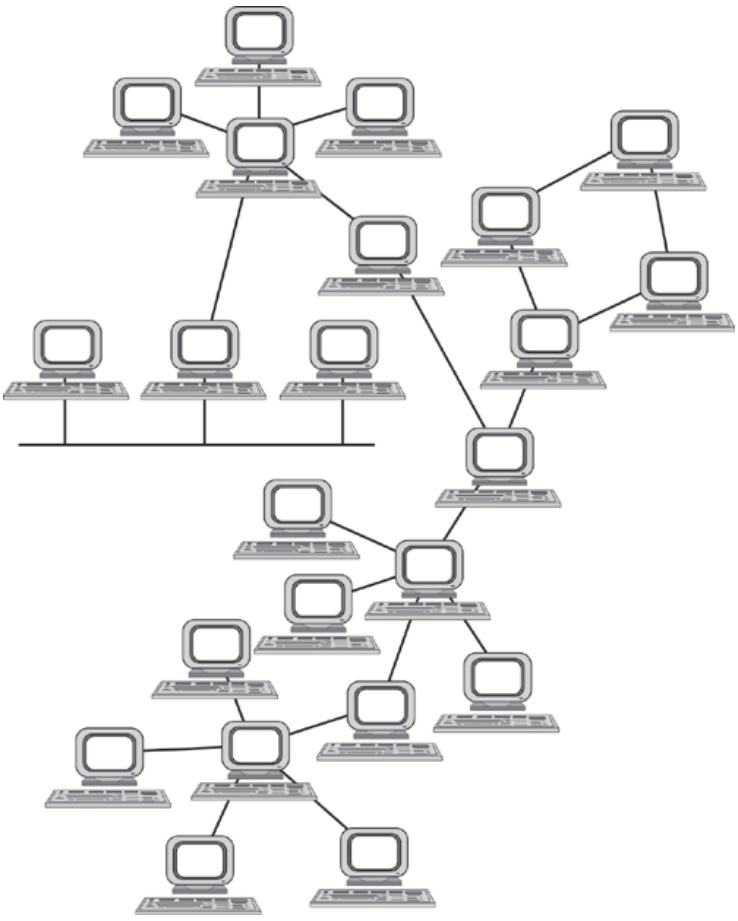

Рис. 3.9. Смешанная топология

3.2. КАБЕЛИ ВЫЧИСЛИТЕЛЬНЫХ СЕТЕЙ

Выбор кабельной подсистемы диктуется выбранной топологией и типом сети. Требуемые же по стандарту физические характеристики кабеля закладываются при его изготовлении, о чем и свидетельствуют нанесенные на кабель маркировки. В результате, сегодня практически все сети проектируются

на базе UTP и волоконно-оптических кабелей, коаксиальный кабель применяют лишь в исключительных случаях и то, как правило, при организации низкоскоростных стеков в монтажных шкафах.

В проекты локальных вычислительных сетей (стандартных) закладываются на сегодня всего три вида кабелей:

1) коаксиальный (двух типов):
   - тонкий коаксиальный кабель (thin coaxial cable);
   - толстый коаксиальный кабель (thick coaxial cable).
2) витая пара (двух основных типов):
   - неэкранированная витая пара (unshielded twisted pair – UTP);
   - экранированная витая пара (shielded twisted pair – STP).
3) волоконно-оптический кабель (двух типов):
   - многомодовый кабель (fiber optic cable multimode);
   - одномодовый кабель (fiber optic cable single mode).

И хотя общая номенклатура всех этих кабелей у многих производителей составляет даже не сотни, а тысячи наименований, выбирать кабель, как правило, приходится исходя не из характеристик конкретной марки, а из правил применения, что существенно облегчает жизнь проектировщику кабельной подсистемы ЛВС.

При проектировании и монтаже ЛВС, как указывалось выше, в качестве стандартных систем передачи данных можно использовать довольно ограниченную номенклатуру кабелей: кабель с витыми парами (UTP-кабель) категорий 3, 4 или 5 с различными типами экранов или без них (STP – экранирование медной оплеткой, FTP – экранирование фольгой, SFTP – экранирование медной оплеткой и фольгой), тонкий коаксиальный кабель (RG-58) с разным исполнением центральной жилы (RG-58/U – сплошная медная жила, RG-58A/U – многожильный, RG-58C/U – специальное /военное/ исполнение кабеля RG-58A/U), толстый коаксиальный кабель (thick coaxial cable) и волоконно-оптический кабель (fiber optic cable single mode-одномодовый multimode-многомодовый). При этом каждый вид кабельной подсистемы накладывает те или иные ограничения на проект сети.

Максимальная длина сегмента:

- у кабеля с витыми парами – 100 м;
- у тонкого коаксиального кабеля – 185 м;
- у толстого коаксиального кабеля – 500 м;
- у многомодового (mm) оптоволоконного кабеля – 1000 м;
- у одномодового (sm) оптоволоконного кабеля (с применением специальных средств до 40 – 70…90 км) – 2000 м.

Количество узлов на сегменте:

- у кабеля с витыми парами – 2;
- у тонкого коаксиального кабеля – 30;
- у толстого коаксиального кабеля – 100;
- у оптоволоконного кабеля – 2.

Возможность работы на скоростях выше 10 Mbit/sec:

- у кабеля с витыми парами и волоконно-оптического кабеля – да;
- у коаксиальных кабелей – нет.

Правила противопожарной безопасности делят кабели на две категории: общего применения и пленумные (разрешенные для прокладки в вентиляционных шахтах). Это деление осуществляется, исходя из материалов, применяемых при изготовлении кабелей. Наиболее распространенные при изготовлении кабелей пластики на базе поливинилхлорида (PVC). При горении они выделяют ядовитые газы. Поэтому PVC-кабели запрещены для прокладки в вентиляционных шахтах. В пленумных пространствах обычно применяются кабели с изоляцией на основе тефлона.

3.2.1. Основные эксплуатационные характеристики кабелей на витой паре

Все кабели должны иметь витые пары проводов, применение кабелей с несвитыми попарно проводами не допускается. Это относится даже к коротким отрезкам плоского кабеля. При использовании экранированных кабелей на витой паре, сегменты последних рекомендуется заземлять на одном (и только на одном!) конце. На практике это удобнее производить на конце, подключенном к концентратору:

- минимальный радиус изгиба – 5 см;
- температура при работе и хранении:
– для кабеля в поливинилхлоридной оболочке –35...+60 °С;
– для кабеля в тефлоновой оболочке –55...+200 °С;
- температура при монтаже:
– для кабеля в поливинилхлоридной оболочке –20...+60 °С;
– для кабеля в тефлоновой оболочке –35...+200 °С;
- относительная влажность:
– для кабеля в поливинилхлоридной оболочке, допускается случайная конденсация –0...+100 %;
– для кабеля в тефлоновой оболочке – не реагирует на влажность, конденсацию и водяные брызги;
- возможность применения на открытом воздухе:
– для кабеля в поливинилхлоридной оболочке – запрещено;
– для кабеля в тефлоновой оболочке – разрешено;
- запрещено применение тонкого коаксиального кабеля для прокладки на открытом воздухе между двумя не связанными друг с другом зданиями (между зданиями, не имеющими общего контура заземления).

При установке новой сети целесообразно применять кабель с витыми парами в рабочей группе. Оптоволоконные кабели – на длинных магистралях и для связи между зданиями. Тонкие коаксиальные кабели наиболее оправдано применять для организации низкоскоростых магистралей внутри монтажных шкафов (смотрите материал "Сложившаяся практика проектирования локальных сетей"). Кабели на витой паре и оптоволоконные кабели позволяют модернизировать сеть, переводя ее с 10 на 100 Mbit-ные технологии.

Наиболее "подвижной" частью любой ЛВС являются подсистемы рабочей группы. Добавление новых пользователей, перемещение рабочих мест и их аннулирование, повреждения кабеля в рамках рабочей группы происходят гораздо чаще, чем изменения в магистральных каналах. Именно поэтому UTP-кабели наиболее удобны для организации подсистем рабочих групп.

На длинных магистралях безусловно наиболее предпочтительно оптоволокно, ибо он обеспечивает наибольшую допустимую длину сегмента, высокую безопасность и помехозащищенность.

Если заказчик вдруг, неоправданно с Вашей точки зрения, настаивает на применении других, более дешевых кабелей или не хочет принимать Ваши рекомендации по вопросам будущего расширения сети, попробуйте объяснить ему, что сам кабель сравнительно дешев, а его установка обходится весьма дорого. Когда приходится прокладывать кабель внутри стен, под полом или над потолком, намного дешевле заложить сразу дополнительные кабели, чтобы потом, спустя несколько месяцев, возвращаться к этим работам и снова прокладывать кабель по старым трассам.

Чтобы не иметь проблем с кабельной подсистемой, при ее проектировании можно воспользоваться следующими правилами (рекомендации даны для применения UTP-кабелей):

- если это сеть здания офисного типа (например, банк или собственно офисное здание), закладывайте один UTP кабель на каждые 3…4 м^2 помещения. Рабочие места в зданиях такого типа подвержены наиболее частым переездам и очень плотному оснащению средствами вычислительной и оргтехники;
- если это сеть обычной фирмы или предприятия, удвойте потребность в средствах вычислительной техники, которую заявил Вам Заказчик;
- выполнив монтаж кабельной подсистемы обязательно проведите ее сертификацию на соответствие требованиям пятой категории (каждый линк и патч-корд). Даже если применяются качественные компоненты, факторы монтажа и окружающих условий могли вызвать ухудшение рабочих характеристик. Распечатайте и сохраните результаты испытаний;
- используйте соответствующие LAN-тестеры для проверки работоспособности соединительных кабелей.

Соблюдение этих правил позволит избежать проблем с расширением кабельной сети при переходах на новые технологии как в рамках собственно ЛВС, так и в телефонных коммуникациях.

Для подсистем на базе тонких коаксиальных кабелей такие рекомендации выработать нельзя, так как в таких подсистемах необходимо стараться решить другую задачу – минимизировать количество рабочих мест. Вообще говоря, тонкий коаксиальный кабель не рекомендуется для сетей рабочей группы, хотя проблема при его использовании заключается не собственно в кабеле. Дело в том, что проводка тонкого коаксиального кабеля выполняется открытой и пользователи имеют к ней доступ. Нередко пользователь некорректно отключает кабель, разрушая целостность кабельного сегмента. При этом выходит из строя вся сеть, может нарушиться работа сетевого программного обеспечения. К этим же по-

следствиям приводит снятие терминатора с конца кабельного сегмента, применение отрезков кабеля с другим волновым сопротивлением. По этим причинам целесообразно применять тонкий коаксиальный кабель только в защищенных от несанкционированного доступа местах, например в монтажном шкафу. Кроме того, шинная топология сетей на тонком коаксиальном кабеле затрудняет диагностирование, так как кабель является общим для множества узлов. Неисправность может быть вызвана любым узлом, любым отрезком кабеля или любым терминатором. Отыскать неисправность в таких сетях обычно довольно сложно.

3.2.2. Проблемы монтажа кабельной системы

В дополнение можно отметить, что управление сетью наиболее удобно на топологиях, поддерживаемых UTP-кабелем, а толстый коаксиальный кабель на наших территориях применяется настолько редко, что о возможности его применения проектировщики начинают забывать (хотя случаются ситуации, где его применение приводит к красивым техническим решениям). В этом отношении интересен североамериканский регион, где объемы продаж толстого коаксиального кабеля и трансиверов к нему довольно велики по настоящее время. Наиболее подходящая область применения UTP-кабелей – кабельные подсистемы рабочей группы, горизонтальные подсистемы зданий и вертикальные подсистемы (при использовании STP-кабеля). Тонкий коаксиальный кабель целесообразно использовать для организации магистралей в монтажных шкафах, рабочих групп в помещениях с жесткой привязкой рабочих мест, низкоскоростных вертикальных кабельных подсистем. Оптоволоконный кабель – лучшее решение для организации скоростной среды передачи данных вертикальной подсистемы, магистрали между коммутационными узлами и между зданиями (административная и базовая подсистемы). Толстый коаксиальный кабель сегодня находит применение только в частных случаях: для организации низкоскоростных магистралей между соседними зданиями (до 500 м). При этом его применение нередко определяется тем, что кабель "уже есть" или даже "ранее проложен для иных целей".

При монтаже любой кабельной подсистемы в любом здании приходится сталкиваться с огромным количеством проблем. Одна из причин – достаточно высокая (для того, чтобы создать проблемы) насыщенность зданий целой системой кабельных и проводных сетей: телефонные, телевизионные, системы пожарной и охранной сигнализации, локальные вычислительные сети компьютерных систем, системы электрообеспечения и т.п. кабельные коммуникации зачастую просто опутывают все помещения. Так называемые "интеллектуальные здания" у нас пока практически не строятся. Поэтому при проведении работ по монтажу компьютерных сетей в такого рода зданиях приходится решать следующие проблемы:

- ранее установленные локальные сети независимы и, как правило, работают на граничных длинах кабельных коммуникаций;
- обслуживающий персонал любой кабельной подсистемы здания (пожарной или компьютерной) считает свою подсистему главной и не принимает во внимание Ваши требования;
- малейшие изменения в архитектуре любой сети (например компьютерной или сети электроснабжения) приводят к затратам не только на дополнительные материалы, но и на проведение изменений в действующей части;
- заложенные при строительстве коммуникации полностью забиты как действующим, так и бесхозным кабелем, но освободить их от неиспользуемых кабельных систем невозможно без повреждения работающих сетей, а использовать бесхозные нельзя из-за множественных повреждений.

Решить эти проблемы в комплексе возможно только в том случае, если требовать, чтобы кабельные системы служили длительные периоды времени, не претерпевая кардинальных изменений, допуская при этом простое расширение. Но надо отдавать себе отчет в том, что это возможно лишь при капитальных затратах на внутренние кабельные системы здания. Километры кабеля, прибитого по плинтусам в коридорах здания, – это не система связи, способная надежно просуществовать несколько лет. Строго говоря, для решения проблем связанных с кабельными коммуникациями, необходимо плотное сотрудничество с проектно-конструкторскими организациями, так как специалисты, выполняющие эти работы, не знают что такое компьютерная сеть, и СНИП на проектирование кабельных подсистем ЛВС по-моему пока тоже не существует.

В мире несколько фирм специализируются на производстве так называемых структурированных систем монтажа. Наиболее известные из них AT&T с системой SYSTIMAX SCS, Digital – DEC Connect, AMP – NET Connect, а также Legrand, Panduit, Hubbell и др. предлагают такое количество готовых стандартных решений, такой набор кабельной фурнитуры, что проблем с монтажом и обслуживанием кабельного хозяйства, на наш взгляд, возникнуть не может. В состав структурированных кабельных систем вхо-

дят специальные короба разного сечения для укладки кабеля, фурнитура крепления, розетки (компьютерные, телефонные, электропитания), монтажные шкафы, кроссировочные или патч-панели, заделанные на концах коаксиальные, UTP и волоконно-оптические кабели разной длины. При этом топология кабельной системы собирается только на кроссировочной панели, позволяя организовывать в пределах одной кросс-панели несколько различных топологий локальных сетей без изменения физической конфигурации кабелей.

При относительно высокой начальной стоимости структурированные кабельные системы оправдывают капиталовложения за счет:

- длительного использования;
- возможности одновременного использования разных протоколов и сред передачи данных;
- модульности и возможности внесения изменений, а также наращивания мощности без влияния на существующие сети;
- обеспечения одновременного и быстрого доступа ко всем системам, проложенным в кабельных каналах;
- независимости от поставщика сетевого оборудования;
- являясь единой сетью, позволяют создавать независимые участки сети;
- использования ранее установленного оборудования;
- независимости от изменений в информационных технологиях;
- обеспечения зрительного восприятия разделения кабельных подсистем по функциональному признаку.

Структурированные кабельные системы – это реализация модульного представления о кабельных системах связи, рассматривающая последние в виде набора подсистем. Для того чтобы проектирование проистекало менее болезненно, а главное, для того чтобы в процессе эксплуатации было несложно модернизировать, расширить или даже перепрофилировать кабельную подсистему, ее желательно рассматривать в виде нескольких стандартизованных компонент – подсистем.

СКС выделяют три таких подсистемы: горизонтальную, вертикальную и кампус (базовую подсистему – магистраль между зданиями).

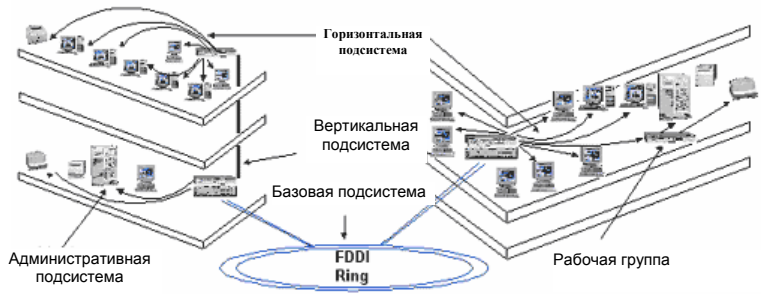

Рис. 3.10. Кабельные подсистемы на примере сети масштаба предприятия

Можно дополнить список подсистем СКС еще подсистемами рабочей группы и административной подсистемой. Подсистемы рабочей группы не всегда совпадают с горизонтальной подсистемой, особенно на развивающихся объектах. А на проектирование административной подсистемы накладывают свою специфику некоторые аппаратные комплексы по дистанционному управлению, разграничению доступа, безопасности и т.п., а также "взгляды на жизнь" ряда системных администраторов и руководителей, выпадающие из общей идеологии проектирования кабельных подсистем в рамках конкретного объекта (рис. 3.10). Практика показывает, что если не вычленить эти две подсистемы из общего проекта, то процесс принятия решения Заказчиком почти наверняка затянется.

3.2.3. Подсистема рабочей группы

Подсистема рабочей группы – это функционально-территориальная подсистема. Как правило, пользователь начинает думать о локальной вычислительной сети уже имея рабочие места, оснащенные компьютерами. Очень часто при этом некоторые компьютеры оказываются сопряженными или друг с дру-

гом, или с какими-то устройствами (обычно приборами, принтерами и модемами коллективного использования). То есть пользователь перед началом выполнения работ по проектированию ЛВС уже имеет кабельную подсистему той или иной степени сложности. Эту подсистему можно сохранить, если она в достаточной степени развита, или заменить на более приспособленную для решения задач данной рабочей группы. При необходимости сохранения старого кабельного хозяйства и включения его в состав новой ЛВС, целесообразно использовать кабельную подсистему, построенную на базе витой пары, так как среди выпускаемого промышленностью оборудования для витой пары есть полный спектр переходников с данного типа соединителя.

3.2.4. Горизонтальная подсистема

Горизонтальная подсистема – это территориальная подсистема. Обычно основной объем работ по прокладкам кабеля приходится на нее. Подсистема рабочей группы и административная подсистема, как правило, являются ее составными частями. В зависимости от характеристик объекта, на котором она устанавливается (производственный цех, этаж административного здания, спортивный стадион, морской порт, выставочный павильон и т.п.), эту подсистему приходится проектировать на оптоволокне, защищенной или незащищенной витой паре, коаксиальном кабеле. Однако, в последнее время, для этих целей редко используется коаксиальный кабель. Обычно применяют витую пару или волоконно-оптический кабель.

В последнее время все чаще принимается решение о применении в горизонтальных подсистемах оборудования, работающего со скоростью 100 Мбит/с. В тех же случаях, когда в ближайшей перспективе нет смысла в использовании сетевого оборудования с пропускной способностью выше 10 Мбит/с (оборудование 3-й категории), но есть перспектива развития сети, желательно сразу установить кабельную систему, способную работать со скоростью 100 Мбит/с (5-й категории). Это позволит, во-первых, немного приподнять общую производительность сети, благодаря уменьшению количества коллизий, связанных с чистотой каналов связи (кабели 5-й категории значительно "чище"), а во-вторых, и это самое главное, при дальнейшем развитии сети (переходе на оборудование 5-й категории) не придется производить никаких работ, связанных с заменой кабельного хозяйства.

Однако, для того чтобы кабельная подсистема 5-й категории, собранная на базе четырехпарных неэкранированных витых парах (а именно, UTP кабель, как правило, применяется в данных подсистемах), работала надежно, необходимо соблюдать определенные правила:

- все четыре пары кабеля имеют цветовую маркировку, с помощью которой различаются номера пар проводов. Помните о том, что существуют два основных стандарта распределения пар проводов по контактам разъемов RJ45: EIA-T568A и EIA-T568B. Существуют еще внутрифирменные стандарты для работы с определенными марками кабелей и коммутационного оборудования, но правила применения данных видов кабельной продукции описываются в сопроводительных документах. По стандарту EIA-T568A пары распределяются как показано на рис 3.11 и 3.12;
- обращать внимание на упаковочные листы к соединителям, некоторые фирмы (например Hubbell Premise Wiring) выпускают соединители с отличным от приведенного выше распределением пар;

Рис. 3.11. Соответствие цветовых маркировок парам проводников

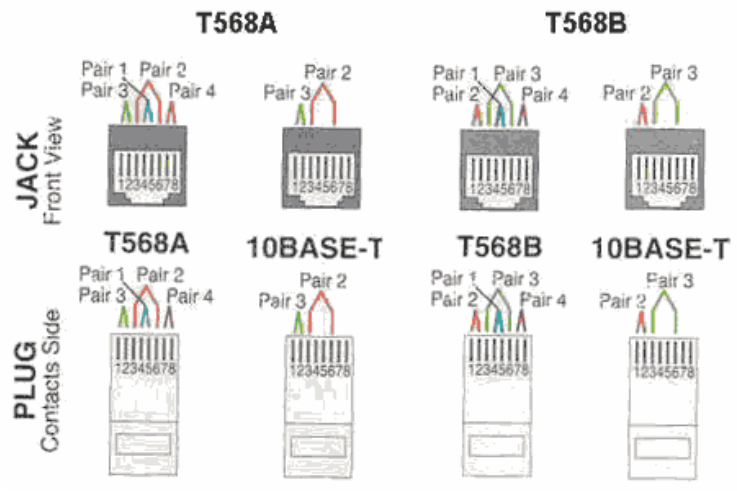

Рис. 3.12. Расположение пар и контактов на разъемах по стандартам T568A, T568B и Ethernet

- в пределах одной горизонтальной подсистемы использовать кабель одной марки одного и того же производителя;
- вся подсистема должна содержать изделия только 5-й категории (включая патч-панели, розетки и разъемы);
- горизонтальные кабели должны иметь длину порядка 90 м (стандарт IEEE 802.3 запрещает применение кабеля длиной более 90 м);
- соединительные кабели (кабели, прокладываемые от розетки до сетевого адаптера компьютера) не должны иметь длину более 10 м;
- общая длина горизонтального и соединительного кабелей не должна превышать 100 м;
- расплетение пар при их заделке допускается не более чем на 1/2 дюйма (12,7 мм);
- общее количество соединителей в горизонтальной проводке не должно превышать четырех устройств.

Обратите внимание на то, что номера пар в стандартах 568A и 568B меняют свое месторасположение и даже цвет, но при этом "информационная" принадлежность контактов остается прежней.

3.2.5. Вертикальная подсистема

Вертикальные подсистемы – территориальные подсистемы, служащие для подключения горизонтальных подсистем друг к другу. Обычно реализуются на базе коаксиального кабеля, защищенной витой пары или волоконно-оптического кабеля.

3.2.6. Административная подсистема

Административную кабельную подсистему, как правило, не выделяют в виде самостоятельной структуры. С одной стороны, это правильно, но часто ее желательно обозначить перед Заказчиком как отдельную структуру. Административная подсистема кабельного монтажа – это функциональная подсистема, связывающая подсистемы рабочих групп и горизонтальные подсистемы в единое целое. Она должна обеспечивать возможность установления резервных связей, подключение дополнительных рабочих мест и других подсистем. Нередко в рамках административной подсистемы требуется поддержка автономной системы энергоснабжения, голосовой и видео-связи. Одно из основных требований к административной подсистеме – гибкость и возможность увеличения мощности.

3.2.7. Базовая подсистема (кампус)

Базовые подсистемы служат для объединения вертикальных или административных подсистем друг с другом. В этом случае наиболее оправдано применение оптоволокна. В настоящее время на оптоволокне Ethernet работает с скоростями 10 Мбит/с и 100 Мбит/с, ожидается появление оборудования со скоростью 660 Мбит/с (теоретическая пропускная способность оптических кабелей на сегодня оценивается цифрой 200 Гбит/с). Многие компании используют для организации базовых подсистем оборудо-

вание, поддерживающее FDDI стандарт – волоконный распределенный интерфейс данных, имеющий производительность 100 Мбит/с.

Предприятия, выпускающие оборудование для ЛВС, поддержали идеи организации сетей на базе структурированных систем монтажа. Более того, работая над проблемой объединения между собой разных типов кабельных сетей, они выработали универсальный подход для решения этой проблемы – интеллектуальный модульный концентратор (Intelligent Hub). Этот вид оборудования выпускается в виде блока со сменными модулями, обеспечивающими связи со всеми типами кабельных систем. Большинство проектировщиков ЛВС давно пришли к мысли, что Hub-технологии и структурированный монтаж должны стать составной частью обычной рядовой офисной сети.

4. КОММУТАЦИОННОЕ ОБОРУДОВАНИЕ

В сетях с небольшим (10–30) количеством компьютеров чаще всего используется одна из типовых *топологий* – "общая шина", "кольцо", "звезда" или полносвязная сеть. Все перечисленные топологии обладают свойством однородности, то есть все компьютеры в такой сети имеют одинаковые права в отношении доступа к другим компьютерам (за исключением центрального компьютера при соединении "звезда"). Такая однородность структуры упрощает процедуру наращивания числа компьютеров, облегчает обслуживание и эксплуатацию сети.

Однако при построении больших сетей однородная структура связей превращается из преимущества в недостаток. В таких сетях использование типовых структур порождает различные ограничения, важнейшими из которых являются:

- ограничения на длину связи между узлами;
- ограничения на количество узлов в сети;
- ограничения на интенсивность трафика, который генерируют узлы сети.

Например, технология Ethernet на тонком коаксиальном кабеле позволяет использовать кабель длиной не более 185 метров, к которому можно подключить не более 30 компьютеров. Однако, если компьютеры интенсивно обмениваются информацией, иногда приходится снижать число подключенных к кабелю машин до 20, а то и до 10, чтобы каждому компьютеру доставалась приемлемая доля общей пропускной способности сети.

Для снятия этих ограничений используются особые методы *структуризации* сети и специальное структурообразующее оборудование – *повторители*, *концентраторы*, *мосты*, *коммутаторы*, *маршрутизаторы*. Такого рода оборудование также называют *коммуникационным*, имея в виду, что с его помощью отдельные сегменты сети взаимодействуют между собой.

Различают:

- топологию физических связей (*физическую структуру*) сети. В этом случае конфигурация физических связей определяется электрическими соединениями компьютеров, то есть ребрам графа соответствуют отрезки кабеля, связывающие пары узлов;
- топологию логических связей (*логическую структуру*) сети. Здесь в качестве логических связей выступают маршруты передачи данных между узлами сети, которые образуются путем соответствующей настройки коммуникационного оборудования.

4.1. Физическая структуризация сети

Простейшее из коммуникационных устройств – *повторитель* (repeator) – используется для физического соединения различных сегментов кабеля локальной сети с целью увеличения общей длины сети. Повторитель передает сигналы, приходящие из одного сегмента сети, в другие ее сегменты (рис. 4.1). Повторитель позволяет преодолеть ограничения на длину линий связи за счет улучшения качества передаваемого сигнала – восстановления его мощности и амплитуды, улучшения фронтов и т.п.

Повторитель, который имеет несколько портов и соединяет несколько физических сегментов, часто называют *концентратором* (concentrator) или *хабом* (hub). Эти названия (hub – основа, центр деятельности) отражают тот факт, что в данном устройстве сосредоточены все связи между сегментами сети.

Использование концентраторов характерно практически для всех базовых технологий локальных сетей – Ethernet, ArcNet, Token Ring, FDDI, Fast Ethernet, Gigabit Ethernet.

Нужно подчеркнуть, что в работе любых концентраторов много общего – они повторяют сигналы, пришедшие с одного из их портов, на других своих портах. Разница состоит в том, на каких именно

портах повторяются входные сигналы. Так, концентратор Ethernet повторяет входные сигналы на всех своих портах, кроме того, с которого сигналы поступают (рис. 4.2).

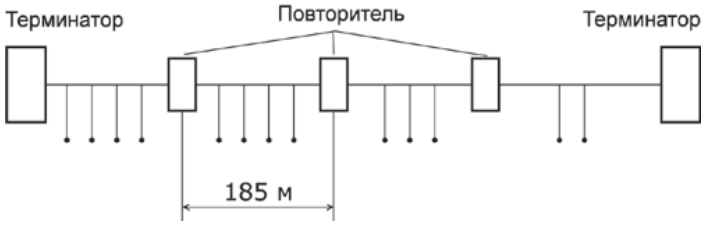

Рис. 4.1. Повторитель позволяет увеличить длину сети Ethernet

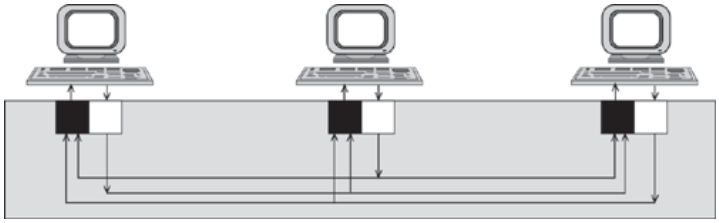

Рис. 4.2. Концентратор Ethernet

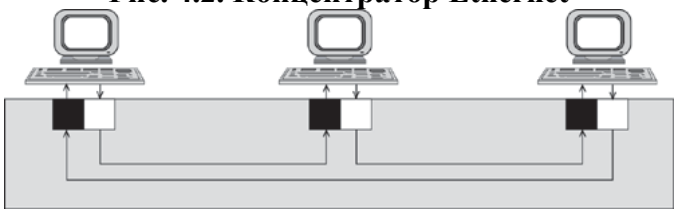

Рис. 4.3. Концентратор Token Ring

Концентратор Token Ring (рис. 4.3) повторяет входные сигналы, поступающие с некоторого порта, только на одном порту – на том, к которому подключен следующий в кольце компьютер.

Добавление в сеть концентратора всегда изменяет физическую топологию сети, но при этом оставляет без изменений ее логическую топологию.

Как уже было сказано, под физической топологией понимается конфигурация связей, образованных отдельными частями кабеля, а под логической – конфигурация информационных потоков между компьютерами сети. Во многих случаях физическая и логическая топологии сети совпадают. Например, сеть, представленная на рис. 4.4, *а*, имеет физическую топологию "кольцо". Компьютеры такой сети получают доступ к кабелям кольца за счет передачи друг другу специального кадра – маркера, причем этот маркер также передается последовательно от компьютера к компьютеру в том же порядке, в котором компьютеры образуют физическое кольцо, то есть компьютер А передает маркер компьютеру В, компьютер В – компьютеру С и т.д.

а)

б)

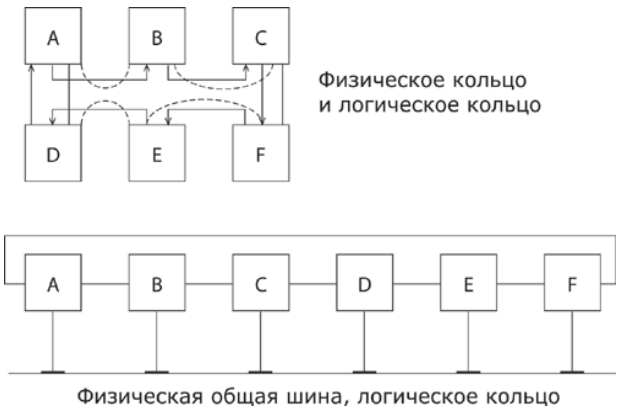

Рис. 4.4. Связь логической и физической структур сети:
а – логическая и физическая структуры сети совпадают;
б – логическая структура не совпадает с физической

Сеть, показанная на рис. 4.4, *б*, демонстрирует пример несовпадения физической и логической топологии. Физически компьютеры соединены по топологии "общая шина". Доступ же к шине происходит не по алгоритму случайного доступа, применяемому в технологии Ethernet, а путем передачи маркера в кольцевом порядке: от компьютера А – компьютеру В, от компьютера В – компьютеру С и т.д. Здесь порядок передачи маркера уже не повторяет физические связи, а определяется логическим конфигурированием драйверов сетевых адаптеров. Ничто не мешает настроить сетевые адаптеры и их драйверы так, чтобы компьютеры образовали кольцо в другом порядке, например: В, А, С... При этом физическая структура сети не изменяется.

Другим примером несовпадения физической и логической топологий сети является уже рассмотренная сеть на рис. 4.2. Концентратор Ethernet поддерживает в сети физическую топологию "звезда". Однако логическая топология сети осталась без изменений – это "общая шина". Так как концентратор повторяет данные, пришедшие с любого порта, на всех остальных портах, то они появляются на всех физических сегментах сети одновременно, как и в сети с физической общей шиной. Логика доступа к сети не меняется: все компоненты алгоритма случайного доступа – определение незанятости среды, захват среды, распознавание и отработка коллизий – остаются в силе.

Физическая структуризация сети с помощью концентраторов полезна не только для увеличения расстояния между узлами сети, но и для повышения ее надежности. Например, если какой-либо компьютер сети Ethernet с физической общей шиной из-за сбоя начинает непрерывно передавать данные по общему кабелю, то вся сеть выходит из строя, и остается только одно – вручную отсоединить сетевой адаптер этого компьютера от кабеля. В сети Ethernet, построенной с использованием концентратора, эта проблема может быть решена автоматически – концентратор отключает свой порт, если обнаруживает, что присоединенный к нему узел слишком долго монопольно занимает сеть. Концентратор может блокировать некорректно работающий узел и в других случаях, выполняя роль некоторого управляющего узла.

4.2. Логическая структуризация сети

Физическая структуризация сети полезна во многих отношениях, однако в ряде случаев, обычно относящихся к сетям большого и среднего размера, без логической структуризации сети обойтись невозможно. Наиболее важной проблемой, не решаемой путем физической структуризации, остается проблема перераспределения передаваемого трафика между различными физическими сегментами сети.

В большой сети естественным образом возникает неоднородность информационных потоков: *сеть* состоит из множества *подсетей* рабочих групп, отделов, филиалов предприятия и других административных образований. В одних случаях наиболее интенсивный обмен данными наблюдается между компьютерами, принадлежащими одной подсети, и только небольшая часть обращений происходит к ресурсам компьютеров, находящихся вне локальных рабочих групп. На других предприятиях, особенно там, где имеются централизованные хранилища корпоративных данных, активно используемые всеми сотрудниками предприятия, наблюдается обратная ситуация: интенсивность внешних обращений выше интенсивности обмена между "соседними" машинами. Но независимо от того, как распределяются внешний и внутренний трафик, для повышения эффективности работы сети неоднородность информационных потоков необходимо учитывать.

Сеть с типовой топологией ("шина", "кольцо", "звезда"), в которой все физические сегменты рассматриваются в качестве одной разделяемой среды, оказывается неадекватной структуре информационных потоков в большой сети. Например, в сети с общей шиной взаимодействие любой пары компьютеров занимает ее на все время обмена, поэтому при увеличении числа компьютеров в сети шина становится узким местом. Компьютеры одного отдела вынуждены ждать, когда завершит обмен пара компьютеров другого отдела.

Этот случай иллюстрирует рис. 4.5, на котором показана сеть, построенная с использованием концентраторов. Пусть компьютер А, на-

Рис. 4.5. Физическая структуризация на основе концентраторов

ходящийся в одной подсети с компьютером B, посылает ему данные. Несмотря на разветвленную физическую структуру сети, концентраторы распространяют любой кадр по всем ее сегментам. Поэтому кадр, посылаемый компьютером A компьютеру B, хотя и не нужен компьютерам отделов 2 и 3, в соответствии с логикой работы концентраторов поступает на эти сегменты тоже (на рисунке кадр, посланный компьютером A, показан в виде заштрихованного кружка, который повторяется на всех сетевых интерфейсах данной сети). И до тех пор, пока компьютер B не получит адресованный ему кадр, ни один из компьютеров этой сети не сможет передавать данные.

Такая ситуация возникает из-за того, что логическая структура данной сети осталась однородной – она никак не учитывает возможность локальной обработки трафика внутри отдела и предоставляет всем парам компьютеров равные возможности по обмену информацией (рис. 4.6).

Для решения проблемы придется отказаться от идеи единой однородной разделяемой среды (рис. 4.7). Например, в рассмотренном выше примере желательно было бы сделать так, чтобы кадры, которые передают компьютеры отдела 1, выходили бы за пределы этой части сети в том и только в том случае, если эти кадры направлены какому-либо компьютеру из других отделов. С другой стороны, в сеть каждого из отделов должны попадать только те кадры, которые адресованы узлам этой сети. При такой организации работы сети ее производительность существенно повысится, так как компьютеры одного отдела не будут простаивать в то время, когда обмениваются данными компьютеры других отделов.

Нетрудно заметить, что в предложенном решении отказались от идеи общей разделяемой среды в пределах всей сети, хотя и оставили ее в пределах каждого отдела. Пропускная способность линий связи между отделами не должна совпадать с пропускной способностью среды внутри отделов. Если трафик между отделами составляет только

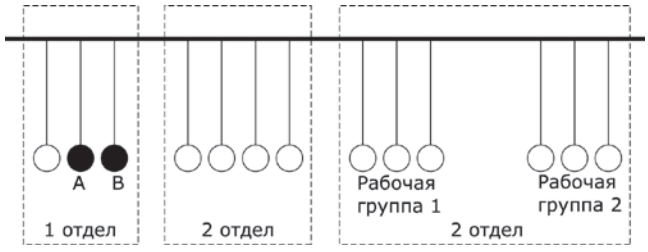

Рис. 4.6. Логическая структура продолжает соответствовать "общей шине"

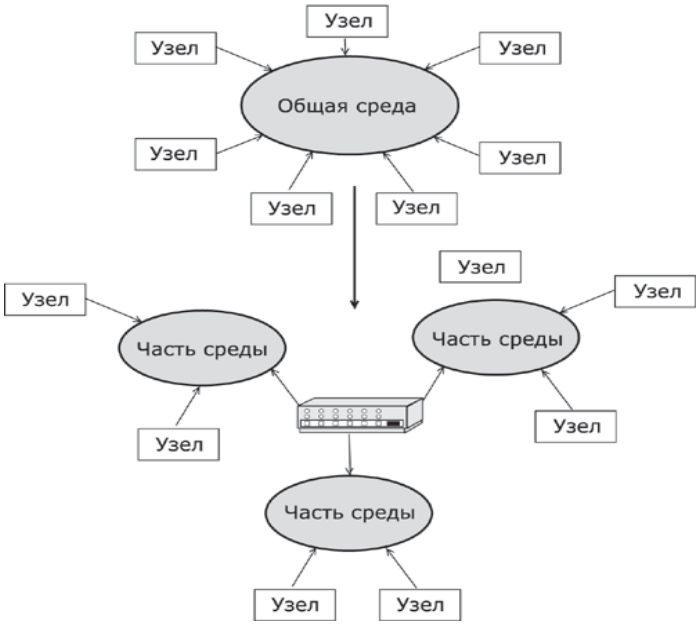

Рис. 4.7. Отказ от единой разделяемой среды

20 % трафика внутри отдела (как уже отмечалось, эта величина может быть другой), то и пропускная способность линий связи и коммуникационного оборудования, соединяющего отделы, может быть значительно ниже внутреннего трафика сети отдела.

Распространение трафика, предназначенного для компьютеров некоторого сегмента сети, только в пределах этого сегмента, называется *локализацией трафика*. Логическая структуризация сети – это процесс разбиения сети на сегменты с локализованным трафиком.

Для логической структуризации сети используются коммуникационные устройства: *мосты*, *коммутаторы*, *маршрутизаторы*, *шлюзы*.

Мост (bridge) делит разделяемую среду передачи сети на части (часто называемые логическими сегментами), передавая информацию из одного сегмента в другой только в том случае, если такая передача действительно необходима, то есть если адрес компьютера назначения принадлежит другой подсети. Тем самым мост изолирует трафик одной подсети от трафика другой, повышая общую производительность передачи данных в сети. Локализация трафика не только экономит пропускную способность, но и уменьшает возможность несанкционированного доступа к данным, так как кадры не выходят за пределы своего сегмента, и злоумышленнику сложнее перехватить их.

На рис. 4.8 показана сеть, которая была получена из сети с центральным концентратором (см. рис. 4.5) путем его замены на мост. Сети 1-го и 2-го отделов состоят из отдельных логических сегментов, а сеть отдела 3 – из двух логических сегментов. Каждый логический сегмент построен на базе концентратора и имеет простейшую физическую структуру, образованную отрезками кабеля, связывающими компьютеры с портами концентратора. Если пользователь компьютера A пошлет данные пользователю компьютера B, находящемуся в одном с ним сегменте, то эти данные будут повторены только на тех сетевых интерфейсах, которые отмечены на рисунке заштрихованными кружками.

Мосты используют для локализации трафика аппаратные адреса компьютеров. Это затрудняет распознавание принадлежности того или иного компьютера к определенному логическому сегменту – сам адрес не содержит подобной информации. Поэтому мост достаточно упрощенно представляет деление сети на сегменты – он запоминает, через какой порт на него поступил кадр данных от каждого компьютера сети, и в дальнейшем передает кадры, предназначенные для данного компьютера, на этот порт. Точной топологии связей между логическими сегментами мост не знает. Из-за этого применение мостов приводит к значительным ограничениям на конфигурацию связей сети – сегменты должны быть соединены таким образом, чтобы в сети не образовывались замкнутые контуры.

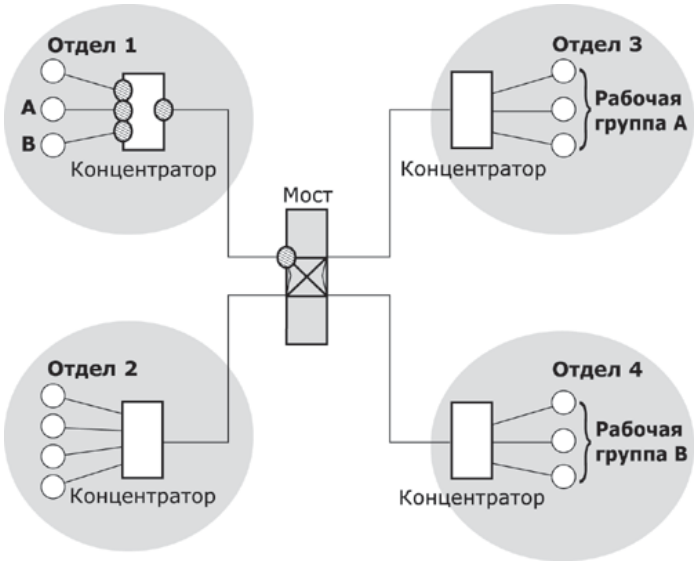

Рис. 4.8. Логическая структуризация сети с помощью моста

Коммутатор (switch) по принципу обработки кадров от моста практически ничем не отличается. Единственное его отличие состоит в том, что он является своего рода коммуникационным мультипроцессором, так как каждый его порт оснащен специализированной микросхемой, которая обрабатывает кадры по алгоритму моста независимо от микросхем других портов. За счет этого общая производительность коммутатора обычно намного выше производительности традиционного моста, имеющего один процессорный блок. Можно сказать, что коммутаторы – это мосты нового поколения, которые обрабатывают кадры в параллельном режиме.

Производительность сети, построенной на коммутаторе, обычно в несколько раз превышает производительность аналогичной сети, построенной с использованием концентратора. Этот эффект является следствием нескольких факторов:

- Передача кадров только на тот сегмент (микросегменты, если к ним подключен один конечный узел), на котором этот кадр действительно нужен. Точно также работали локальные мосты, но они не обладали следующим свойством.
- Параллельная передача кадров между входными и выходными портами, неблокирующий характер всех обрабатывающих узлов коммутатора – процессора порта, коммутирующей матрицы, внутренней межмодульной шины и т.п. Параллельная передача кадров совмещает во времени все этапы по передаче кадра от входного порта – помещение его во внутренний буфер, просмотр адресной таблицы, перенос кадра в выходной буфер выходного порта – и поэтому позволяет потенциально ускорить пропускную способность участка сети в $N / 2$ раз по сравнению с применением концентратора, где N – число портов, работающих в классическом полудуплексном режиме. Если порты работают в полнодуплексном режиме, то ускорение может составить N раз (рис. 4.9). Неблокирующий характер означает, что все узлы коммутатора могут обработать поток кадров, поступающий на входные порты с максимальной для протокола скоростью.
- Конвейерный способ передачи кадра между входным и выходным портами, когда кадр начинает передаваться на выходной порт сразу же после прихода нескольких первых байт с адресом назначения. Этот фактор не так значим, как параллельная обработка кадров, поступающих на несколько портов.

Однако существуют ситуации, когда применение коммутатора не приводит к заметному повышению производительности работы сегмента.

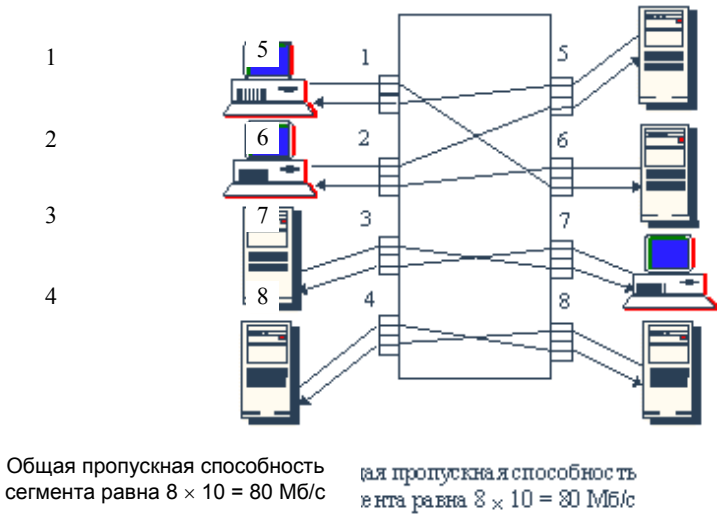

Рис. 4.9. Ускорение процесса обмена данными при использовании коммутаторов

1. Весь трафик или его большая часть предназначен для одного выходного порта. Типичный пример такой ситуации приведен на рис. 4.10, он соответствует сети с одним выделенным сервером, который особенно часто наблюдается при использовании ОС NovellNetWare. Коммутатору нечего распараллеливать. Максимальный эффект в этом случае может быть достигнут при использовании полнодуплексных связей, но все равно ускорение будет далеко от N – всего в два раза. Для GigabitEthernet с полнодуплексным повторителем эффект вообще будет близок к нулевому. Отсюда видно, что эффект применения коммутаторов зависит от распределения трафика между узлами сети – чем больше распределение близко к равновероятному, тем больше коммутаторы повышают производительность сети.

2. В сети существует очень интенсивный источник широковещательного трафика (часто это является следствием сбойной работы конечного узла), и все сегменты, подключенные к портам коммутатора, засоряются этим трафиком. Коммутатор в силу алгоритма своей работы обязан передавать широковещательные кадры на все сегменты, если кадр правильно оформлен. Поэтому для борьбы с широковещательным штормом приходится использовать другие средства – виртуальные сети VLAN и маршрутизаторы – как автономные, так и совмещенные с коммутатором. В последнем случае коммутатор обычно называют коммутатором третьего уровня.

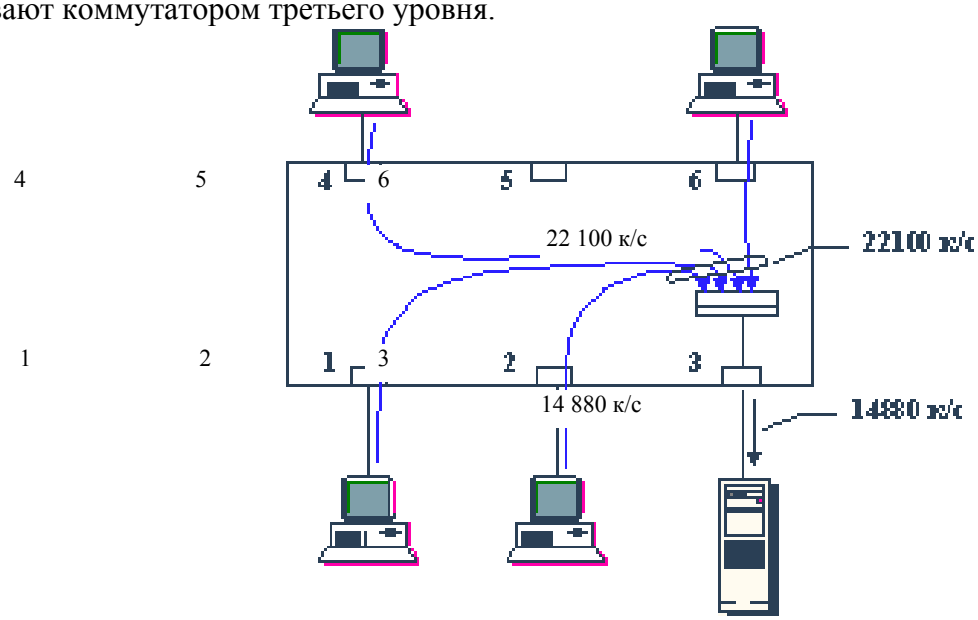

Рис. 4.10. Переполнение буфера порта из-за несбалансированности трафика

3. Коммутатор работает в режиме перегрузки и не справляется с потоком поступающих на него кадров. Часть кадров теряется и производительность сети не только не повышается, но иногда даже и существенно снижается.

Врожденная болезнь коммутаторов – потеря кадров. Причины потери коммутаторами кадров можно разделить на две категории – блокирующая производительность коммутатора и перегрузка выходного порта или портов.

Блокирующая производительность – это ситуация, когда выходные порты коммутатора могли бы передать в сеть все поступающие потоки кадров, но производительность внутренних элементов коммутатора недостаточна для обработки этих потоков. Многие модульные модели коммутаторов страдают от этого недостатка. Как правило, он является следствием появления в номенклатуре коммутатора высокоскоростных модулей, например, модулей FastEthernet, ATM и GigabitEthernet, уже после выпуска общих элементов коммутатора, например, коммутационной матрицы, блока общей памяти, пассивной общей шины и т.п. В результате – нехватка производительности для особенно тяжелых режимов работы.

Но, если недостаточную производительность элементов коммутатора можно устранить, заменив их более скоростными, то производительность порта коммутатора ограничена стандартом соответствующего протокола – Ethernet, TokenRing, FastEthernet и ATM.

Например, порт Ethernet не может передавать больше 14 880 кадров в секунду, если он не нарушает временных соотношений, установленных стандартом. Какой бы ни был объем буфера порта, он в какой-то момент времени может переполниться, так как является разделяемым ресурсом для многих потоков кадров.

Потеря кадра может привести к снижению полезной для приложения пропускной способности в несколько раз. Количественно ущерб зависит от того, какой протокол занимается восстановлением потерянных в кадре данных. Если это протокол прикладного или транспортного уровня, то при тайм-аутах в сотни миллисекунд, рассчитанных на работу в больших маршрутизируемых составных сетях, снижение производительности может доходить до десятков раз. Исправление потерь кадров на нижних уровнях делает сеть менее чувствительной к этому фактору, но в локальных сетях такая ситуация наблюдается только при использовании протокола NetBIOS или же стека SNA компании IBM в сетях TokenRing, а это не такой уж большой процент сетей. Тенденции таковы, что все чаще в локальных сетях будет использоваться протокол стека TCP/IP, а протокол TCP, решающий проблему восстановления потерянной информации, вычисляет тайм-аут адаптивно, в зависимости от времени прихода подтверждений от узла-собеседника. При работе через Internet это время может составлять сотни миллисекунд, замедляя процесс восстановления кадров так же, как и в сетях NetWare.

Потери кадров при перегрузках выходных портов вообще типичны для полнодуплексных сетей, построенных на основе коммутаторов. Нужно заметить, что такие сети давно используются – это почти все виды глобальных сетей: X.25, framerelay и ATM. Индивидуальными в них являются только связи между абонентами сети и первыми коммутаторами, а связи коммутатор-коммутатор всегда разделяются во времени между потоками кадров нескольких абонентов. И если в сетях framerelay и ATM разработаны достаточно тонкие процедуры для поддержания требуемого баланса использования общих каналов между абонентами сети – процедуры управления качеством обслуживания, то в коммутаторах локальных сетей такие процедуры либо не поддерживаются вообще, либо реализуются в самой рудиментарной форме.

Угроза потери кадров – плата за отказ от разделяемой среды на участке сети "компьютер-концентратор". Коммутатор ускоряет работу сети, но при этом пропадает эффект, создаваемый алгоритмом доступа, когда на среду передачи данных разрешалось в каждый момент времени помещение кадра только от одного компьютера. Тем самым регулировался входной поток кадров от компьютеров и гарантировалась невозможность потери кадра при корректной работе оборудования. Поэтому локальные сети на основе коммутаторов, как и глобальные, требуют решения *задачи управления потоком данных*, то есть встраивания в коммутаторы и сетевые адаптеры механизма, который бы притормаживал конечные узлы при перегрузках коммутаторов сети.

Ограничения, связанные с применением мостов и коммутаторов – по топологии связей, а также ряд других, – привели к тому, что в ряду коммуникационных устройств появился еще один тип оборудования – *маршрутизатор* [router; в терминологии Internet маршрутизатор часто называют *шлюзом* (gateway)]. Маршрутизаторы более надежно и более эффективно, чем мосты, изолируют трафик отдельных частей сети друг от друга. Маршрутизаторы образуют логические сегменты посредством явной адресации, поскольку используют не плоские аппаратные, а составные числовые адреса. В этих адресах имеется поле номера сети, так что все компьютеры, у которых значение этого поля одинаковое, принадлежат одному сегменту, называемому в данном случае *подсетью* (subnet).

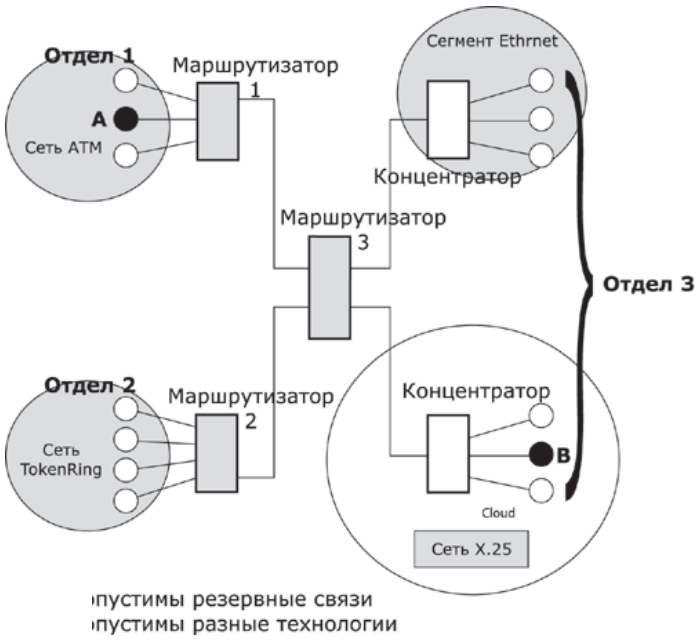

Рис. 4.11. Логическая структуризация сети с помощью маршрутизаторов

Кроме локализации трафика, маршрутизаторы выполняют еще много других полезных функций. Так, маршрутизаторы могут работать в сети с замкнутыми контурами, при этом они осуществляют выбор наиболее рационального маршрута из нескольких возможных. Сеть, представленная на рис. 4.11, отличается от своей предшественницы (см. рис. 4.8) тем, что между подсетями отделов 1 и 2 проложена дополнительная связь, которая может использоваться для повышения как производительности сети, так и ее надежности.

Другой очень важной функцией маршрутизаторов является их способность связывать в единую сеть подсети, построенные с использованием разных сетевых технологий, например Ethernet и X.25.

Кроме перечисленных устройств, отдельные части сети может соединять *шлюз* (gateway). Обычно основной причиной использования шлюза в сети является необходимость объединить сети с разными типами системного и прикладного программного обеспечения, а не желание локализовать трафик. Тем не менее, шлюз обеспечивает и локализацию трафика в качестве некоторого побочного эффекта.

Большие сети практически никогда не строятся без логической структуризации. Для отдельных сегментов и подсетей характерны типовые однородные топологии базовых технологий, и для их объединения всегда используется оборудование, обеспечивающее локализацию трафика: мосты, коммутаторы, маршрутизаторы и шлюзы.

5. Многослойная модель сети

Даже поверхностно рассматривая работу сети, можно заключить, что вычислительная сеть – это сложный комплекс взаимосвязанных и согласованно функционирующих программных и аппаратных компонентов. Изучение сети в целом предполагает знание принципов работы отдельных ее элементов, таких, как:

- компьютеры;
- коммуникационное оборудование;
- операционные системы;
- сетевые приложения.

Весь комплекс программно-аппаратных средств сети может быть описан *многослойной моделью* (рис. 5.1). В основе любой сети лежит аппаратный слой стандартизированных компьютерных *платформ*.

В настоящее время в сетях успешно применяются компьютеры различных классов – от персональных компьютеров до мэйнфреймов и супер-ЭВМ. Набор компьютеров в сети должен соответствовать набору решаемых сетью задач.

Второй слой – это коммуникационное оборудование. Хотя компьютеры и являются центральными элементами обработки данных в сетях, в последнее время не менее важную роль стали играть коммуникационные устройства. Кабельные системы, повторители, мосты, коммутаторы, маршрутизаторы и модульные концентраторы из вспомогательных компонентов сети превратились в основные наряду с компьютерами и системным программным обеспечением как по влиянию на характеристики сети, так и по стоимости. Сегодня коммуникационное устройство может представлять собой сложный специализированный мультипроцессор, который нужно конфигурировать, оптимизировать и администрировать. Изучение принципов работы коммуникационного оборудования требует знакомства с большим количеством протоколов, используемых как в локальных, так и в глобальных сетях.

Третьим слоем, образующим программную платформу сети, являются операционные системы (ОС). От того, какие концепции управления локальными и распределенными ресурсами положены в основу сетевой ОС, зависит эффективность работы всей сети. При проектировании сети важно учитывать, насколько легко данная операционная система может взаимодействовать с другими ОС сети, какой она обеспечивает уровень безопасности и защищенности данных, до какой степени позволяет наращивать число пользователей, можно ли перенести ее на компьютер другого типа и многие другие соображения.

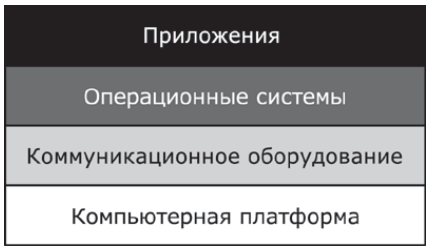

Рис. 5.1. Многослойная модель сети

Самый верхний слой сетевых средств образуют различные сетевые приложения, такие как сетевые *базы данных* (БД), почтовые системы, средства архивирования данных, системы автоматизации коллективной работы и т.д. Очень важно представлять диапазон возможностей, предоставляемых приложениями для различных областей применения, а также знать, насколько они совместимы с другими сетевыми приложениями и операционными системами.

Вычислительная сеть – это многослойный комплекс взаимосвязанных и согласованно функционирующих программных и аппаратных компонентов: компьютеров, коммуникационного оборудования, операционных систем, сетевых приложений.

5.1. Функциональные роли компьютеров в сети

В зависимости от того, как распределены функции между компьютерами сети, они могут выступать в трех разных ролях:

1) выделенного сервера сети (рис. 5.2);
2) узла-клиента (рис. 5.3);
3) одноранговогo узла (рис. 5.4).

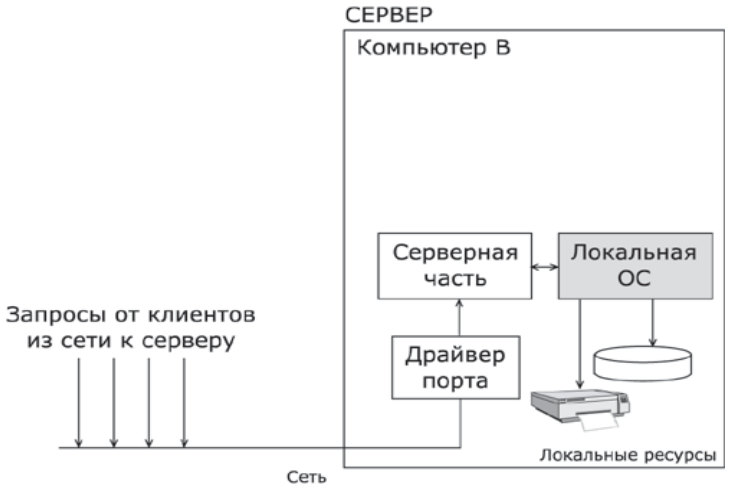

Рис. 5.2. Компьютер, занимающийся исключительно обслуживанием запросов других компьютеров

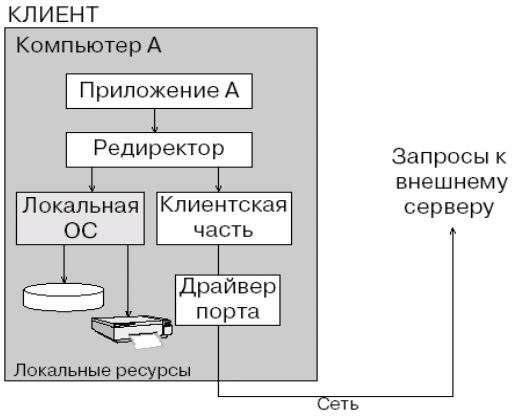

Рис. 5.3. Компьютер, обращающийся с запросами к ресурсам другой машины

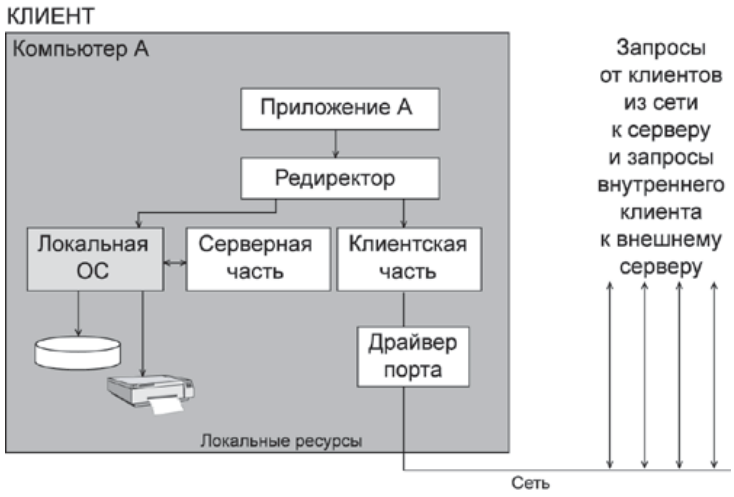

Рис. 5.4. Компьютер, совмещающий функции клиента и сервера

Очевидно, что сеть не может состоять только из клиентских или только из *серверных узлов*.

Сеть может быть построена по одной из трех схем:

1) на основе одноранговых узлов – *одноранговая сеть*;
2) на основе клиентов и серверов – *сеть с выделенными серверами*;
3) включающая узлы всех типов – *гибридная сеть*.

Каждая из этих схем имеет свои достоинства и недостатки, определяющие их области применения.

5.2. ОДНОРАНГОВЫЕ СЕТИ

В одноранговых сетях (рис. 5.5) все компьютеры равны в возможностях доступа к ресурсам друг друга. Каждый пользователь может по своему желанию объявить какой-либо ресурс своего компьютера разделяемым, после чего другие пользователи могут с ним работать. В одноранговых сетях на всех компьютерах устанавливается такая операционная система, которая предоставляет всем компьютерам в сети потенциально равные возможности. Сетевые операционные системы такого типа называются одноранговыми ОС. Очевидно, что одноранговые ОС должны включать как серверные, так и клиентские компоненты сетевых служб (на рисунке они обозначены буквами, соответственно, С и К). Примерами одноранговых ОС могут служить LANtastic, Personal Ware, Windows for Workgroups, Windows NT Workstation, Windows 95/98.

При потенциальном равноправии всех компьютеров в одноранговой сети часто возникает функциональная несимметричность. Обычно некоторые пользователи не желают предоставлять свои ресурсы для совместного доступа. В таком случае серверные возможности их операционных систем не активизируются, и компьютеры играют роль "чистых" клиентов (на рисунке неиспользуемые компоненты ОС изображены затемненными).

В то же время администратор может закрепить за некоторыми компьютерами сети только функции, связанные с обслуживанием запросов от остальных компьютеров, превратив их таким образом в "чистые" серверы, за которыми пользователи не работают. В такой конфигурации одноранговые сети становятся похожими на сети с выделенными серверами, но это только внешнее сходство – между этими

двумя типами сетей остается существенное различие. Изначально в одноранговых сетях отсутствует специализация ОС в зависимости от того, какую роль играет компьютер – клиента или сервера. Изменение роли компьютера в одноранговой сети достигается за счет того, что функции серверной или клиентской частей просто не используются.

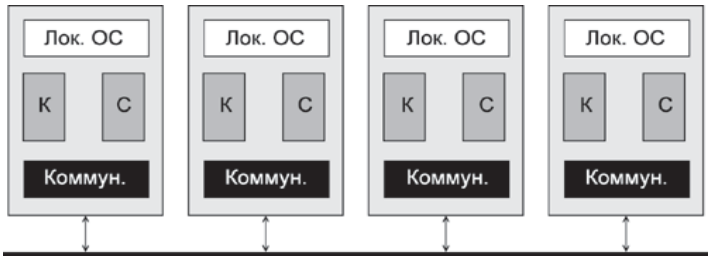

Рис. 5.5. Одноранговая сеть

(здесь словом "Коммун." обозначены коммуникационные средства)

Одноранговые сети проще в развертывании и эксплуатации; по этой схеме организуется работа в небольших сетях, в которых количество компьютеров не превышает 10–20. В этом случае нет необходимости в применении централизованных средств администрирования – нескольким пользователям нетрудно договориться между собой о перечне разделяемых ресурсов и паролях доступа к ним.

Однако в больших сетях средства централизованного администрирования, хранения и обработки данных, а особенно защиты данных, необходимы. Такие возможности легче обеспечить в сетях с выделенными серверами.

5.3. СЕТИ С ВЫДЕЛЕННЫМ СЕРВЕРОМ

В сетях с выделенными серверами (рис. 5.6) используются специальные варианты сетевых ОС, которые оптимизированы для работы в роли серверов и называются *серверными ОС*. Пользовательские компьютеры в таких сетях работают под управлением *клиентских ОС*.

Специализация операционной системы для работы в роли сервера является естественным способом повышения эффективности серверных операций. Необходимость такого повышения часто ощущается весьма остро, особенно в большой сети. При существовании в сети сотен или даже тысяч пользователей интенсивность запросов к разделяемым ресурсам может быть очень значительной, и сервер должен справляться с этим потоком запросов без больших задержек. Очевидным решением этой проблемы является использование в качестве сервера компьютера с мощной аппаратной платформой и операционной системой, оптимизированной для серверных функций.

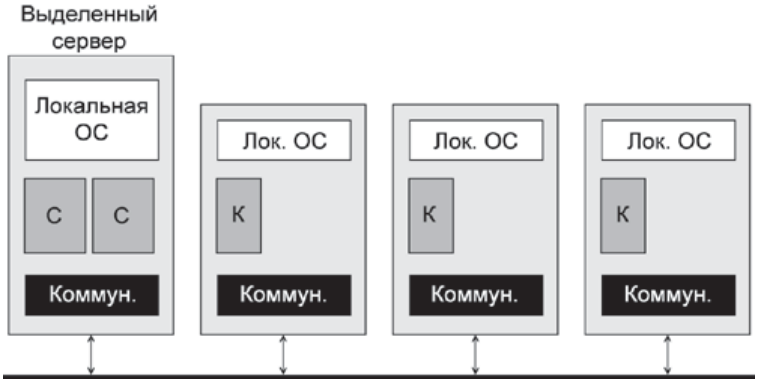

Рис. 5.6. Сеть с выделенным сервером

Чем меньше функций выполняет ОС, тем более эффективно можно их реализовать, поэтому для оптимизации серверных операций разработчики ОС вынуждены ущемлять некоторые другие ее функции, причем иногда даже полностью отказываться от них. Одним из ярких примеров такого подхода является серверная ОС NetWare. Ее разработчики поставили перед собой цель оптимизировать выполнение файлового сервиса и сервиса печати. Для этого они полностью исключили из системы многие элементы, важные для универсальной ОС, в частности, графический интерфейс пользователя, поддержку универсальных приложений, защиту приложений мультипрограммного режима друг от друга, механизм вирту-

альной памяти. Все это позволило добиться уникальной скорости файлового доступа и вывело NetWare в лидеры серверных ОС на долгое время.

Однако слишком узкая специализация некоторых серверных ОС является одновременно и их слабой стороной. Так, отсутствие в NetWare 4 универсального интерфейса программирования и средств защиты приложений, не позволившее использовать эту ОС в качестве среды для выполнения приложений, приводит к необходимости применения в сети других серверных ОС в тех случаях, когда требуется выполнение функций, отличных от файлового сервиса и сервиса печати. Поэтому разработчики многих серверных операционных систем отказываются от функциональной ограниченности и включают в состав серверных ОС все компоненты, позволяющие задействовать их в качестве универсальных серверов и даже клиентских ОС. Такие серверные ОС снабжаются развитым графическим пользовательским интерфейсом и поддерживают универсальный API. Это сближает их с одноранговыми операционными системами, но существует несколько отличий, которые позволяют отнести их именно к классу серверных ОС:

- поддержка мощных аппаратных платформ, в том числе мультипроцессорных;
- поддержка большого числа одновременно выполняемых процессов и сетевых соединений;
- включение в состав ОС компонентов централизованного администрирования сети (например, справочной службы или службы аутентификации и авторизации пользователей сети);
- более широкий набор сетевых служб.

Клиентские операционные системы в сетях с выделенными серверами обычно освобождаются от серверных функций, что значительно упрощает их организацию. Разработчики клиентских ОС уделяют основное внимание пользовательскому интерфейсу и клиентским частям сетевых служб. Наиболее простые клиентские ОС поддерживают только базовые сетевые службы, обычно, *файловую* и *службу печати*. В то же время существуют так называемые универсальные клиенты, которые поддерживают широкий набор клиентских частей, позволяющих им работать практически со всеми серверами сети.

Многие компании, разрабатывающие сетевые ОС, выпускают две версии одной и той же операционной системы. Одна версия предназначена для работы в качестве серверной ОС, а другая – для работы на клиентской машине. Эти версии чаще всего основаны на одном и том же базовом коде, но отличаются набором служб и утилит, а также параметрами конфигурации, в том числе устанавливаемыми по умолчанию и не поддающимися изменению.

Например, операционная система Windows NT выпускалась в версии для рабочей станции – Windows NT Workstation – и в версии для выделенного сервера – Windows NT Server. Оба эти варианта операционной системы включают клиентские и серверные части многих сетевых служб.

Так, ОС Windows NT Workstation, кроме выполнения функций сетевого клиента, может предоставлять сетевым пользователям файловый сервис, сервисы печати, удаленного доступа и другие, а, следовательно, может служить основой для одноранговой сети. С другой стороны, ОС Windows NT Server содержит все необходимые средства, которые позволяют задействовать компьютер в качестве клиентской рабочей станции. Под управлением ОС Windows NT Server локально запускаются прикладные программы, которые могут потребовать выполнения клиентских функций ОС при появлении запросов к ресурсам других компьютеров сети. Windows NT Server имеет такой же развитый графический интерфейс, как и Windows NT Workstation, что позволяет с равным успехом применять эти ОС для интерактивной работы пользователя или администратора.

Однако версия Windows NT Server имеет больше возможностей для предоставления ресурсов своего компьютера другим пользователям сети, так как может выполнять более широкий набор функций, поддерживает большее количество одновременных соединений с клиентами, реализует централизованное управление сетью, имеет более развитые средства защиты. Поэтому рекомендуется применять Windows NT Server в качестве ОС для выделенных серверов, а не клиентских компьютеров.

5.4. ГИБРИДНАЯ СЕТЬ

В больших сетях наряду с отношениями клиент-сервер сохраняется необходимость и в одноранговых связях, поэтому такие сети чаще всего строятся по гибридной схеме (рис. 5.7).

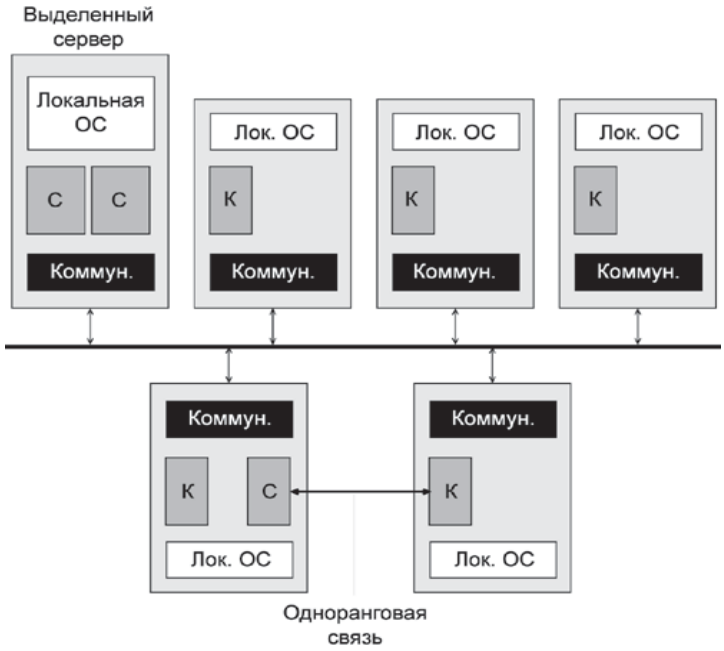

Рис. 5.7. Гибридная сеть

5.5. Сетевые службы и операционная система

Для конечного пользователя сеть – это не компьютеры, кабели и концентраторы и даже не информационные потоки, для него сеть – это, прежде всего, набор сетевых служб, с помощью которых он получает возможность просмотреть список имеющихся в сети компьютеров, прочитать удаленный файл, распечатать документ на "чужом" принтере или послать почтовое сообщение. Именно совокупность предоставляемых возможностей – насколько широк их выбор, насколько они удобны, надежны и безопасны – определяет для пользователя облик той или иной сети.

Кроме собственно обмена данными, сетевые службы должны решать и другие, более специфические, задачи, например, задачи, связанные с распределенной обработкой данных. К таким задачам относится обеспечение непротиворечивости нескольких копий данных, размещенных на разных машинах (служба репликации), или организация выполнения одной задачи параллельно на нескольких машинах сети (служба вызова удаленных процедур). Среди сетевых служб можно выделить *административные*, то есть такие, которые в основном ориентированы не на простого пользователя, а на администратора, и служат для обеспечения правильной работы сети в целом. Служба администрирования пользовательских учетных записей, которая позволяет администратору вести общую базу данных о пользователях сети, система мониторинга сети, позволяющая захватывать и анализировать сетевой трафик, служба безопасности, в функции которой может входить, помимо прочего, выполнение процедуры логического входа с последующей проверкой пароля, – все это примеры административных служб.

Реализация сетевых служб осуществляется программными средствами. Все сетевые службы построены в *архитектуре* "*клиент-сервер*".

Основные службы – *файловая служба* и *служба печати* – обычно предоставляются сетевой операционной системой, а вспомогательные, например *служба баз данных*, *факсимильной связи или передачи голоса*, – системными сетевыми приложениями или утилитами, работающими в тесном контакте с сетевой ОС. Вообще говоря, распределение служб между ОС и утилитами достаточно условно и меняется в зависимости от реализации ОС.

При разработке сетевых служб приходится решать задачи, свойственные любым распределенным приложениям: определение протокола взаимодействия между клиентской и серверной частями, распределение функций между ними, выбор схемы адресации приложений и т.д.

Одним из главных показателей качества сетевой службы является ее удобство. Для одного и того же ресурса может быть разработано несколько служб, по-разному решающих в общем-то одну и ту же задачу. Отличия могут заключаться в производительности или в уровне удобства предоставляемых услуг. Например, файловая служба может быть основана на использовании команды передачи файла из

одного компьютера в другой по имени файла, а это требует от пользователя знания имени нужного файла. Та же файловая служба может быть реализована и так, что пользователь монтирует удаленную файловую систему к локальному каталогу, а далее обращается к удаленным файлам как к своим собственным, что гораздо удобнее. Качество сетевой службы зависит и от качества пользовательского интерфейса – интуитивной понятности, наглядности, рациональности.

При определении степени удобства разделяемого ресурса часто употребляют термин "прозрачность". Прозрачный доступ – это такой доступ, при котором пользователь не замечает, где расположен нужный ему ресурс, – на его компьютере или на удаленном. После того как он смонтировал удаленную файловую систему в свое дерево каталогов, доступ к удаленным файлам становится для него совершенно прозрачным. Сама операция монтирования также может иметь разную степень прозрачности – в сетях с меньшей прозрачностью пользователь должен знать и задавать в команде имя компьютера, на котором хранится удаленная файловая система, в сетях с большей степенью прозрачности соответствующий программный компонент сети производит поиск разделяемых томов файлов независимо от мест их хранения, а затем предоставляет их пользователю в удобном для него виде, например в виде списка или набора пиктограмм.

Для обеспечения прозрачности важен способ адресации (именования) разделяемых сетевых ресурсов. Имена разделяемых сетевых ресурсов не должны зависеть от их физического расположения на том или ином компьютере. В идеале пользователь не должен ничего менять в своей работе, если администратор сети переместил том или каталог с одного компьютера на другой. Сам администратор и сетевая операционная система имеют информацию о расположении файловых систем, но от пользователя она скрыта. Такая степень прозрачности пока редко встречается в сетях, – обычно для получения доступа к ресурсам определенного компьютера сначала приходится устанавливать с ним логическое соединение. Такой подход применяется, например, в сетях Windows NT.

6. МОДЕЛЬ OSI

Тезис о пользе стандартизации, справедливый для всех отраслей, в компьютерных сетях приобретает особое значение. Суть сети – это соединение разного оборудования, а значит, проблема совместимости является одной из наиболее острых. Без соблюдения всеми производителями общепринятых правил разработки оборудования прогресс в деле "строительства" сетей был бы невозможен. Поэтому все развитие компьютерной отрасли, в конечном счете, отражено в стандартах – любая новая технология только тогда приобретает "законный" статус, когда ее содержание закрепляется в соответствующем стандарте.

В компьютерных сетях идеологической основой стандартизации является *многоуровневый подход* к разработке средств сетевого взаимодействия. Именно на основе этого подхода была создана стандартная семиуровневая модель взаимодействия *открытых систем*, ставшая своего рода универсальным языком сетевых специалистов.

6.1. Многоуровневый подход

Организация взаимодействия между устройствами сети является сложной задачей. Как известно, для решения сложных задач используется универсальный прием – *декомпозиция*, то есть разбиение одной задачи на несколько задач-модулей (рис. 6.1). Декомпозиция состоит в четком определении функций каждого модуля, а также порядка их взаимодействия (интерфейсов). В результате достигается логическое упрощение задачи, а кроме того, появляется возможность модификации отдельных модулей без изменения остальной части системы.

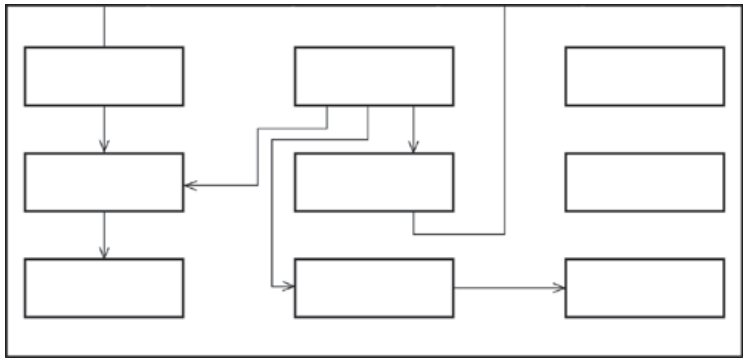

Рис. 6.1. Пример декомпозиции задачи

При декомпозиции часто используют многоуровневый подход. Он заключается в следующем:

– все множество модулей, решающих частные задачи, разбивают на группы и упорядочивают по уровням, образующим *иерархию*;

– в соответствии с принципом иерархии для каждого промежуточного уровня можно указать непосредственно примыкающие к нему соседние вышележащий и нижележащий уровни (рис. 6.2);

– группа модулей, составляющих каждый уровень, должна быть сформирована таким образом, чтобы все модули этой группы для выполнения своих задач обращались с запросами только к модулям соседнего нижележащего уровня;

– результаты работы всех модулей, отнесенных к некоторому уровню, могут быть переданы только модулям соседнего вышележащего уровня.

Такая иерархическая декомпозиция задачи предполагает четкое определение функции каждого уровня и интерфейсов между уровнями. Интерфейс определяет набор функций, которые нижележащий уровень предоставляет вышележащему. В результате иерархической декомпозиции достигается относительная независимость уровней, а значит, возможность их автономной разработки и модификации.

Средства решения задачи организации сетевого взаимодействия, конечно, тоже могут быть представлены в виде иерархически организованного множества модулей. Например, модулям нижнего уровня можно поручить вопросы, связанные с надежной передачей информации между двумя соседними узлами, а модулям следующего, более высокого, уровня – транспортировку сообщений в пределах всей сети. Очевидно, что последняя задача – организация связи двух любых, не обязательно соседних, узлов – является более общей и поэтому ее можно решить посредством многократных обращений к нижележащему уровню.

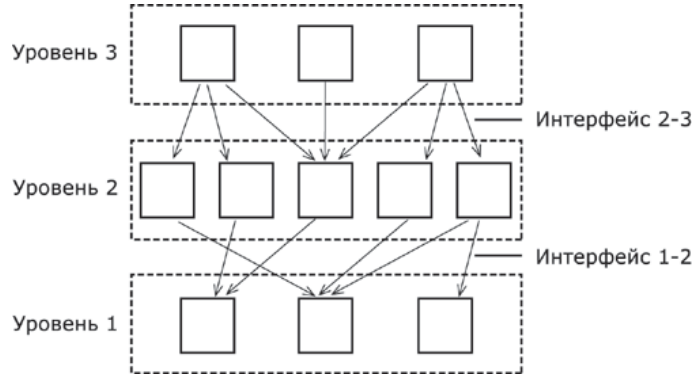

Рис. 6.2. Многоуровневый подход – создание иерархии задач

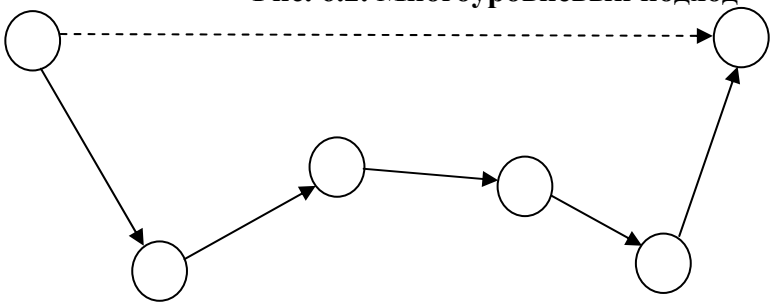

Рис. 6.3. Декомпозиция задачи связывания произвольной пары узлов на более частные задачи связывания пар соседних узлов

Так, связывание узлов А и Б (рис. 6.3) может быть сведено к последовательному связыванию пар промежуточных смежных узлов. Таким образом, модули вышележащего уровня при решении своих задач рассматривают средства нижележащего уровня как инструмент.

6.2. ПРОТОКОЛ. ИНТЕРФЕЙС. СТЕК ПРОТОКОЛОВ

Многоуровневое представление средств сетевого взаимодействия имеет свою специфику, связанную с тем, что в процессе обмена сообщениями участвуют две стороны, то есть в данном случае необходимо организовать согласованную работу двух "иерархий", работающих на разных компьютерах. Оба участника сетевого обмена должны принять множество соглашений. Например, они должны согласовать уровни и форму электрических сигналов, способ определения длины сообщений, договориться о методах контроля достоверности и т.п. Другими словами, соглашения должны быть приняты для всех уровней, начиная от самого низкого – уровня передачи битов – до самого высокого, реализующего сервис для пользователей сети.

На рис. 6.4 показана модель взаимодействия двух узлов. С каждой стороны средства взаимодействия представлены четырьмя уровнями. Процедура взаимодействия этих двух узлов может быть описана в виде набора правил взаимодействия каждой пары соответствующих уровней обеих участвующих сторон.

Формализованные правила, определяющие последовательность и формат сообщений, которыми обмениваются сетевые компоненты, лежащие на одном уровне, но в разных узлах, называются протоколом.

Модули, реализующие протоколы соседних уровней и находящиеся в одном узле, также взаимодействуют друг с другом в соответствии с четко определенными правилами с помощью стандартизированных форматов сообщений. Эти правила принято называть интерфейсом.

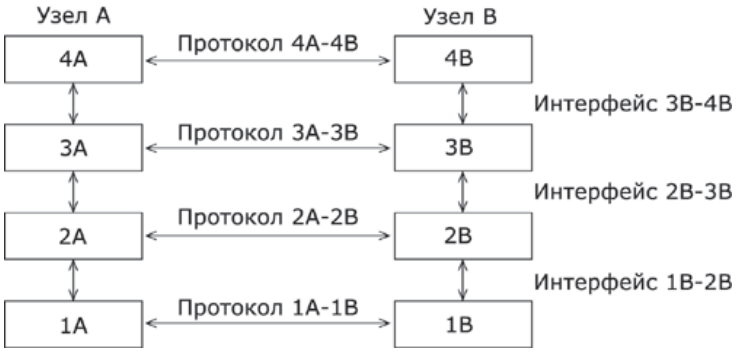

Рис. 6.4. Взаимодействие двух узлов

Интерфейс – определяет последовательность и формат сообщений, которыми обмениваются сетевые компоненты, лежащие на соседних уровнях в одном узле. Интерфейс определяет набор услуг, предоставляемый данным уровнем соседнему уровню.

В сущности, протокол и интерфейс выражают одно и то же понятие, но традиционно в сетях за ними закреплены разные области действия: протоколы определяют правила взаимодействия модулей одного уровня в разных узлах, а интерфейсы – модулей соседних уровней в одном узле.

Средства каждого уровня должны отрабатывать, во-первых, собственный протокол, а во-вторых, интерфейсы с соседними уровнями.

Иерархически организованный набор протоколов, достаточный для организации взаимодействия узлов в сети, называется *стеком коммуникационных протоколов*.

Коммуникационные протоколы могут быть реализованы как программно, так и аппаратно. Протоколы нижних уровней часто реализуются комбинацией программных и аппаратных средств, а протоколы верхних уровней – как правило, чисто программными средствами.

Программный модуль, реализующий некоторый протокол, часто для краткости также называют протоколом. При этом соотношение между протоколом как формально определенной процедурой и протоколом – программным модулем, реализующим эту процедуру, – аналогично соотношению между алгоритмом решения некоторой задачи и программой, решающей эту задачу.

Понятно, что один и тот же алгоритм может быть запрограммирован с разной степенью эффективности. Точно так же и протокол может иметь несколько программных реализаций. Именно поэтому при сравнении протоколов следует учитывать не только логику их работы, но и качество программных решений. Более того, на эффективность взаимодействия устройств в сети влияет качество всей совокупности протоколов, составляющих стек, в частности, то, насколько рационально распределены функции между протоколами разных уровней и насколько хорошо определены интерфейсы между ними.

Протоколы реализуются не только компьютерами, но и другими сетевыми устройствами – концентраторами, мостами, коммутаторами, маршрутизаторами и т.д. Действительно, в общем случае связь компьютеров в сети осуществляется не напрямую, а через различные коммуникационные устройства. В зависимости от типа устройства в нем должны быть встроенные средства, реализующие тот или иной набор протоколов.

Чтобы еще раз пояснить понятия "протокол" и "интерфейс", рассмотрим пример, не имеющий отношения к вычислительным сетям, а именно, обсудим взаимодействие двух предприятий, А и В. Между этими предприятиями существуют многочисленные договоренности и соглашения, например о регулярных поставках продукции. В соответствии с договоренностью начальник отдела продаж предприятия А регулярно в начале каждого месяца посылает официальное сообщение начальнику отдела закупок предприятия В о том, сколько какого товара может быть поставлено в этом месяце. В ответ на это сообщение начальник отдела закупок предприятия В посылает заявку установленного образца на нужное количество продукции. Возможно, подобная процедура включает дополнительные согласования; в любом случае, существует установленный порядок взаимодействия, который можно считать "протоколом уровня начальников". Начальники посылают свои сообщения и заявки через секретарей. Порядок взаимодействия начальника и секретаря соответствует понятию межуровневого интерфейса "начальник–секретарь". На предприятии А обмен документами между начальником и секретарем идет через специальную папку, а на предприятии В начальник общается с секретарем по факсу. Таким образом, интерфейсы "начальник–секретарь" на этих двух предприятиях отличаются.

После того как сообщения переданы секретарям, начальников не волнует, каким образом эти сообщения будут перемещаться дальше – по обычной почте или электронной, факсом или нарочным. Выбор

способа передачи – это уровень компетенции секретарей, они могут решать этот вопрос, не уведомляя о том своих начальников, так как их протокол взаимодействия связан только с передачей поступающих сверху сообщений, и не касается содержания этих сообщений. На рис. 6.5 показано, что в качестве протокола взаимодействия "секретарь–секретарь" используется обмен письмами. При решении иных вопросов начальники могут взаимодействовать по другим правилам-протоколам, но это не повлияет на работу секретарей, для которых не важно, какие сообщения отправ-

Рис. 6.5. Пример многоуровневого взаимодействия предприятий

лять, а важно, чтобы они дошли до адресата. Итак, в данном случае мы имеем дело с двумя уровнями – уровнем начальников и уровнем секретарей, и каждый из них имеет собственный протокол, который может быть изменен независимо от протокола другого уровня. В этой независимости протоколов друг от друга и состоит преимущество многоуровневого подхода.

6.3. ОБЩАЯ ХАРАКТЕРИСТИКА МОДЕЛИ OSI

Из того, что протокол представляет собой соглашение, принятое двумя взаимодействующими объектами, в данном случае двумя работающими в сети компьютерами, совсем не следует, что он обязательно является стандартным. Но на практике при реализации сетей обычно используются стандартные протоколы. Это могут быть фирменные, национальные или международные стандарты.

В начале 80-х годов ряд международных организаций по стандартизации – ISO, ITU-T и некоторые другие – разработали модель, которая сыграла значительную роль в развитии сетей. Эта модель называется моделью ISO/OSI.

Модель взаимодействия открытых систем (Open System Intercon-nection, OSI) определяет различные уровни взаимодействия систем в сетях с коммутацией пакетов, дает им стандартные имена и указывает, какие функции должен выполнять каждый уровень.

Модель OSI была разработана на основании большого опыта, полученного при создании компьютерных сетей, в основном глобальных, в 70-е годы. Полное описание этой модели занимает более 1000 страниц текста.

В модели OSI (рис. 6.6) средства взаимодействия делятся на семь уровней: прикладной, представительный, сеансовый, транспортный, сетевой, канальный и физический. Каждый уровень имеет дело с определенным аспектом взаимодействия сетевых устройств.

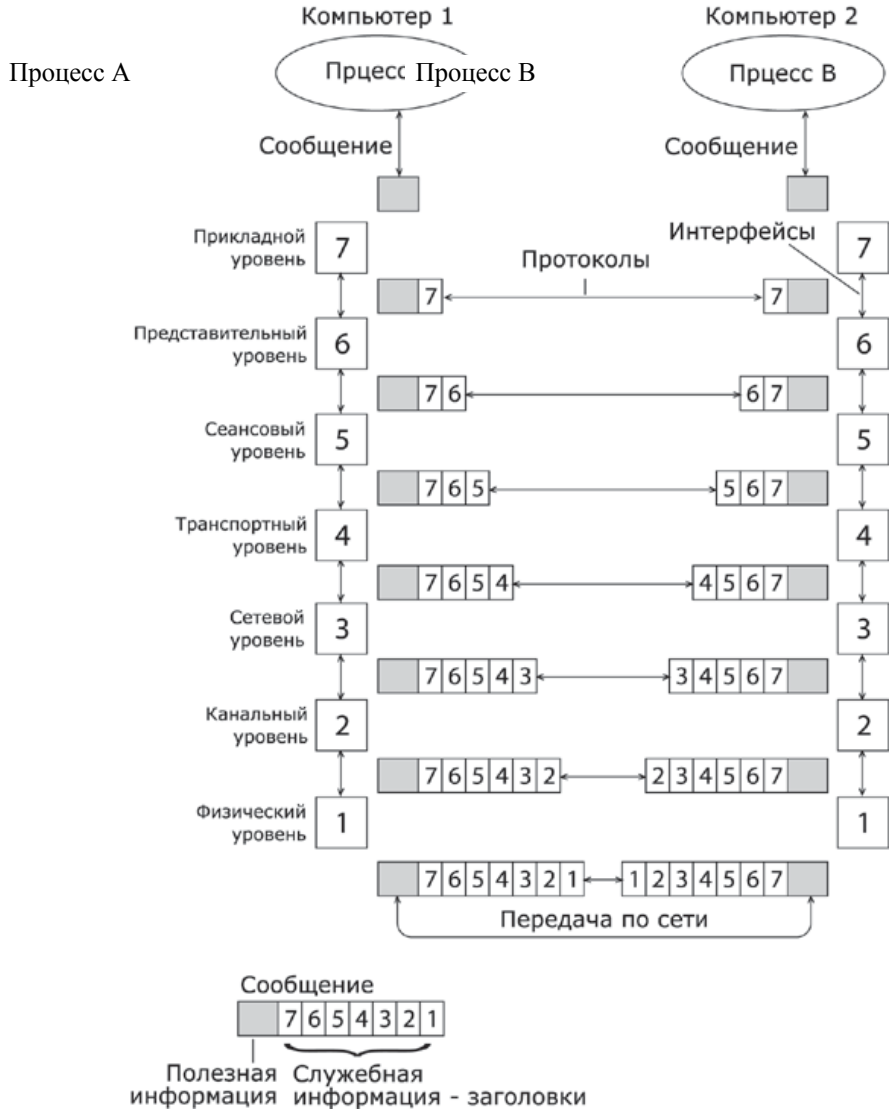

Рис. 6.6. Модель взаимодействия открытых систем ISO/OSI

Модель OSI описывает только системные средства взаимодействия, реализуемые операционной системой, системными утилитами и аппаратными средствами. Модель не включает средства взаимодействия приложений конечных пользователей. Собственные протоколы взаимодействия приложения реализуют, обращаясь к системным средствам. Поэтому необходимо различать уровень взаимодействия приложений и прикладной уровень.

Следует также иметь в виду, что приложение может взять на себя функции некоторых верхних уровней модели OSI. Например, некоторые СУБД имеют встроенные средства удаленного доступа к файлам.

В этом случае приложение, выполняя доступ к удаленным ресурсам, не использует системную файловую службу; оно обходит верхние уровни модели OSI и обращается напрямую к системным средствам, ответственным за транспортировку сообщений по сети, которые располагаются на нижних уровнях модели OSI.

Итак, пусть приложение обращается с запросом к прикладному уровню, например к файловой службе. На основании этого запроса программное обеспечение прикладного уровня формирует сообщение стандартного формата. Обычное сообщение состоит из заголовка и поля данных. Заголовок содержит служебную информацию, которую необходимо передать через сеть прикладному уровню машины-адресата, чтобы сообщить ему, какую работу надо выполнить. В нашем случае заголовок, очевидно, должен содержать информацию о местонахождении файла и о типе операции, которую необходимо выполнить. Поле данных сообщения может быть пустым или содержать какие-либо данные, например те, которые необходимо записать в удаленный файл. Но для того чтобы доставить эту информацию по назначению, предстоит решить еще много задач, ответственность за которые несут нижележащие уровни.

После формирования сообщения прикладной уровень направляет его вниз по стеку представительному уровню. Протокол представительного уровня на основании информации, полученной из заголовка

прикладного уровня, выполняет требуемые действия и добавляет к сообщению собственную служебную информацию – заголовок представительного уровня, в котором содержатся указания для протокола представительного уровня машины-адресата. Полученное в результате сообщение передается вниз сеансовому уровню, который в свою очередь добавляет свой заголовок, и т.д. (Некоторые протоколы помещают служебную информацию не только в начале сообщения в виде заголовка, но и в конце, в виде так называемого "концевика".) Наконец, сообщение достигает нижнего, физического уровня, который, собственно, и передает его по линиям связи машине-адресату. К этому моменту сообщение "обрастает" заголовками всех уровней (рис. 6.7).

Когда сообщение по сети поступает на машину-адресат, оно принимается ее физическим уровнем и последовательно перемещается вверх с уровня на уровень. Каждый уровень анализирует и обрабатывает заголовок своего уровня, выполняя соответствующие данному уровню функции, а затем удаляет этот заголовок и передает сообщение вышележащему уровню.

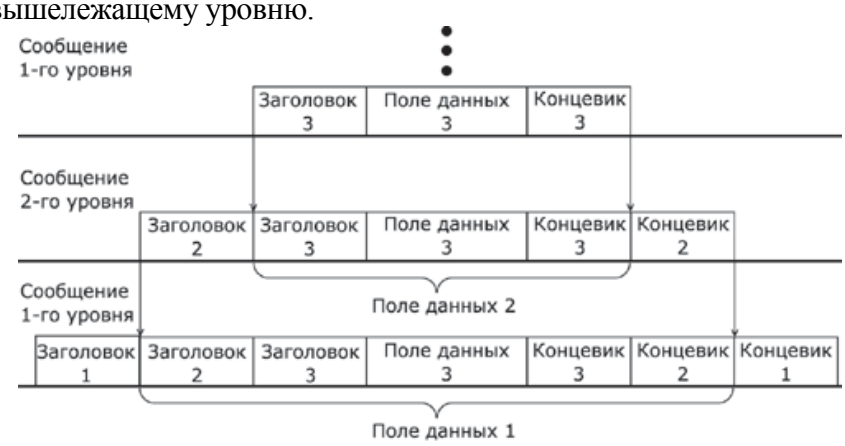

Рис. 6.7. Вложенность сообщений различных уровней

Наряду с термином *сообщение* (message) существуют и другие термины, применяемые сетевыми специалистами для обозначения единиц данных в процедурах обмена. В стандартах ISO для обозначения единиц данных, с которыми имеют дело протоколы разных уровней, используется общее название *протокольный блок данных* (Protocol Data Unit, PDU). Для обозначения блоков данных определенных уровней
часто используются специальные названия: *кадр* (frame), *пакет* (packet), *дейтаграмма* (datagram), *сегмент* (segment).

Физический уровень (Physical layer) имеет дело с передачей битов по физическим каналам связи, таким, как коаксиальный кабель, витая пара, оптоволоконный кабель или цифровой территориальный канал. К этому уровню имеют отношение характеристики физических сред передачи данных, такие как полоса пропускания, помехозащищенность, волновое сопротивление и другие. На этом же уровне определяются характеристики электрических сигналов, передающих дискретную информацию, такую как крутизна фронтов импульсов, уровни напряжения или тока передаваемого сигнала, тип кодирования, скорость передачи сигналов. Кроме того, здесь стандартизируются типы разъемов и назначение каждого контакта.

Физический уровень:

- передача битов по физическим каналам;
- формирование электрических сигналов;
- кодирование информации;
- синхронизация;
- модуляция.

Физический уровень реализуется аппаратно.

Функции физического уровня реализуются во всех устройствах, подключенных к сети. Со стороны компьютера функции физического уровня выполняются сетевым адаптером или последовательным портом.

Примером протокола физического уровня может служить спецификация 10Base-T технологии Ethernet, которая определяет в качестве используемого кабеля неэкранированную витую пару категории 3 с волновым сопротивлением 100 Ом, разъем RJ-45, максимальную длину физического сегмента 100 метров, манчестерский код для представления данных в кабеле, а также некоторые другие характеристики среды и электрических сигналов.

Канальный уровень. На физическом уровне просто пересылаются биты. При этом не учитывается, что в тех сетях, в которых линии связи используются (разделяются) попеременно несколькими парами взаимодействующих компьютеров, физическая среда передачи может быть занята. Поэтому одной из задач канального уровня (Data Link layer) является проверка доступности среды передачи. Другая задача канального уровня – реализация механизмов обнаружения и коррекции ошибок. Для этого на канальном уровне биты группируются в наборы, называемые кадрами (frames). Канальный уровень обеспечивает корректность передачи каждого кадра, помещая специальную последовательность бит в начало и конец каждого кадра, для его выделения, а также вычисляет контрольную сумму, обрабатывая все байты кадра определенным способом, и добавляет контрольную сумму к кадру. Когда кадр приходит по сети, получатель снова вычисляет контрольную сумму полученных данных и сравнивает результат с контрольной суммой из кадра. Если они совпадают, кадр считается правильным и принимается. Если же контрольные суммы не совпадают, то фиксируется ошибка. Канальный уровень может не только обнаруживать ошибки, но и исправлять их за счет повторной передачи поврежденных кадров. Необходимо отметить, что функция исправления ошибок для канального уровня не является обязательной, поэтому в некоторых протоколах этого уровня она отсутствует, например в Ethernet и frame relay.

Функции канального уровня. Надежная доставка пакета:

- между двумя соседними станциями в сети с произвольной топологией;
- между любыми станциями в сети с типовой топологией:
  - проверка доступности разделяемой среды;
  - выделение кадров из потока данных, поступающих по сети; формирование кадров при отправке данных;
  - подсчет и проверка контрольной суммы.

Канальный уровень реализуется программно-аппаратно.

В протоколах канального уровня, используемых в локальных сетях, заложена определенная структура связей между компьютерами и способы их адресации. Хотя канальный уровень и обеспечивает доставку кадра между любыми двумя узлами локальной сети, он это делает только в сети с определенной топологией связей, именно той топологией, для которой он был разработан. К таким типовым топологиям, поддерживаемым протоколами канального уровня локальных сетей, относятся "общая шина", "кольцо" и "звезда", а также структуры, полученные из них с помощью мостов и коммутаторов. Примерами протоколов канального уровня являются протоколы Ethernet, Token Ring, FDDI, 100VG-AnyLAN.

В локальных сетях протоколы канального уровня используются компьютерами, мостами, коммутаторами и маршрутизаторами. В компьютерах функции канального уровня реализуются совместными усилиями сетевых адаптеров и их драйверов.

В глобальных сетях, которые редко обладают регулярной топологией, канальный уровень часто обеспечивает обмен сообщениями только между двумя соседними компьютерами, соединенными индивидуальной линией связи. Примерами протоколов "точка-точка" (как часто называют такие протоколы) могут служить широко распространенные протоколы PPP и LAP-B. В таких случаях для доставки сообщений между конечными узлами через всю сеть используются средства сетевого уровня. Именно так организованы сети X.25. Иногда в глобальных сетях функции канального уровня в чистом виде выделить трудно, так как в одном и том же протоколе они объединяются с функциями сетевого уровня. Примерами такого подхода могут служить протоколы технологий ATM и frame relay.

В целом канальный уровень представляет собой весьма мощный набор функций по пересылке сообщений между узлами сети. В некоторых случаях протоколы канального уровня оказываются самодостаточными транспортными средствами, и тогда поверх них могут работать непосредственно протоколы прикладного уровня или приложения, без привлечения средств сетевого и транспортного уровней. Например, существует реализация протокола управления сетью SNMP непосредственно поверх Ethernet, хотя стандартно этот протокол работает поверх сетевого протокола IP и транспортного протокола UDP. Естественно, что применение такой реализации будет ограниченным – она не подходит для составных сетей разных технологий, например Ethernet и X.25, и даже для такой сети, в которой во всех сегментах применяется Ethernet, но между сегментами существуют петлевидные связи. А вот в двухсегментной сети Ethernet, объединенной мостом, реализация SNMP над канальным уровнем будет вполне работоспособна.

Тем не менее, для обеспечения качественной транспортировки сообщений в сетях любых топологий и технологий функций канального уровня оказывается недостаточно, поэтому в модели OSI решение этой задачи возлагается на два следующих уровня – сетевой и транспортный.

Канальный уровень обеспечивает передачу пакетов данных, поступающих от протоколов верхних уровней, узлу назначения, адрес которого также указывает протокол верхнего уровня. Протоколы канального уровня оформляют переданные им пакеты в кадры собственного формата, помещая указанный адрес назначения в одно из полей такого кадра, а также сопровождая кадр контрольной суммой. Протокол канального уровня имеет локальный смысл, он предназначен для доставки кадров данных, как правило, в пределах сетей с простой топологией связей и однотипной или близкой технологией, например в односегментных сетях Ethernet или же в многосегментных сетях Ethernet и Token Ring иерархической топологии, разделенных только мостами и коммутаторами. Во всех этих конфигурациях адрес назначения имеет локальный смысл для данной сети и не изменяется при прохождении кадра от узла-источника к узлу назначения. Возможность передавать данные между локальными сетями разных технологий связана с тем, что в этих технологиях используются адреса одинакового формата, к тому же производители сетевых адаптеров обеспечивают уникальность адресов независимо от технологии.

Другой областью действия протоколов канального уровня являются связи типа "точка-точка" глобальных сетей, когда протокол канального уровня ответственен за доставку кадра непосредственному соседу. Адрес в этом случае не имеет принципиального значения, а на первый план выходит способность протокола восстанавливать искаженные и утерянные кадры, так как плохое качество территориальных каналов, особенно коммутируемых телефонных, часто требует выполнения подобных действий. Если же перечисленные выше условия не соблюдаются, например связи между сегментами Ethernet имеют петлевидную структуру, либо объединяемые сети используют различные способы адресации, как в сетях Ethernet и X.25, то протокол канального уровня не может в одиночку справиться с задачей передачи кадра между узлами и требует помощи протокола сетевого уровня.

Сетевой уровень (Network layer) служит для образования единой транспортной системы, объединяющей несколько сетей, причем эти сети могут использовать различные принципы передачи сообщений между конечными узлами и обладать произвольной структурой связей. Функции сетевого уровня достаточно разнообразны. Рассмотрим их на примере объединения локальных сетей.

Протоколы канального уровня локальных сетей обеспечивают доставку данных между любыми узлами только в сети с соответствующей типовой топологией, например топологией иерархической звезды. Это жесткое ограничение, которое не позволяет строить сети с развитой структурой, например сети, объединяющие несколько сетей предприятия в единую сеть, или высоконадежные сети, в которых существуют избыточные связи между узлами. Можно было бы усложнять протоколы канального уровня для поддержания петлевидных избыточных связей, но принцип разделения обязанностей между уровнями приводит к другому решению. Чтобы, с одной стороны, сохранить простоту процедур передачи данных для типовых топологий, а с другой – допустить использование произвольных топологий, вводится дополнительный сетевой уровень.

На сетевом уровне сам термин "сеть" наделяют специфическим значением. В данном случае под сетью понимается совокупность компьютеров, соединенных между собой в соответствии с одной из стандартных типовых топологий и использующих для передачи данных один из протоколов канального уровня, определенный для этой топологии.

Внутри сети доставка данных обеспечивается соответствующим канальным уровнем, а вот доставкой данных между сетями занимается сетевой уровень, который и поддерживает возможность правильного выбора маршрута передачи сообщения даже в том случае, когда структура связей между составляющими сетями имеет характер, отличный от принятого в протоколах канального уровня.

Сети соединяются между собой специальными устройствами, называемыми маршрутизаторами. Маршрутизатор – это устройство, которое собирает информацию о топологии межсетевых соединений и пересылает пакеты сетевого уровня в сеть назначения. Чтобы передать сообщение от отправителя, находящегося в одной сети, получателю, находящемуся в другой сети, нужно совершить некоторое количество транзитных передач между сетями, или *хопов* (от слова hop – прыжок), каждый раз выбирая подходящий маршрут. Таким образом, маршрут представляет собой последовательность маршрутизаторов, через которые проходит пакет.

Сетевой уровень – доставка пакета:

- между любыми двумя узлами сети с произвольной топологией;
- между любыми двумя сетями в составной сети;
- сеть – совокупность компьютеров, использующих для обмена данными единую сетевую технологию;
- маршрут – последовательность прохождения пакетом маршрутизаторов в составной сети.

На рис. 6.8 показаны четыре сети, связанные тремя маршрутизаторами. Между узлами A и B данной сети пролегает два маршрута: первый – через маршрутизаторы 1 и 3, а второй – через маршрутизаторы 1, 2 и 3.

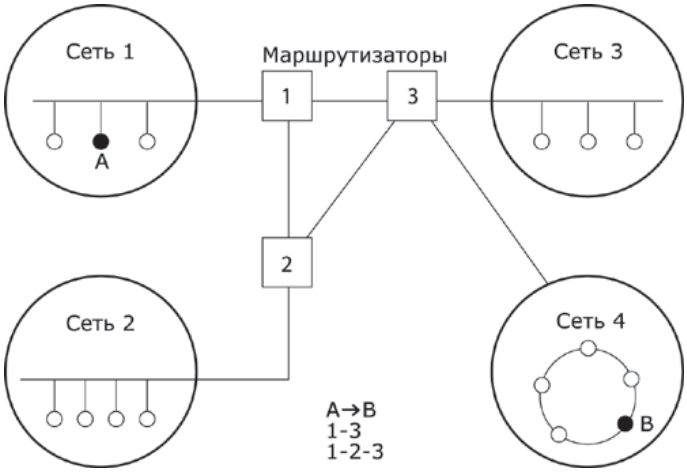

Рис. 6.8. Пример составной сети

Проблема выбора наилучшего пути называется маршрутизацией, и ее решение является одной из главных задач сетевого уровня. Эта проблема осложняется тем, что самый короткий путь – не всегда самый лучший. Часто критерием при выборе маршрута является время передачи данных; оно зависит от пропускной способности каналов связи и интенсивности трафика, которая может с течением времени изменяться. Некоторые алгоритмы маршрутизации пытаются приспособиться к изменению нагрузки, в то время как другие принимают решения на основе средних показателей за длительное время. Выбор маршрута может осуществляться и по другим критериям, таким как надежность передачи.

В общем случае функции сетевого уровня шире, чем функции передачи сообщений по связям с нестандартной структурой, которые мы рассмотрели на примере объединения нескольких локальных сетей. Сетевой уровень также решает задачи согласования разных технологий, упрощения адресации в крупных сетях и создания надежных и гибких барьеров на пути нежелательного трафика между сетями.

Сообщения сетевого уровня принято называть пакетами (packet). При организации доставки пакетов на сетевом уровне используется понятие "номер сети". В этом случае адрес получателя состоит из старшей части – номера сети и младшей – номера узла в этой сети. Все узлы одной сети должны иметь одну и ту же старшую часть адреса, поэтому термину "сеть" на сетевом уровне можно дать и другое, более формальное, определение: сеть – это совокупность узлов, сетевой адрес которых содержит один и тот же номер сети.

На сетевом уровне определяется два вида протоколов. Первый вид – *сетевые протоколы* (routed protocols) – реализуют продвижение пакетов через сеть. Именно эти протоколы обычно имеют в виду, когда говорят о протоколах сетевого уровня. Однако часто к сетевому уровню относят и другой вид протоколов, называемых протоколами обмена маршрутной информацией или просто *протоколами маршрутизации* (routing protocols). С помощью этих протоколов маршрутизаторы собирают информацию о топологии межсетевых соединений. Протоколы сетевого уровня реализуются программными модулями операционной системы, а также программными и аппаратными средствами маршрутизаторов.

На сетевом уровне работают протоколы еще одного типа, которые отвечают за отображение адреса узла, используемого на сетевом уровне, в локальный адрес сети. Такие протоколы часто называют *протоколами разрешения адресов* – Address Resolution Protocol, ARP. Иногда их относят не к сетевому уровню, а к канальному, хотя тонкости классификации не изменяют сути.

Примерами протоколов сетевого уровня являются протокол межсетевого взаимодействия IP стека TCP/IP и протокол межсетевого обмена пакетами IPX стека Novell.

Транспортный уровень. На пути от отправителя к получателю пакеты могут быть искажены или утеряны. Хотя некоторые приложения имеют собственные средства обработки ошибок, существуют и такие, которые предпочитают сразу иметь дело с надежным соединением. Транспортный уровень (Transport layer) обеспечивает приложениям или верхним уровням стека – прикладному и сеансовому – передачу данных с той степенью надежности, которая им требуется. Модель OSI определяет пять классов сервиса, предоставляемых транспортным уровнем. Эти виды сервиса отличаются качеством предоставляемых услуг: срочностью, возможностью восстановления прерванной связи, наличием средств мультиплексирования нескольких соединений между различными прикладными протоколами через общий транс-

портный протокол, а главное – способностью к обнаружению и исправлению ошибок передачи, таких как искажение, потеря и дублирование пакетов.

Выбор класса сервиса транспортного уровня определяется, с одной стороны, тем, в какой степени задача обеспечения надежности решается самими приложениями и протоколами более высоких, чем транспортный, уровней, а с другой стороны, зависит от того, насколько надежной является система транспортировки данных в сети, обеспечиваемая уровнями, расположенными ниже транспортного – сетевым, канальным и физическим. Так, например, если качество каналов передачи связи очень высокое и вероятность наличия ошибок, не обнаруженных протоколами более низких уровней, невелика, стоит воспользоваться одним из облегченных сервисов транспортного уровня, не обремененных многочисленными проверками, квитированием и другими приемами повышения надежности. Если же транспортные средства нижних уровней изначально очень ненадежны, то целесообразно обратиться к наиболее развитому сервису транспортного уровня, который работает, используя максимум средств для обнаружения и устранения ошибок, – с помощью предварительного установления логического соединения, отслеживания доставки сообщений по контрольным суммам и циклической *нумерации* пакетов, установления тайм-аутов доставки и т.п.

Транспортный уровень – обеспечение доставки информации с требуемым качеством между любыми узлами сети:

- разбивка сообщения сеансового уровня на пакеты, их нумерация;
- буферизация принимаемых пакетов;
- упорядочивание прибывающих пакетов;
- адресация прикладных процессов;
- управление потоком.

Как правило, все протоколы, начиная с транспортного уровня и выше, реализуются программными средствами конечных узлов сети – компонентами их сетевых операционных систем. В качестве примера транспортных протоколов можно привести протоколы TCP и UDP стека TCP/IP и протокол SPX стека Novell.

Протоколы четырех нижних уровней обобщенно называют сетевым транспортом или транспортной подсистемой, так как они полностью решают задачу транспортировки сообщений с заданным уровнем качества в составных сетях с произвольной топологией и различными технологиями. Остальные три верхних уровня решают задачи предоставления прикладных сервисов на основании имеющейся транспортной подсистемы.

Сеансовый уровень (Session layer) обеспечивает управление диалогом: фиксирует, какая из сторон является активной в настоящий момент, предоставляет средства синхронизации. Последние позволяют вставлять контрольные точки в длинные передачи, чтобы в случае отказа можно было вернуться назад к последней контрольной точке, а не начинать все сначала. На практике немногие приложения используют сеансовый уровень, и он редко реализуется в виде отдельных протоколов, хотя функции этого уровня часто объединяют с функциями прикладного уровня и реализуют в одном протоколе.

Сеансовый уровень – управление диалогом объектов прикладного уровня:

- установление способа обмена сообщениями (дуплексный или полудуплексный);
- синхронизация обмена сообщениями;
- организация "контрольных точек" диалога.

Представительный уровень (Presentation layer) имеет дело с формой представления передаваемой по сети информации, не меняя при этом ее содержания. За счет уровня представления информация, передаваемая прикладным уровнем одной системы, всегда понятна прикладному уровню другой системы. С помощью средств данного уровня протоколы прикладных уровней могут преодолеть синтаксические различия в представлении данных или же различия в кодах символов, например в кодах ASCII и EBCDIC. На этом уровне может выполняться шифрование и дешифрование данных, благодаря которому секретность обмена данными обеспечивается сразу для всех прикладных служб. Примером такого протокола является протокол Secure Socket Layer (SSL), который обеспечивает секретный обмен сообщениями для протоколов прикладного уровня стека TCP/IP.

Уровень представления – согласовывает представление (синтаксис) данных при взаимодействии двух прикладных процессов:

- преобразование данных из внешнего формата во внутренний;
- шифрование и расшифровка данных.

Прикладной уровень (Application layer) – это в действительности просто набор разнообразных протоколов, с помощью которых пользователи сети получают доступ к разделяемым ресурсам, таким как файлы, принтеры или гипертекстовые Web-страницы, а также организуют совместную работу, напри-

мер с помощью протокола электронной почты. Единица данных, которой оперирует прикладной уровень, обычно называется сообщением (message).

Прикладной уровень – набор всех сетевых сервисов, которые предоставляет система конечному пользователю:

- идентификация, проверка прав доступа;
- принт- и файл-сервис, почта, удаленный доступ...

Существует очень много различных служб прикладного уровня. Приведем в качестве примера хотя бы несколько наиболее распространенных реализаций файловых служб: NCP в операционной системе Novell NetWare, SMB в Microsoft Windows NT, NFS, FTP и TFTP, входящие в стек TCP/IP.

6.4. СЕТЕЗАВИСИМЫЕ И СЕТЕНЕЗАВИСИМЫЕ УРОВНИ

Функции всех уровней модели OSI могут быть отнесены к одной из двух групп: либо к функциям, зависящим от конкретной технической реализации сети, либо к функциям, ориентированным на работу с приложениями.

Три нижних уровня – физический, канальный и сетевой – являются сетезависимыми, то есть протоколы этих уровней тесно связаны с технической реализацией сети и используемым коммуникационным оборудованием. Например, переход на оборудование FDDI означает полную смену протоколов физического и канального уровней во всех узлах сети.

Три верхних уровня – прикладной, представительный и сеансовый – ориентированы на приложения и мало зависят от технических особенностей построения сети. На протоколы этих уровней не влияют какие бы то ни было изменения в топологии сети, замена оборудования или переход на другую сетевую технологию. Так, переход от Ethernet к высокоскоростной технологии 100VG-AnyLAN не потребует никаких изменений в программных средствах, реализующих функции прикладного, представительного и сеансового уровней.

Транспортный уровень является промежуточным, он скрывает все детали функционирования нижних уровней от верхних. Это позволяет разрабатывать приложения, не зависящие от технических средств непосредственной транспортировки сообщений.

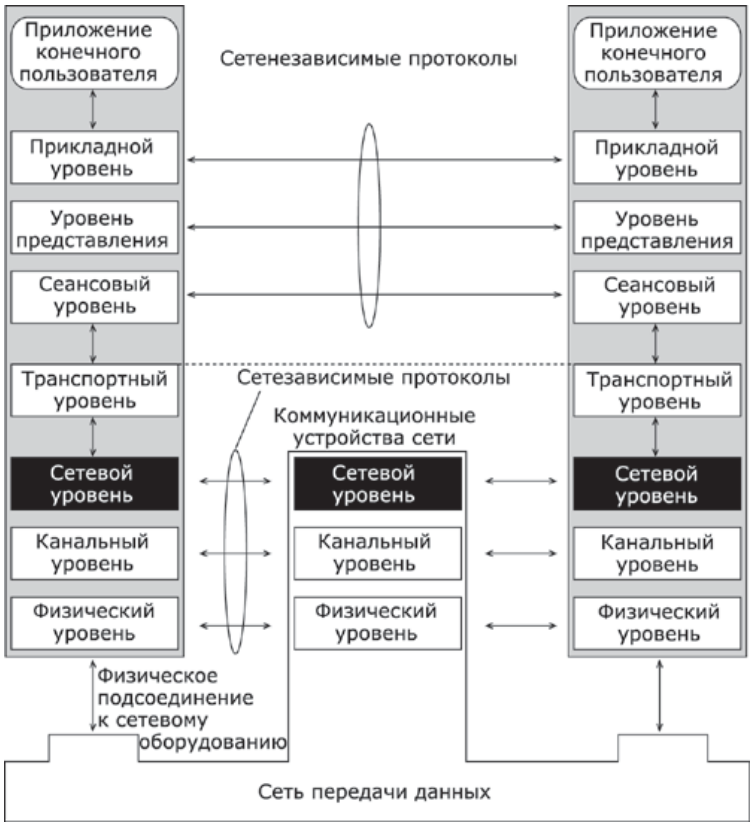

Рис. 6.9. Сетезависимые и сетенезависимые уровни модели OSI

На рис. 6.9 показаны уровни модели OSI, на которых работают различные элементы сети. Компьютер с установленной на нем сетевой ОС взаимодействует с другим компьютером с помощью протоколов всех семи уровней. Это взаимодействие компьютеры осуществляют опосредованно, через различные

коммуникационные устройства: концентраторы, модемы, мосты, коммутаторы, маршрутизаторы, мультиплексоры. В зависимости от типа коммуникационное устройство может работать либо только на физическом уровне (повторитель), либо на физическом и канальном (мост), либо на физическом, канальном и сетевом, иногда захватывая и транспортный уровень (маршрутизатор). На рис. 6.10 показано соответствие функций различных коммуникационных устройств уровням модели OSI.

Модель OSI представляет хотя и очень важную, но только одну из многих моделей коммуникаций. Эти модели и связанные с ними стеки протоколов могут отличаться количеством уровней, их функциями, форматами сообщений, службами, поддерживаемыми на верхних уровнях, и прочими параметрами.

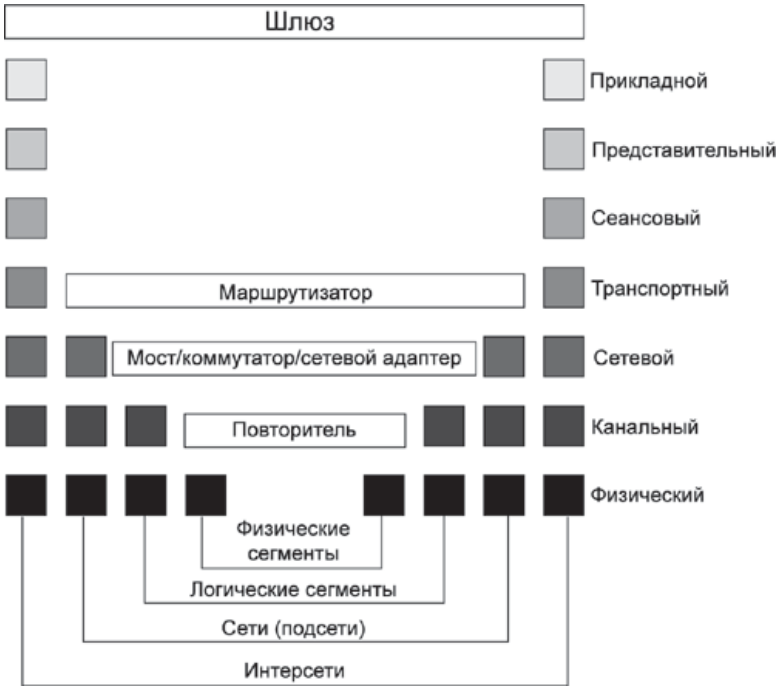

Рис. 6.10. Соответствие функций различных устройств сети уровням модели OSI

7. Понятие "открытая система"

Модель OSI, как следует из ее названия (Open System Interconnection), описывает взаимосвязи открытых систем. Что же такое открытая система?

В широком смысле открытой системой может быть названа любая система (компьютер, вычислительная сеть, ОС, программный пакет, другие аппаратные и программные продукты), построенная в соответствии с открытыми *спецификациями*.

Напомним, что под термином "спецификация" (в вычислительной технике) понимают формализованное описание аппаратных или программных компонентов, способов их функционирования, взаимодействия с другими компонентами, условий эксплуатации, ограничений и особых характеристик. Понятно, что не всякая спецификация является стандартом.

Под *открытыми спецификациями* понимаются опубликованные, общедоступные спецификации, соответствующие стандартам и принятые в результате достижения согласия после всестороннего обсуждения всеми заинтересованными сторонами.

Использование при разработке систем открытых спецификаций позволяет третьим сторонам разрабатывать для этих систем различные аппаратные или программные средства расширения и модификации, а также создавать программно-аппаратные комплексы из продуктов разных производителей.

Для реальных систем полная открытость является недостижимым идеалом. Как правило, даже в системах, называемых открытыми, этому определению соответствуют лишь некоторые части, поддерживающие внешние интерфейсы. Например, открытость семейства операционных систем Unix заключается, кроме всего прочего, в наличии стандартизованного программного интерфейса между ядром и приложениями, что позволяет легко переносить приложения из среды одной версии Unix в среду другой версии. Еще одним примером частичной открытости является применение в достаточно закрытой операционной системе Novell NetWare открытого интерфейса Open Driver Interface (ODI) для включения в

систему драйверов сетевых адаптеров производства независимых компаний. Чем больше открытых спецификаций использовано при разработке системы, тем более открытой она является.

Модель OSI касается только одного аспекта открытости, а именно открытости средств взаимодействия устройств, связанных в вычислительную сеть. Здесь под открытой системой понимается сетевое устройство, готовое взаимодействовать с другими сетевыми устройствами с использованием стандартных правил, определяющих формат, содержание и значение принимаемых и отправляемых сообщений.

Если две сети построены с соблюдением принципов открытости, то это дает следующие преимущества:

- возможность построения сети из аппаратных и программных средств различных производителей, придерживающихся одного и того же стандарта;
- возможность безболезненной замены одних компонентов сети другими, что позволяет сети развиваться с минимальными затратами;
- возможность легкого сопряжения одной сети с другой;
- простота освоения и обслуживания сети.

Ярким примером открытой системы является сеть Internet. Эта сеть развивалась в полном соответствии с требованиями, предъявляемыми к открытым системам. В разработке ее стандартов принимали участие тысячи специалистов-пользователей из различных университетов, научных организаций и фирм-производителей вычислительной аппаратуры и программного обеспечения, работающих в разных странах. Само название стандартов, определяющих работу Internet – Request For Comments (RFC, что можно перевести как "запрос на комментарии", – говорит об открытом характере принимаемых стандартов. В результате сеть Internet объединила в себе разнообразное оборудование и программное обеспечение огромного количества сетей, разбросанных по всему миру.

7.1. Модульность и стандартизация

Модульность – это одно из неотъемлемых свойств вычислительных сетей. Модульность проявляется не только в многоуровневом представлении коммуникационных протоколов в конечных узлах сети, хотя это, безусловно, важная и принципиальная особенность сетевой архитектуры. Сеть состоит из огромного числа различных модулей – компьютеров, сетевых адаптеров, мостов, маршрутизаторов, модемов, операционных систем и модулей приложений.

Разнообразные требования, предъявляемые предприятиями к компьютерным сетям, привели к появлению многочисленных и разнообразных устройств и программ для построения сети. Эти продукты отличаются не только основными функциями (имеются в виду функции, выполняемые, например, повторителями, мостами или программными редиректорами), но и многочисленными вспомогательными функциями, предоставляющими пользователям или администраторам дополнительные удобства, такие, как автоматизированное конфигурирование параметров устройства, автоматическое обнаружение и устранение некоторых неисправностей, возможность программного изменения связей в сети и т.п. Разнообразие увеличивается также потому, что многие устройства и программы отличаются сочетаниями тех или иных основных и дополнительных функций. Существуют, например, устройства, объединяющие в себе основные возможности коммутаторов и маршрутизаторов, к которым добавляется еще и набор некоторых дополнительных функций, характерный только для данного продукта.

В результате не существует компании, которая смогла бы обеспечить производство полного набора всех типов и подтипов оборудования и программного обеспечения, необходимого для построения сети. Но, так как все компоненты сети должны работать согласованно, потребовалось принимать многочисленные стандарты, которые, если не во всех, то хотя бы в большинстве случаев, гарантировали бы совместимость оборудования и программ различных фирм-изготовителей.

Таким образом, понятия "модульность" и "стандартизация" в сетях неразрывно связаны, и модульный подход только тогда дает преимущества, когда он сопровождается следованием стандартам.

В результате открытый характер стандартов и спецификаций важен не только для коммуникационных протоколов, но и для всех многочисленных функций разнообразных устройств и программ, выпускаемых для построения сети. Нужно отметить, что большинство принимаемых сегодня стандартов носит открытый характер. Время закрытых систем, точные спецификации на которые были известны только фирме-производителю, прошло. Стало очевидно, что возможность взаимодействия с продуктами конкурентов не снижает, а наоборот, повышает ценность изделия, так как его можно применить в большем количестве работающих сетей, построенных на основе продуктов разных изготовителей. Поэтому

даже компании, ранее выпускавшие весьма закрытые системы, такие как IBM, Novell или Microsoft, сегодня активно участвуют в разработке открытых стандартов и применяют их в своих продуктах.

Сегодня в секторе сетевого оборудования и программ с совместимостью продуктов разных производителей сложилась следующая ситуация. Практически все продукты, как программные, так и аппаратные, совместимы по функциям и свойствам, которые были реализованы уже достаточно давно, и соответствующие стандарты разработаны и приняты, по крайней мере, три-четыре года назад. В то же время очень часто принципиально новые устройства, протоколы и свойства оказываются несовместимыми даже у ведущих производителей. Такая картина характерна не только для тех устройств или функций, стандарты на которые еще не успели принять (это естественно), но и для устройств, стандарты на которые существуют уже несколько лет. Совместимость достигается только после того, как все производители реализуют соответствующий стандарт в своих изделиях, причем одинаковым образом.

7.2. Источники стандартов

Работы по стандартизации вычислительных сетей ведутся большим количеством организаций.

В зависимости от статуса организаций различают следующие виды стандартов:

- *стандарты отдельных фирм* (например, стек протоколов DECnet компании Digital Equipment или графический интерфейс OPEN LOOK для Unix-систем компании Sun);
- *стандарты специальных комитетов и объединений*, создаваемых несколькими фирмами, например стандарты технологии ATM, разрабатываемые специально созданным объединением ATM Forum, насчитывающим около 100 коллективных участников, или стандарты союза Fast Ethernet Alliance по разработке стандартов 100 Мбит Ethernet;
- *национальные стандарты*, например стандарт FDDI, один из многочисленных стандартов, разработанных Американским национальным институтом стандартов (ANSI), или стандарты безопасности для операционных систем, разработанные Национальным центром компьютерной безопасности (NCSC) Министерства обороны США;
- *международные стандарты*, например модель и стек коммуникационных протоколов Международной организации по стандартизации (ISO), многочисленные стандарты Международного союза электросвязи (ITU), в том числе стандарты на сети с коммутацией пакетов X.25, сети frame relay, ISDN, модемы и многие другие.

Некоторые стандарты, непрерывно развиваясь, могут переходить из одной категории в другую. В частности, фирменные стандарты на продукцию, получившую широкое распространение, обычно становятся международными стандартами де-факто, так как вынуждают производителей из разных стран следовать фирменным стандартам, чтобы обеспечить совместимость своих изделий с этими популярными продуктами. Например, из-за феноменального успеха персонального компьютера компании IBM фирменный стандарт на архитектуру IBM PC стал международным стандартом де-факто.

Более того, ввиду широкого распространения некоторые фирменные стандарты становятся основой для национальных и международных стандартов де-юре. Например, стандарт Ethernet, первоначально разработанный компаниями Digital Equipment, Intel и Xerox, через некоторое время и в несколько измененном виде был принят как национальный стандарт IEEE 802.3, а затем организация ISO утвердила его в качестве международного стандарта ISO 8802.3.

Далее приводятся краткие сведения об организациях, наиболее активно и успешно занимающихся разработкой стандартов в области вычислительных сетей.

Международная организация по стандартизации (International Organization for Standardization, ISO, часто называемая также International Standards Organization) представляет собой ассоциацию ведущих национальных организаций по стандартизации разных стран. Главным достижением ISO стала модель взаимодействия открытых систем OSI, которая в настоящее время является концептуальной основой стандартизации в области вычислительных сетей. В соответствии с моделью OSI этой организацией был разработан стандартный стек коммуникационных протоколов OSI.

Международный союз электросвязи (International Telecommunica-tions Union, ITU) – организация, которая в настоящее время является специализированным органом Организации Объединенных Наций. Наиболее значительную роль в стандартизации вычислительных сетей играет постоянно действующий в рамках этой организации Международный консультативный комитет по телефонии и телеграфии (МККТТ) (Consultative Committee on International Telegraphy and Telephony, CCITT). В результате проведенной в 1993 году реорганизации ITU CCITT несколько изменил направление своей деятельности и сменил название – теперь он называется сектором телекоммуникационной стандартизации ITU (ITU Telecommunication Standardization Sector, ITU-T). Основу деятельности ITU-T составляет разработка

международных стандартов в области телефонии, телематических служб (электронной почты, факсимильной связи, телетекста, телекса и т.д.), передачи данных, аудио- и видеосигналов. За годы своей деятельности ITU-T выпустил огромное количество рекомендаций-стандартов. Свою работу ITU-T строит на изучении опыта различных организаций, а также на результатах собственных исследований. Раз в четыре года издаются труды ITU-T в виде так называемой "Книги", которая на самом деле представляет собой целый набор обычных книг, сгруппированных в выпуски, которые, в свою очередь, объединяются в тома. Каждый том и выпуск содержат логически взаимосвязанные рекомендации. Например, том III Синей Книги содержит рекомендации для цифровых сетей с интеграцией услуг (ISDN), а весь том VIII (за исключением выпуска VIII.1, который содержит рекомендации серии V для передачи данных по телефонной сети) посвящен рекомендациям серии X: X.25 – для сетей с коммутацией пакетов, X.400 – для систем электронной почты, X.500 – для глобальной справочной службы и многим другим.

Институт инженеров по электротехнике и радиоэлектронике (Institute of Electrical and Electronics Engineers, IEEE) – национальная организация США, определяющая сетевые стандарты. В 1981 году рабочая группа 802 этого института сформулировала основные требования, которым должны удовлетворять локальные вычислительные сети. Группа 802 определила множество стандартов, из них самыми известными являются стандарты 802.1, 802.2, 802.3 и 802.5, которые описывают общие понятия, используемые в области локальных сетей, а также стандарты на два нижних уровня сетей Ethernet и Token Ring.

Европейская ассоциация производителей компьютеров (European Computer Manufacturers Association, ECMA) – некоммерческая организация, активно сотрудничающая с ITU-T и ISO, занимается разработкой стандартов и технических обзоров, относящихся к компьютерной и коммуникационной технологиям. Известна своим стандартом ECMA-101, используемым при передаче отформатированного текста и графических изображений с сохранением оригинального формата.

Ассоциация производителей компьютеров и оргтехники (Computer and Business Equipment Manufacturers Association, CBEMA) – организация американских производителей аппаратного обеспечения; аналогична европейской ассоциации ЕКМА; участвует в разработке стандартов на обработку информации и соответствующее оборудование.

Ассоциация электронной промышленности (Electronic Industries Association, EIA) – промышленно-торговая группа производителей электронного и сетевого оборудования; является национальной коммерческой ассоциацией США; проявляет значительную активность в разработке стандартов для проводов, коннекторов и других сетевых компонентов. Ее наиболее известный стандарт – RS-232C.

Министерство обороны США (Department of Defense, DoD) имеет многочисленные подразделения, занимающиеся созданием стандартов для компьютерных систем. Одной из самых известных разработок DoD является стек транспортных протоколов TCP/IP.

Американский национальный институт стандартов (American National Standards Institute, ANSI). Эта организация представляет США в Международной организации по стандартизации ISO. Комитеты ANSI занимаются разработкой стандартов в различных областях вычислительной техники. Так, комитет ANSI X3T9.5 совместно с компанией IBM осуществляет стандартизацию локальных сетей крупных ЭВМ (архитектура сетей SNA). Известный стандарт FDDI также является результатом деятельности этого комитета ANSI. В области микрокомпьютеров ANSI разрабатывает стандарты на языки программирования, интерфейс SCSI. ANSI разработал рекомендации по переносимости для языков C, FORTRAN, COBOL.

7.3. Стандарты Internet

Особую роль в выработке международных открытых стандартов играют стандарты Internet. Ввиду постоянно растущей популярности Internet, эти стандарты становятся международными стандартами "де-факто", и многие из них приобретают впоследствии статус официальных международных стандартов за счет утверждения одной из вышеперечисленных организаций, в том числе ISO и ITU-T. Существует несколько организационных подразделений, отвечающих за развитие Internet и, в частности, за стандартизацию средств Internet.

Основным из них является Internet Society (ISOC) – профессиональное сообщество, которое занимается общими вопросами эволюции и роста Internet как глобальной коммуникационной инфраструктуры. Под управлением ISOC работает Internet Architecture Board (IAB) – организация, в ведении которой находится технический контроль и координация работ для Internet. IAB координирует направление иссле-

дований и новых разработок для стека TCP/IP и является конечной инстанцией при определении новых стандартов Internet (рис. 7.1).

В IAB входят две основные группы: Internet Engineering Task Force (IETF) и Internet Research Task Force (IRTF). IETF – это инженерная группа, которая занимается решением наиболее актуальных технических проблем Internet. Именно IETF определяет спецификации, которые затем становятся стандартами Internet. В свою очередь, IRTF координирует долгосрочные исследовательские проекты по протоколам TCP/IP.

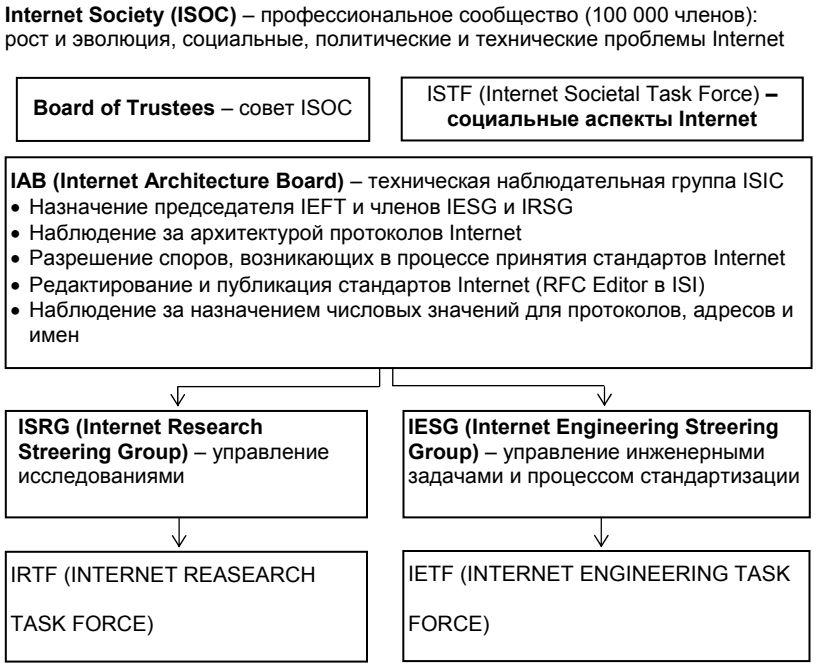

Рис. 7.1. Стандартизация Internet

Рис. 7.2. Стадии стандартизации протокола Internet

В любой организации, занимающейся стандартизацией, процесс выработки и принятия стандарта состоит из ряда обязательных этапов, которые, собственно, и составляют процедуру стандартизации. Рассмотрим эту процедуру на примере разработки стандартов Internet. (Рис. 7.2, на котором показана схема прохождения стандарта через все этапы, сам является документом RFC; заметим, что он выполнен средствами псевдографики, для того чтобы его можно было прочесть практически в любой операционной среде.)

1. Сначала в IETF представляется так называемый рабочий проект (draft) в виде, доступном для комментариев (на рисунке данный этап обозначен enter). Он публикуется в Internet, после чего широкий круг заинтересованных лиц включается в обсуждение этого документа, в него вносятся исправления, и, наконец, наступает момент, когда можно зафиксировать содержание документа. На данном этапе проекту присваивается номер RFC (возможен и другой вариант развития событий – после обсуждения рабочий проект отвергается и удаляется из Internet).

2. После присвоения номера проект приобретает статус предлагаемого стандарта (на рисунке proposed). В течение шести месяцев этот предлагаемый стандарт проходит проверку практикой, в результате в него вносятся изменения.

3. Если результаты практических исследований свидетельствуют об эффективности предлагаемого стандарта, то ему, со всеми внесенными изменениями, присваивается статус проекта стандарта (на рисунке draft std). Затем в течение как минимум четырех месяцев проходят его дальнейшие испытания "на прочность", при этом создается по крайней мере две программных реализации.

4. Если во время пребывания в ранге проекта стандарта в документ не было внесено никаких исправлений, ему может быть присвоен статус официального стандарта Internet (на рисунке standart).

Следует заметить, что все стандарты Internet носят название RFC с соответствующим порядковым номером, но далеко не все RFC являются стандартами Internet. Часто эти документы представляют собой комментарии к какому-либо стандарту или просто описания некоторой проблемы Internet.

Список утвержденных официальных стандартов Internet публикуется в виде документа RFC и доступен в Internet, например по адресу http://ds.internic.net/rfs/rfc-index.txt.

7.4. Стандартные стеки коммуникационных протоколов

Важнейшим направлением стандартизации в области вычислительных сетей является стандартизация коммуникационных протоколов. В настоящее время в сетях используется большое количество стеков коммуникационных протоколов. Наиболее популярны следующие стеки: TCP/IP, IPX/SPX, NetBIOS/SMB, DECnet, SNA, OSI.

Все эти стеки, кроме SNA на нижних уровнях – физическом и канальном, – используют одни и те же хорошо стандартизованные протоколы Ethernet, Token Ring, FDDI и ряд других, которые позволяют задействовать во всех сетях одну и ту же аппаратуру. Зато на верхних уровнях все стеки работают по своим протоколам. Эти протоколы часто не соответствуют рекомендуемому моделью OSI разбиению на уровни. В частности, функции сеансового и представительного уровня, как правило, объединены с прикладным уровнем. Такое несоответствие связано с тем, что модель OSI появилась как результат обобщения уже существующих и реально используемых стеков, а не наоборот.

Стек OSI. Следует четко различать модель OSI и стек OSI. Если модель OSI является концептуальной схемой взаимодействия открытых систем, то стек OSI представляет собой набор вполне конкретных спецификаций протоколов.

В отличие от других стеков протоколов, стек OSI полностью соответствует модели OSI, он включает спецификации протоколов для всех семи уровней взаимодействия, определенных в этой модели. На нижних уровнях стек OSI поддерживает Ethernet, Token Ring, FDDI, протоколы глобальных сетей, X.25 и ISDN, то есть использует разработанные вне стека протоколы нижних уровней, как и все другие стеки. Протоколы сетевого, транспортного и сеансового уровней стека OSI специфицированы и реализованы различными производителями, но распространены пока мало. Наиболее популярными протоколами стека OSI являются прикладные протоколы. К ним относятся: протокол передачи файлов FTAM, протокол эмуляции терминала VTP, протоколы справочной службы X.500, электронной почты X.400 и ряд других.

Протоколы стека OSI отличаются сложностью и неоднозначностью спецификаций. Эти свойства стали результатом общей политики разработчиков стека, стремившихся учесть в своих протоколах все случаи и все существующие технологии. К этому нужно еще добавить и последствия большого количества политических компромиссов, неизбежных при принятии международных стандартов по такому злободневному вопросу, как построение открытых вычислительных сетей.

Из-за своей сложности протоколы OSI требуют больших затрат вычислительной мощности центрального процессора, что делает их наиболее подходящими для мощных машин, а не для сетей персональных компьютеров.

Стек OSI – независимый от производителей международный стандарт. Его поддерживает правительство США в своей программе GOSIP, в соответствии с которой все компьютерные сети, устанавливаемые в правительственных учреждениях США после 1990 года, должны или непосредственно поддерживать стек OSI, или обеспечивать средства для перехода на этот стек в будущем. Тем не менее, стек OSI более популярен в Европе, чем в США, так как в Европе осталось меньше старых сетей, работающих по собственным протоколам. Большинство организаций пока только планируют переход к стеку OSI, и очень немногие приступили к созданию пилотных проектов. Из тех, что работают в этом направлении, можно назвать Военно-морское ведомство США и сеть NFSNET. Одним из крупнейших

производителей, поддерживающих OSI, является компания AT&T, ее сеть Stargroup полностью базируется на этом стеке.

Стек TCP/IP был разработан по инициативе Министерства обороны США более 20 лет назад для связи экспериментальной сети ARPAnet с другими сетями как набор общих протоколов для разнородной вычислительной среды. Большой вклад в развитие стека TCP/IP, который получил свое название от популярных протоколов IP и TCP, внесли специалисты из университета Беркли, реализовавшие протоколы стека в версии ОС UNIX. Популярность этой операционной системы привела к широкому распространению протоколов TCP, IP и других протоколов стека. Сегодня этот стек используется для связи компьютеров всемирной информационной сети Internet, а также в огромном количестве корпоративных сетей.

Стек TCP/IP на нижнем уровне поддерживает все популярные стандарты физического и канального уровней: для локальных сетей – это Ethernet, Token Ring, FDDI, для глобальных – протоколы работы на аналоговых коммутируемых и выделенных линиях SLIP, PPP, протоколы территориальных сетей X.25 и ISDN.

Основными протоколами стека, давшими ему название, являются протоколы IP и TCP. Эти протоколы в терминологии модели OSI относятся к сетевому и транспортному уровням, соответственно. IP обеспечивает продвижение пакета по составной сети, а TCP гарантирует надежность его доставки.

За долгие годы использования в сетях различных стран и организаций стек TCP/IP вобрал в себя большое количество протоколов прикладного уровня. К ним относятся такие популярные протоколы, как протокол пересылки файлов FTP, протокол эмуляции терминала telnet, почтовый протокол SMTP, используемый в электронной почте сети Internet, гипертекстовые сервисы службы WWW и многие другие.

Сегодня стек TCP/IP представляет собой один из самых распространенных стеков транспортных протоколов вычислительных сетей.

Действительно, только в сети Internet объединено около 10 миллионов компьютеров по всему миру, которые взаимодействуют друг с другом с помощью стека протоколов TCP/IP.

Стремительный рост популярности Internet привел и к изменениям в расстановке сил в мире коммуникационных протоколов – протоколы TCP/IP, на которых построен Internet, стали быстро теснить бесспорного лидера прошлых лет – стек IPX/SPX компании Novell. Сегодня в мире общее количество компьютеров, на которых установлен стек TCP/IP, превысило количество компьютеров, на которых работает стек IPX/SPX, и это говорит об изменении отношения администраторов локальных сетей к протоколам, используемым на настольных компьютерах, так как именно на них раньше почти везде работали протоколы компании Novell, необходимые для доступа к файловым серверам NetWare. Процесс продвижения стека TCP/IP на лидирующие позиции в любых типах сетей продолжается, и сейчас в комплекте поставки любой промышленной операционной системы обязательно имеется программная реализация этого стека.

Хотя протоколы TCP/IP неразрывно связаны с Internet, и каждый из многомиллионной армады компьютеров Internet работает на основе этого стека, существует большое количество локальных, корпоративных и территориальных сетей, непосредственно не являющихся частями Internet, в которых также используются протоколы TCP/IP. Чтобы отличать эти сети от Internet, их называют сетями TCP/IP или просто IP-сетями.

Поскольку стек TCP/IP изначально создавался для глобальной сети Internet, он имеет много особенностей, которые обеспечивают ему преимущество перед другими протоколами, когда речь заходит о построении сетей, включающих глобальные связи. В частности, очень полезным свойством, благодаря которому этот протокол может применяться в больших сетях, является его способность фрагментировать пакеты. Действительно, сложная составная сеть часто состоит из сетей, построенных на совершенно разных принципах. В каждой из этих сетей может быть установлена собственная величина максимальной длины единицы передаваемых данных (кадра). В таком случае при переходе из одной сети, имеющей большую максимальную длину, в другую, с меньшей максимальной длиной, может возникнуть необходимость разделения передаваемого кадра на несколько частей. Протокол IP стека TCP/IP эффективно решает эту задачу.

Другой особенностью технологии TCP/IP является гибкая система адресации, позволяющая более просто по сравнению с другими протоколами аналогичного назначения включать в интерсеть (объединенную или составную сеть) сети других технологий. Это свойство также способствует применению стека TCP/IP для построения больших гетерогенных сетей.

В стеке TCP/IP очень экономно используются возможности широковещательных рассылок. Это свойство просто необходимо при работе на медленных каналах связи, характерных для территориальных сетей.

Однако платой за преимущества здесь оказываются высокие требования к ресурсам и сложность администрирования IP-сетей. Для реализации мощных функциональных возможностей протоколов стека TCP/IP требуются большие вычислительные затраты. Гибкая система адресации и отказ от широковещательных рассылок приводят к наличию в IP-сети различных централизованных служб типа DNS, DHCP и т.п. Каждая из этих служб упрощает администрирование сети и конфигурирование оборудования, но в то же время сама требует пристального внимания со стороны администраторов.

Можно приводить и другие доводы за и против, однако факт остается фактом: сегодня TCP/IP – самый популярный стек протоколов, широко используемый как в глобальных, так и в локальных сетях.

Стек IPX/SPX. Этот стек является оригинальным стеком протоколов фирмы Novell, разработанным для сетевой операционной системы NetWare еще в начале 80-х годов. Протоколы сетевого и сеансового уровней Internetwork Packet Exchange (IPX и Sequenced Packet Exchange, SPX), которые дали название стеку, являются прямой адаптацией протоколов XNS фирмы Xerox, распространенных в гораздо меньшей степени, чем стек IPX/SPX.

Популярность стека IPX/SPX непосредственно связана с операционной системой Novell NetWare, которая долгое время сохраняла мировое лидерство по числу установленных систем, хотя в последнее время ее популярность намного снизилась, и по темпам роста она заметно отстает от Microsoft Windows NT.

Многие особенности стека IPX/SPX обусловлены ориентацией ранних версий ОС NetWare (до версии 4.0) на работу в локальных сетях небольших размеров, состоящих из персональных компьютеров со скромными ресурсами. Понятно, что для таких компьютеров компании Novell нужны были протоколы, на реализацию которых требовалось бы минимальное количество оперативной памяти (ограниченной в IBM-совместимых компьютерах под управлением MS-DOS объемом 640 Кбайт) и которые быстро работали бы на процессорах небольшой вычислительной мощности. В результате протоколы стека IPX/SPX до недавнего времени хорошо работали в локальных сетях и не очень – в больших корпоративных сетях, так как они слишком перегружали медленные глобальные связи широковещательными пакетами, которые интенсивно используются несколькими протоколами этого стека (например, для установления связи между клиентами и серверами). Это обстоятельство, а также тот факт, что стек IPX/SPX является собственностью фирмы Novell, и на его реализацию нужно получать лицензию (то есть открытые спецификации не поддерживались), долгое время ограничивали его поле деятельности только сетями NetWare. Однако с момента выпуска версии NetWare 4.0 специалисты Novell внесли и продолжают вносить в протоколы серьезные изменения, направленные на их адаптацию для работы в корпоративных сетях. Сейчас стек IPX/SPX реализован не только в NetWare, но и в нескольких других популярных сетевых ОС, например SCO UNIX, Sun Solaris, Microsoft Windows NT.

Стек NetBIOS/SMB широко применяется в продуктах компаний IBM и Microsoft. На его физическом и канальном уровнях используются все наиболее распространенные протоколы Ethernet, Token Ring, FDDI и другие. На верхних уровнях работают протоколы NetBEUI и SMB.

Протокол NetBIOS (Network Basic Input/Output System) появился в 1984 году как сетевое расширение стандартных функций базовой системы ввода/вывода (BIOS) IBM PC для сетевой программы PC Network компании IBM. В дальнейшем этот протокол был заменен так называемым протоколом расширенного пользовательского интерфейса NetBEUI–NetBIOS Extended User Interface. Для обеспечения совместимости приложений в качестве интерфейса к протоколу NetBEUI был сохранен интерфейс NetBIOS. Протокол NetBEUI разрабатывался как эффективный протокол, потребляющий немного ресурсов и предназначенный для сетей, насчитывающих не более 200 рабочих станций.

Протокол NetBEUI выполняет много полезных сетевых функций, которые можно отнести к сетевому, транспортному и сеансовому уровням модели OSI, однако он не обеспечивает возможность маршрутизации пакетов. Это ограничивает применение протокола NetBEUI локальными сетями, не разделенными на подсети, и делает невозможным его использование в составных сетях.

Некоторые ограничения NetBEUI снимаются в реализации этого протокола NBF (NetBEUI Frame), которая включена в операционную систему Microsoft Windows NT.

<table>
<tr><th>Модель OSI</th><th colspan="2">IBM/Microsoft</th><th>TCP/IP</th><th>Novell</th><th>Стек OSI</th></tr>
<tr><td>Прикладной</td><td rowspan="2">SMB</td><td rowspan="2">Telnet
FTP
SNMP
WWW</td><td rowspan="2">NCP
SAP</td><td>X.400 X.500 FTAM</td></tr>
<tr><td>Представительный</td><td>Представительный протокол OSI</td></tr>
<tr><td>Сеансовый</td><td rowspan="2">NetBios</td><td rowspan="2">TCP</td><td></td><td>Сеансовый протокол OSI</td></tr>
<tr><td>Транспортный</td><td>SPX</td><td>Транспортный протокол OSI</td></tr>
<tr><td>Сетевой</td><td colspan="2">IP
RIP
OSPF</td><td>IPX
RIP
NLSP</td><td>ES-ES
IS-IS</td></tr>
<tr><td>Канальный</td><td colspan="4" rowspan="2">802.3 (Ethernet), 802.5 (Tocken Ring), FDDI, Fast Ethernet, SLIP, 100VG-AnyLAN, X.25, ATM, LAP-B, PPP
Коаксиал, экранированная и неэкранированная витая пара, оптоволокно, радиоволны</td></tr>
<tr><td>Физический</td></tr>
</table>

Рис. 7.3. Соответствие популярных стеков протоколов модели OSI

Протокол SMB (Server Message Block) выполняет функции сеансового, представительного и прикладного уровней. На основе SMB реализуется файловая служба, а также службы печати и передачи сообщений между приложениями.

Стеки протоколов SNA компании IBM, DECnet корпорации Digital Equipment и AppleTalk/AFP компании Apple применяются в основном в операционных системах и сетевом оборудовании этих фирм.

На рис. 7.3 показано соответствие некоторых наиболее популярных протоколов уровням модели OSI. Часто это соответствие весьма условно, так как модель OSI – это только руководство к действию, причем достаточно общее, а конкретные протоколы разрабатывались для решения специфических задач, причем многие из них появились до разработки модели OSI. В большинстве случаев разработчики стеков отдавали предпочтение скорости работы сети в ущерб модульности: ни один стек, кроме стека OSI, не разбит на семь уровней. Чаще всего в стеке явно выделяются три-четыре уровня: уровень сетевых адаптеров, в котором реализуются протоколы физического и канального уровней, сетевой уровень, транспортный уровень и уровень служб, объединяющий функции сеансового, представительного и прикладного уровней.

8. АДРЕСАЦИЯ В IP-СЕТЯХ

8.1. ТИПЫ АДРЕСОВ

Каждый компьютер в сети TCP/IP имеет адреса трех уровней.

1. Локальный адрес узла, определяемый технологией, с помощью которой построена отдельная сеть, в которую входит данный узел. Для узлов, входящих в локальные сети, – это MAC-адрес сетевого адаптера или порта маршрутизатора, например, 11-A0-17-3D-BC-01. Эти адреса назначаются производителями оборудования и являются уникальными адресами, так как управляются централизовано. Для всех существующих технологий локальных сетей MAC-адрес имеет формат 6 байтов: старшие 3 байта – идентификатор фирмы производителя, а младшие 3 байта назначаются уникальным образом самим производителем. Для узлов, входящих в глобальные сети, такие как X.25 или frame relay, локальный адрес назначается администратором глобальной сети.

2. IP-адрес, состоящий из 4 байт, например, 109.26.17.100. Этот адрес используется на сетевом уровне. Он назначается администратором во время конфигурирования компьютеров и маршрутизаторов. IP-адрес состоит из двух частей: номера сети и номера узла. Номер сети может быть выбран администратором произвольно, либо назначен по рекомендации специального подразделения Internet (Network Information Center, NIC), если сеть должна работать как составная часть Internet. Обычно про-

вайдеры услуг Internet получают диапазоны адресов у подразделений NIC, а затем распределяют их между своими абонентами.

Номер узла в протоколе IP назначается независимо от локального адреса узла. Деление IP-адреса на поле номера сети и номера узла – гибкое, и граница между этими полями может устанавливаться весьма произвольно. Узел может входить в несколько IP-сетей. В этом случае узел должен иметь несколько IP-адресов, по числу сетевых связей. Таким образом IP-адрес характеризует не отдельный компьютер или маршрутизатор, а одно сетевое соединение.

3. Символьный идентификатор-имя, например, SERV1.IBM.COM. Этот адрес назначается администратором и состоит из нескольких частей, например, имени машины, имени организации, имени домена. Такой адрес, называемый также DNS-именем, используется на прикладном уровне, например, в протоколах FTP или telnet.

8.2. ТРИ ОСНОВНЫХ КЛАССА IP-АДРЕСОВ

IP-адрес имеет длину 4 байта и обычно записывается в виде четырех чисел, представляющих значения каждого байта в десятичной форме, и разделенных точками, например:

128.10.2.30 – традиционная десятичная форма представления адреса,
10000000 00001010 00000010 00011110 – двоичная форма представления этого же адреса.

На рис. 8.1 показана структура IP-адреса.

Адрес состоит из двух логических частей – номера сети и номера узла в сети. Какая часть адреса относится к номеру сети, а какая к номеру узла, определяется значениями первых битов адреса.

- Если адрес начинается с 0, то сеть относят к классу A, и номер сети занимает один байт, остальные 3 байта интерпретируются как номер узла в сети. Сети класса A имеют номера в диапазоне от 1 до 126. (Номер 0 не используется, а номер 127 зарезервирован для специальных целей, о чем будет сказано ниже.) В сетях класса A количество узлов должно быть больше 216 , но не превышать 224.
- Если первые два бита адреса равны 10, то сеть относится к классу B и является сетью средних размеров с числом узлов 28…216. В сетях класса B под адрес сети и под адрес узла отводится по 16 битов, то есть по 2 байта.

Класс A

0	N сети	N узла

Класс B

1	0	N сети	N узла

Класс C

1	1	0	N сети	N узла

Класс D

1	1	1	0	адрес группы multicast

Класс E

1	1	1	1	0	зарезервирован

Рис. 8.1. Структура IP-адреса

- Если адрес начинается с последовательности 110, то это сеть класса C с числом узлов не больше 28. Под адрес сети отводится 24 бита, а под адрес узла – 8 битов.
- Если адрес начинается с последовательности 1110, то он является адресом класса D и обозначает особый, групповой адрес – multicast. Если в пакете в качестве адреса назначения указан адрес класса D, то такой пакет должны получить все узлы, которым присвоен данный адрес.
- Если адрес начинается с последовательности 11110, то это адрес класса E, он зарезервирован для будущих применений.

В табл. 8.1 приведены диапазоны номеров сетей, соответствующих каждому классу сетей.

8.1. Диапазоны номеров сетей

Класс	Наименьший адрес	Наибольший адрес
A	01.0.0	126.0.0.0
B	128.0.0.0	191.255.0.0
C	192.0.1.0.	223.255.255.0
D	224.0.0.0	239.255.255.255
E	240.0.0.0	247.255.255.255

В протоколе IP существует несколько соглашений об особой интерпретации IP-адресов.

- Если IP-адрес состоит только из двоичных нулей,

0 0 0 0 ..... 0 0 0 0

то он обозначает адрес того узла, который сгенерировал этот пакет.

- Если в поле номера сети стоят 0,

0 0 0 0 0	Номер узла

то по умолчанию считается, что этот узел принадлежит той же самой сети, что и узел, который отправил пакет.

- Если все двоичные разряды IP-адреса равны 1,

1 1 1 1 .....1 1

то пакет с таким адресом назначения должен рассылаться всем узлам, находящимся в той же сети, что и источник этого пакета. Такая рассылка называется ограниченным широковещательным сообщением (limited broadcast).

- Если в поле адреса назначения стоят сплошные 1,

Номер сети	1111.....11

то пакет, имеющий такой адрес, рассылается всем узлам сети с заданным номером. Такая рассылка называется широковещательным сообщением (broadcast).

- Адрес 127.0.0.1 зарезервирован для организации обратной связи при тестировании работы программного обеспечения узла без реальной отправки пакета по сети. Этот адрес имеет название loopback.

Уже упоминавшаяся форма группового IP-адреса – multicast – означает, что данный пакет должен быть доставлен сразу нескольким узлам, которые образуют группу с номером, указанным в поле адреса. Узлы сами идентифицируют себя, то есть определяют, к какой из групп они относятся. Один и тот же узел может входить в несколько групп. Такие сообщения в отличие от широковещательных называются мультивещательными. Групповой адрес не делится на поля номера сети и узла и обрабатывается маршрутизатором особым образом.

В протоколе IP нет понятия широковещательности в том смысле, в котором оно используется в протоколах канального уровня локальных сетей, когда данные должны быть доставлены абсолютно всем узлам. Как ограниченный широковещательный IP-адрес, так и широковещательный IP-адрес имеют пределы распространения в интерсети – они ограничены либо сетью, к которой принадлежит узел – источник пакета, либо сетью, номер которой указан в адресе назначения. Поэтому деление сети с помощью маршрутизаторов на части локализует широковещательный шторм пределами одной из составляющих общую сеть частей просто потому, что нет способа адресовать пакет одновременно всем узлам всех сетей составной сети.

8.3. Отображение физических адресов на IP-адреса: протоколы ARP и RARP

В протоколе IP-адрес узла, то есть адрес компьютера или порта маршрутизатора, назначается произвольно администратором сети и прямо не связан с его локальным адресом, как это сделано, например, в протоколе IPX. Подход, используемый в IP, удобно использовать в крупных сетях и по причине его независимости от формата локального адреса, и по причине стабильности, так как в противном случае, при смене на компьютере сетевого адаптера это изменение должны бы были учитывать все адресаты всемирной сети Internet (в том случае, конечно, если сеть подключена к Internet).

Локальный адрес используется в протоколе IP только в пределах локальной сети при обмене данными между маршрутизатором и узлом этой сети. Маршрутизатор, получив пакет для узла одной из сетей, непосредственно подключенных к его портам, должен для передачи пакета сформировать кадр в соответствии с требованиями принятой в этой сети технологии и указать в нем локальный адрес узла, например его MAC-адрес. В пришедшем пакете этот адрес не указан, поэтому перед маршрутизатором встает задача поиска его по известному IP-адресу, который указан в пакете в качестве адреса назначения. С аналогичной задачей сталкивается и конечный узел, когда он хочет отправить пакет в удаленную сеть через маршрутизатор, подключенный к той же локальной сети, что и данный узел.

Для определения локального адреса по IP-адресу используется протокол разрешения адреса *Address Resolution Protocol*, *ARP*. Протокол ARP работает различным образом в зависимости от того, какой протокол канального уровня работает в данной сети – протокол локальной сети (Ethernet, Token Ring, FDDI) с возможностью широковещательного доступа одновременно ко всем узлам сети, или же протокол глобальной сети (X.25, frame relay), как правило не поддерживающий широковещательный доступ. Существует также протокол, решающий обратную задачу – нахождение IP-адреса по известному локальному адресу. Он называется реверсивный ARP – *RARP* (*Reverse Address Resolution Protocol*) и используется при старте бездисковых станций, не знающих в начальный момент своего IP-адреса, но знающих адрес своего сетевого адаптера.

В локальных сетях протокол ARP использует широковещательные кадры протокола канального уровня для поиска в сети узла с заданным IP-адресом.

Узел, которому нужно выполнить отображение IP-адреса на локальный адрес, формирует ARP запрос, вкладывает его в кадр протокола канального уровня, указывая в нем известный IP-адрес, и рассылает запрос широковещательно. Все узлы локальной сети получают ARP запрос и сравнивают указанный там IP-адрес с собственным. В случае их совпадения узел формирует ARP-ответ, в котором указывает свой IP-адрес и свой локальный адрес и отправляет его уже направленно, так как в ARP запросе отправитель указывает свой локальный адрес. ARP-запросы и ответы используют один и тот же формат пакета. Так как локальные адреса могут в различных типах сетей иметь различную длину, то формат пакета протокола ARP зависит от типа сети. На рис. 8.2 показан формат пакета протокола ARP для передачи по сети Ethernet.

ТИП СЕТИ		**Тип протокола**
Длина локального адреса	Длина сетевого адреса	Операция
Локальный адрес отправителя (байты 0 – 3)		

Локальный адрес отправителя (байты 4 – 5)	IP-адрес отправителя (байты 0 – 1)
IP-адрес отправителя (байты 2 – 3)	Искомый локальный адрес (байты 0 – 1)
Искомый локальный адрес (байты 2 – 5)	
Искомый IP-адрес (байты 0 – 3)	

Рис. 8.2. Формат пакета протокола ARP

В поле типа сети для сетей Ethernet указывается значение 1. Поле типа протокола позволяет использовать пакеты ARP не только для протокола IP, но и для других сетевых протоколов. Для IP значение этого поля равно 0800_{16}.

Длина локального адреса для протокола Ethernet равна 6 байтам, а длина IP-адреса – 4 байтам. В поле операции для ARP запросов указывается значение 1 – для протокола ARP и 2 – для протокола RARP.

Узел, отправляющий ARP-запрос, заполняет в пакете все поля, кроме поля искомого локального адреса (для RARP-запроса не указывается искомый IP-адрес). Значение этого поля заполняется узлом, опознавшим свой IP-адрес.

В глобальных сетях администратору сети чаще всего приходится вручную формировать ARP-таблицы, в которых он задает, например, соответствие IP-адреса адресу узла сети X.25, который имеет смысл локального адреса. В последнее время наметилась тенденция автоматизации работы протокола ARP и в глобальных сетях. Для этой цели среди всех маршрутизаторов, подключенных к какой-либо глобальной сети, выделяется специальный маршрутизатор, который ведет ARP-таблицу для всех остальных узлов и маршрутизаторов этой сети. При таком централизованном подходе для всех узлов и маршрутизаторов вручную нужно задать только IP-адрес и локальный адрес выделенного маршрутизатора. Затем каждый узел и маршрутизатор регистрирует свои адреса в выделенном маршрутизаторе, а при необходимости установления соответствия между IP-адресом и локальным адресом узел обращается к выделенному маршрутизатору с запросом и автоматически получает ответ без участия администратора.

8.4. Отображение символьных адресов на IP-адреса: служба DNS

DNS (Domain Name System) – это распределенная база данных, поддерживающая иерархическую систему имен для идентификации узлов в сети Internet. Служба DNS предназначена для автоматического поиска IP-адреса по известному символьному имени узла. Спецификация DNS определяется стандартами RFC 1034 и 1035. DNS требует статической конфигурации своих таблиц, отображающих имена компьютеров в IP-адрес.

Протокол DNS является служебным протоколом прикладного уровня. Этот протокол несимметричен: в нем определены DNS-серверы и DNS-клиенты. DNS-серверы хранят часть распределенной базы данных о соответствии символьных имен и IP-адресов. Эта база данных распределена по административным доменам сети Internet. Клиенты сервера DNS знают IP-адрес сервера DNS своего административного домена и по протоколу IP передают запрос, в котором сообщают известное символьное имя и просят вернуть соответствующий ему IP-адрес.

Если данные о запрошенном соответствии хранятся в базе данного DNS-сервера, то он сразу посылает ответ клиенту, если же нет – то он посылает запрос DNS-серверу другого домена, который может сам обработать запрос, либо передать его другому DNS-серверу. Все DNS-серверы соединены иерархически, в соответствии с иерархией доменов сети Internet. Клиент опрашивает эти серверы имен, пока не найдет нужные отображения. Этот процесс ускоряется из-за того, что серверы имен постоянно кэшируют информацию, предоставляемую по запросам. Клиентские компьютеры могут использовать в своей работе IP-адреса нескольких DNS-серверов, для повышения надежности своей работы.

База данных DNS имеет структуру дерева, называемого доменным пространством имен, в котором каждый домен (узел дерева) имеет имя и может содержать поддомены. Имя домена идентифицирует его положение в этой базе данных по отношению к родительскому домену, причем точки в имени отделяют части, соответствующие узлам домена.

Корень базы данных DNS управляется центром Internet Network Information Center. Домены верхнего уровня назначаются для каждой страны, а также на организационной основе. Имена этих доменов должны следовать международному стандарту ISO 3166. Для обозначения стран используются трехбуквенные и двухбуквенные аббревиатуры, а для различных типов организаций используются следующие аббревиатуры:

com – коммерческие организации (например, microsoft.com);
edu – образовательные (например, tstu.edu);
gov – правительственные организации (например, nsf.gov);
org – некоммерческие организации (например, fidonet.org);
net – организации, поддерживающие сети (например, nsf.net).

Каждый домен DNS администрируется отдельной организацией, которая обычно разбивает свой домен на поддомены и передает функции администрирования этих поддоменов другим организациям. Каждый домен имеет уникальное имя, а каждый из поддоменов имеет уникальное имя внутри своего домена. Имя домена может содержать до 63 символов. Каждый хост в сети Internet однозначно определяется своим *полным доменным именем* (fully qualified domain name, FQDN), которое включает имена всех доменов по направлению от хоста к корню. Пример полного DNS-имени: http://www.tstu.fdo.ru.

8.5. Автоматизация процесса назначения IP-адресов узлам сети – протокол DHCP

Как уже было сказано, IP-адреса могут назначаться администратором сети вручную. Это представляет для администратора утомительную процедуру. Ситуация усложняется еще тем, что многие пользователи не обладают достаточными знаниями для того, чтобы конфигурировать свои компьютеры для работы в интерсети и должны поэтому полагаться на администраторов.

Протокол *Dynamic Host Configuration Protocol* (DHCP) был разработан для того, чтобы освободить администратора от этих проблем. Основным назначением DHCP является динамическое назначение IP-адресов. Однако, кроме динамического, DHCP может поддерживать и более простые способы ручного и автоматического статического назначения адресов.

В ручной процедуре назначения адресов активное участие принимает администратор, который предоставляет DHCP-серверу информацию о соответствии IP-адресов физическим адресам или другим идентификаторам клиентов. Эти адреса сообщаются клиентам в ответ на их запросы к DHCP-серверу.

При автоматическом статическом способе DHCP-сервер присваивает IP-адрес (и, возможно, другие параметры конфигурации клиента) из пула наличных IP-адресов без вмешательства оператора. Границы пула назначаемых адресов задает администратор при конфигурировании DHCP-сервера. Между идентификатором клиента и его IP-адресом по-прежнему, как и при ручном назначении, существует постоянное соответствие. Оно устанавливается в момент первичного назначения сервером DHCP IP-адреса клиенту. При всех последующих запросах сервер возвращает тот же самый IP-адрес.

При динамическом распределении адресов DHCP-сервер выдает адрес клиенту на ограниченное время, что дает возможность впоследствии повторно использовать IP-адреса другими компьютерами. Динамическое разделение адресов позволяет строить IP-сеть, количество узлов в которой намного превышает количество имеющихся в распоряжении администратора IP-адресов.

DHCP обеспечивает надежный и простой способ конфигурации сети TCP/IP, гарантируя отсутствие конфликтов адресов за счет централизованного управления их распределением. Администратор управляет процессом назначения адресов с помощью параметра "продолжительности аренды" (lease duration), которая определяет, как долго компьютер может использовать назначенный IP-адрес, перед тем как снова запросить его от сервера DHCP в аренду.

Примером работы протокола DHCP может служить ситуация, когда компьютер, являющийся клиентом DHCP, удаляется из подсети. При этом назначенный ему IP-адрес автоматически освобождается. Когда компьютер подключается к другой подсети, то ему автоматически назначается новый адрес. Ни

пользователь, ни сетевой администратор не вмешиваются в этот процесс. Это свойство очень важно для мобильных пользователей.

Протокол DHCP использует модель клиент-сервер. Во время старта системы компьютер-клиент DHCP, находящийся в состоянии "инициализация", посылает сообщение discover (исследовать), которое широковещательно распространяется по локальной сети и передается всем DHCP-серверам частной интерсети. Каждый DHCP-сервер, получивший это сообщение, отвечает на него сообщением offer (предложение), которое содержит IP-адрес и конфигурационную информацию.

Компьютер-клиент DHCP переходит в состояние "выбор" и собирает конфигурационные предложения от DHCP-серверов. Затем он выбирает одно из этих предложений, переходит в состояние "запрос" и отправляет сообщение request (запрос) тому DHCP-серверу, чье предложение было выбрано.

Выбранный DHCP-сервер посылает сообщение DHCP-acknow-ledgment (подтверждение), содержащее тот же IP-адрес, который уже был послан ранее на стадии исследования, а также параметр аренды для этого адреса. Кроме того, DHCP-сервер посылает параметры сетевой конфигурации. После того, как клиент получит это подтверждение, он переходит в состояние "связь", находясь в котором он может принимать участие в работе сети TCP/IP. Компьютеры-клиенты, которые имеют локальные диски, сохраняют полученный адрес для использования при последующих стартах системы. При приближении момента истечения срока аренды адреса компьютер пытается обновить параметры аренды у DHCP-сервера, а если этот IP-адрес не может быть выделен снова, то ему возвращается другой IP-адрес.

В протоколе DHCP описывается несколько типов сообщений, которые используются для обнаружения и выбора DHCP-серверов, для запросов информации о конфигурации, для продления и досрочного прекращения лицензии на IP-адрес. Все эти операции направлены на то, чтобы освободить администратора сети от утомительных рутинных операций по конфигурированию сети.

Однако использование DHCP несет в себе и некоторые проблемы. Во-первых, это проблема согласования информационной адресной базы в службах DHCP и DNS. Как известно, DNS служит для преобразования символьных имен в IP-адреса. Если IP-адреса будут динамически изменяться сервером DHCP, то эти изменения необходимо также динамически вносить в базу данных сервера DNS. Хотя протокол динамического взаимодействия между службами DNS и DHCP уже реализован некоторыми фирмами (так называемая служба Dynamic DNS), стандарт на него пока не принят.

Во-вторых, нестабильность IP-адресов усложняет процесс управления сетью. Системы управления, основанные на протоколе SNMP, разработаны с расчетом на статичность IP-адресов. Аналогичные проблемы возникают и при конфигурировании фильтров маршрутизаторов, которые оперируют с IP-адресами.

Наконец, централизация процедуры назначения адресов снижает надежность системы: при отказе DHCP-сервера все его клиенты оказываются не в состоянии получить IP-адрес и другую информацию о конфигурации. Последствия такого отказа могут быть уменьшены путем использовании в сети нескольких серверов DHCP, каждый из которых имеет свой пул IP-адресов.

9. СЕТЕВЫЕ ПРОТОКОЛЫ

9.1. ЭТАЛОННАЯ МОДЕЛЬ TCP/IP

В отличие от эталонной модели OSI, модель TCP/IP в большей степени ориентируется на обеспечение сетевых взаимодействий, нежели на жесткое разделение функциональных уровней. Для этой цели она признает важность иерархической структуры функций, но предоставляет проектировщикам протоколов достаточную гибкость в реализации. Соответственно, эталонная модель OSI гораздо лучше подходит для объяснения механики межкомпьютерных взаимодействий, но протокол TCP/IP стал основным межсетевым протоколом.

Гибкость эталонной модель TCP/IP по сравнению с эталонной моделью OSI продемонстрирована в табл. 9.1.

9.1. Соответствие уровней модели OSI и уровней протокола TCP/IP

<table>
<tr><th>Уровень OSI</th><th>Номер OSI</th><th>Эквивалентный уровень TCP/IP</th></tr>
<tr><td>Прикладной уровень</td><td>7</td><td rowspan="3">Прикладной уровень</td></tr>
<tr><td>Представительский уровень</td><td>6</td></tr>
<tr><td rowspan="2">Сеансовый уровень</td><td rowspan="2">5</td></tr>
<tr><td rowspan="2">Межхостовой уровень</td></tr>
<tr><td>Транспортный уровень</td><td>4</td></tr>
<tr><td>Сетевой уровень</td><td>3</td><td>Межсетевой уровень</td></tr>
<tr><td>Канальный уровень</td><td>2</td><td rowspan="2">Уровень сетевого доступа</td></tr>
<tr><td>Физический уровень</td><td>1</td></tr>
</table>

9.2. Анатомия модели TCP/IP

Стек протоколов TCP/IP состоит из четырех функциональных уровней: прикладного, межхостового, межсетевого и уровня сетевого доступа.

Прикладной уровень содержит протоколы удаленного доступа и совместного использования ресурсов. Хорошо знакомые приложения – такие, как Telnet, FTP, SMTP, HTTP и многие другие – работают на этом уровне и зависят от функциональности уровней, расположенных ниже в иерархии. Любые приложения, использующие взаимодействие в сетях IP (включая любительские и коммерческие программы), относятся к этому уровню модели.

К функциям межхостового уровня относится сегментирование данных в приложениях для пересылки по сети, выполнение математических проверок целостности принятых данных и мультиплексирование потоков данных (как передаваемых, так и принимаемых) для нескольких приложений одновременно. Отсюда следует, что межхостовой уровень располагает средствами идентификации приложений и умеет переупорядочивать данные, принятые не в том порядке.

В настоящее время межхостовой уровень состоит из двух протоколов: протокола управления передачей TCP и протокола пользовательских дейтаграмм UDP. С учетом того, что Интернет становится все более транзакционно-ориентированным, был определен третий протокол, условно названный протоколом управления транзакциями/передачей T/TCP (Transaction/Transmis-sion Control Protocol). Тем не менее, в большинстве прикладных сервисов Интернета на межхостовом уровне используются протоколы TCP и UDP.

Межсетевой уровень IPv4 состоит из всех протоколов и процедур, позволяющих потоку данных между хостами проходить по нескольким сетям. Следовательно, пакеты, в которых передаются данные, должны быть маршрутизируемыми. За маршрутизируемость пакетов отвечает протокол IP (Internet Protocol).

Межсетевой уровень должен поддерживать маршрутизацию и функции управления маршрутами. Эти функции предоставляются внешними протоколами, которые называются протоколами маршрутизации. К их числу относятся протоколы IGP (Interior Gateway Protocols) и EGP (Exterior Gateway Protocols).

Уровень сетевого доступа состоит из всех функций, необходимых для физического подключения и передачи данных по сети. В эталонной модели OSI (Open Systems Interconnection) этот набор функций разбит на два уровня: физический и канальный. Эталонная модель TCP/IP создавалась после протоколов, присутствующих в ее названии, и в ней эти два уровня были слиты воедино, поскольку различные протоколы IP останавливаются на межсетевом уровне. Протокол IP предполагает, что все низкоуровневые функции предоставляются либо локальной сетью, либо подключением через последовательный интерфейс.

Протокол TCP/IP обеспечивает возможность межплатформенных сетевых взаимодействий (т.е. связи в разнородных сетях). Например, сеть под управлением Windows NT/2000 может содержать рабочие станции Unix и Macintosh, и даже другие сети более низкого порядка. TCP/IP обладает следующими характеристиками:

- хорошие средства восстановления после сбоев;
- возможность добавления новых сетей без прерывания текущей работы;
- устойчивость к ошибкам;
- независимость от платформы реализации;
- низкие непроизводительные затраты на пересылку служебных данных.

9.2.1. Уровни и протоколы TCP/IP

Протоколы TCP и IP совместно управляют потоками данных (как входящими, так и исходящими) в сети. Но если протокол IP просто передает пакеты, не обращая внимания на результат, TCP должен проследить за тем, чтобы пакеты прибыли в положенное место. В частности, TCP отвечает за выполнение следующих задач:

- открытие и закрытие сеанса;
- управление пакетами;
- управление потоком данных;
- обнаружение и обработка ошибок.

9.3. Модель TCP/IP

Протокол TCP/IP обычно рассматривается в контексте эталонной модели, определяющей структурное деление его функций. Однако модель TCP/IP разрабатывалась значительно позже самого комплекса протоколов, поэтому она ни как не могла быть взята за образец при проектировании протоколов.

9.3.1. СЕМЕЙСТВО ПРОТОКОЛОВ TCP/IP

Семейство протоколов IP состоит из нескольких протоколов, часто обозначаемых общим термином "TCP/IP":

- IP – протокол межсетевого уровня;
- TCP – протокол межхостового уровня, обеспечивающий надежную доставку;
- UDP – протокол межхостового уровня, не обеспечивающий надежной доставки;
- ICMP – многоуровневый протокол, упрощающий контроль, тестирование и управление в сетях IP.

Различные протоколы ICMP распространяются на межхостовой и прикладной уровни.

Связи между этими протоколами изображены на рис. 9.1

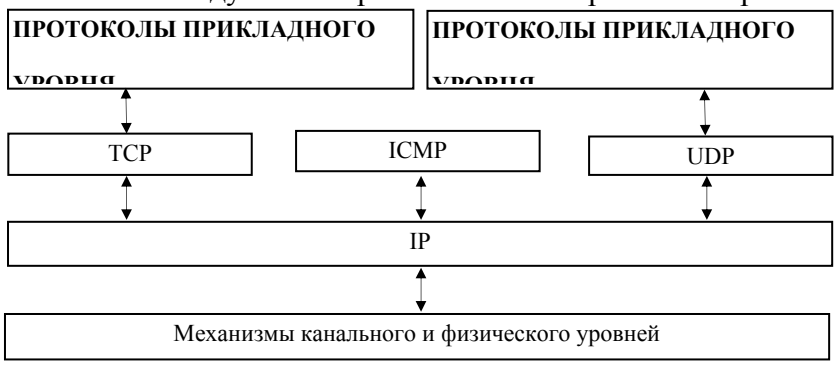

Рис. 9.1. Связи между протоколами

9.3.2. ПРОТОКОЛ TCP

Протокол TCP (Transmission Control Protocol) пользуется сервисом IP для обеспечения надежной доставки приклад-

ных данных. TCP создает между двумя или более хостами сеанс, ориентированный на соединение. Он обладает такими возможностями, как поддержка нескольких потоков данных, координация потока и контроль ошибок и даже восстановление нарушенного порядка пакетов. Протокол TCP также разрабатывался посредством публикации общедоступных документов RFC группой IETF.

В сеансе связи TCP обеспечивает ряд важных функций, большая часть которых связана с обеспечением интерфейса между различными приложениями и сетью. К числу этих функций относятся:

- мультиплексирование данных между приложениями и сетью;
- проверка целостности полученных данных;
- восстановление нарушенного порядка данных;
- подтверждение успешного получения данных;
- регулирование скорости передачи данных;
- измерение временных характеристик;
- координация повторной передачи данных, поврежденных или потерянных в процессе пересылки.

9.3.3. ПРОТОКОЛ TELNET

Термин "Telnet" (TELecommunications NETwork) обычно используется для обозначения как приложения, так и самого протокола, что наделяет его двойным смыслом. Telnet предоставляет в распоряжение пользователя средства для удаленного входа и прямого выполнения терминальных операций по сети. Иначе говоря, Telnet обеспечивает прямой доступ к удаленному компьютеру. Telnet работает на порте 23.

На хосте должен работать сервер Telnet, ожидающий аутентифицированного удаленного входа. В Windows 9x/NT/2000, BeOS, Linux и других операционных системах на платформе x86 необходимо установить отдельный сервер Telnet, настроить его и запустить на прием входящих запросов. Системы на базе MacOS также требуют отдельного сервера Telnet. Только в системах Unix имеется собственный сервер Telnet, который обычно называется telnetd ("d"-"daemon" – серверное приложение, работающее в фоновом режиме). На другом конце соединения работает приложение Telnet, обеспечивающее текстовый или графический интерфейс для пользовательского сеанса.

9.3.4. ПРОТОКОЛ FTP

В отличие от протокола Telnet, позволяющего работать на удаленном хосте, протокол FTP (File Transfer Protocol) играет более пассивную роль и предназначается для приема и отправки файлов на удаленный сервер. Такая возможность идеально подходит для web-мастеров и вообще для всех, кому потребуется переслать большие файлы с одного компьютера на другой без прямого подключения. FTP обычно используется в так называемом "пассивном" режиме, при котором клиент загружает данные о дереве каталогов и отключается, но периодически сигнализирует серверу о необходимости сохранять открытый порт.

В системах Unix поддержка FTP обычно обеспечивается программами ftpd и ftp. По умолчанию протокол FTP работает на портах 20 (пересылка данных) и 21 (пересылка команд). FTP отличается от всех остальных протоколов TCP/IP тем, что команды могут передаваться одновременно с передачей данных в реальном времени; у других протоколов подобная возможность отсутствует.

Клиенты и сервера FTP в той или иной форме существуют во всех операционных системах. Приложения FTP на базе MacOS имеют графический интерфейс, как и большинство приложений для системы Windows. Преимущество графических клиентов FTP заключается в том, что команды, обычно вводимые вручную, теперь автоматически генерируются клиентом, что снижает вероятность ошибок, упрощает и ускоряет работу. С другой стороны, серверы FTP после первоначальной настройки не требуют дополнительного внимания, поэтому графический интерфейс для них оказывается лишним.

9.3.5. ПРОТОКОЛ TFTP

Название протокола TFTP (Trivial FTP) выбрано весьма удачно. TFTP поддерживает лишь малое подмножество функций FTP. Он работает на базе протокола UDP. TFTP не следит за доставкой пакетов и практически не обладает средствами обработки ошибок. С другой стороны, эти ограничения снижают непроизводительные затраты при пересылке. TFTP не выполняет аутентификации; он просто устанав-

ливает соединение. В качестве защитной меры TFTP позволяет перемещать только общедоступные файлы.

Применение TFTP создает серьезную угрозу для безопасности системы. По этой причине TFTP обычно используется во встроенных приложениях, для копирования конфигурационных файлов при настройке маршрутизатора, при необходимости жесткой экономии ресурсов, а также в тех случаях, когда безопасность обеспечивается другими средствами. Протокол TFTP также используется в сетевых конфигурациях, в которых загрузка компьютеров производится с удаленного сервера, а протокол TFTP может быть легко записан в ПЗУ сетевых адаптеров.

9.3.6. ПРОТОКОЛ SMTP

Протокол SMTP (Simple Mail Transfer Protocol) является фактическим стандартом пересылки электронной почты в сетях, особенно в Интернете. Во всех операционных системах имеются почтовые клиенты с поддержкой SMTP, а большинство поставщиков услуг Интернета использует SMTP для работы с исходящей почтой. Серверы SMTP существуют для всех операционных систем, включая Windows 9x/NT/2K, MacOS, семейство Unix, Linux, BeOS, и даже AmigaOS.

Протокол SMTP проектировался для транспортировки сообщений электронной почты в разных сетевых средах. В сущности, SMTP не следит за тем, как перемещается сообщение, а лишь за тем, чтобы оно было доставлено к месту назначения.

SMTP обладает мощными средствами обработки почты, обеспечивающими автоматическую маршрутизацию по определенным критериям. В частности, SMTP может оповестить отправителя о том, что адрес не существует, и вернуть ему сообщение, если почта остается не доставленной в течение определенного периода времени (задаваемого системным администратором сервера, с которого отправляется сообщение). SMTP использует порт TCP с номером 25.

9.3.7. ПРОТОКОЛ SNMP

Протокол SNMP (Simple Network Management Protocol) реализует простые средства сбора данных о работе маршрутизатора и управления им с использованием различных протоколов – таких, как UDP, IPX и IP. При любом обсуждении SNMP важно помнить, что первая буква в названии протокола означает "простой". Протокол поддерживает только четыре команды – GET, GETNEXT, SET и TRAP. Первые две команды предоставляют доступ к информации, а третья позволяет осуществлять удаленное управление некоторыми функциями маршрутизаторов. Команда TRAP включает режим получения от устройства информации о проблемах или происходящих событиях.

Сетевые устройства передают информацию о себе через базу управляющей информации MIB (Management Information Base). Эти данные, описывающие устройство, передаются станции управления SNMP (SNMP Management Station), которая поочередно идентифицирует каждое устройство и сохраняет информацию о нем. Станция управляет всеми SNMP-совместимыми устройствами. Для каждого устройства запускается агент SNMP, представляющий клиентскую сторону операций с устройствами. Когда станция управления запрашивает информацию о порте командой GET, агент возвращает эту информацию.

Протокол SNMP не предназначен для управления всеми сетевыми устройствами с возможностью точного описания операций. Это простой протокол для повседневной работы, который позволяет получить нужную информацию без загрузки 5-6 управляющих интерфейсов. Для отправки сообщений SNMP используется транспортный протокол UDP.

9.3.8. ПРОТОКОЛ HTTP

Протокол HTTP заложен в основу работы World Wide Web. В сущности, именно HTTP принадлежит основная заслуга в бурном развитии Интернета в середине 1990-х годов. Сначала появились первые клиенты HTTP (такие, как Mosaic и Netscape), которые позволяли наглядно "увидеть" Web. Вскоре стали появляться web-серверы с полезной информацией. В наше время в Интернете существует более шести миллионов web-сайтов, работающих на базе HTTP. Протокол HTTP работает на хорошо известном порте TCP с номером 80. Протокол передачи гипертекста (HTTP) – протокол прикладного уровня для распределенных, совместных, многосредных информационных систем. HTTP используется в World Wide Web (WWW) начиная с 1990 года. Первой версией HTTP, известной как HTTP/0.9, был простой

протокол для передачи необработанных данных через Интернет. HTTP/1.0 был улучшением этого протокола, допускал MIME-подобный формат сообщений, содержащий метаинформацию о передаваемых данных и имел модифицированную семантику запросов/ответов. Однако HTTP/1.0 недостаточно учитывал особенности работы с иерархическими прокси-серверами (hierarchical proxies), кэшированием, постоянными соединениями, и виртуальными хостами (virtual hosts). Кроме того, быстрый рост числа не полностью совместимых с протоколом HTTP/1.0 приложений, потребовал введения новой версии протокола, в которой были бы заложены дополнительные возможности, которые помогли бы привести эти приложения к единому стандарту.

Большие информационные системы требуют большего количества функциональных возможностей, чем просто загрузку информации, включая поиск и модификацию данных при помощи внешних интерфейсов. HTTP предоставляет открытый (open-ended) набор методов, которые основаны на системе ссылок, которые обеспечиваются URI (Универсальными Идентификаторами Ресурсов). URI могут идентифицировать как расположение (URL), так и имя (URN) ресурса, к которому применяется данный метод. Сообщения передаются в формате, подобному используемому электронной почтой согласно определениям MIME (Многоцелевых Расширений Электронной Почты).

HTTP также используется как обобщенный протокол связи между агентами пользователей (user agents) и прокси-серверами/шлюзами (proxies/gateways) или другими Интернет-сервисами, включая такие как SMTP, NNTP, FTP, Gopher и WAIS. Таким образом, HTTP определяет основы многосредного доступа к ресурсам для разнообразных приложений.

Протокол HTTP – это протокол запросов/ответов. Клиент посылает по соединению запрос серверу, содержащий: метод запроса, URI, версию протокола, MIME-подобное сообщение, включающее модификаторы запроса, клиентскую информацию и, возможно, тело запроса. Сервер отвечает строкой состояния, включающей версию протокола сообщения, кодом успешного выполнения или ошибки, MIME-подобным сообщением, содержащим информацию о сервере, метаинформацию объекта и, возможно, тело объекта.

Большинство HTTP соединений, инициализируется агентом пользователя и состоит из запроса, который нужно применить к ресурсу на некотором первоначальном сервере. В самом простом случае, он может быть выполнен посредством одиночного соединения между агентом пользователя и первоначальным сервером.

Более сложная ситуация возникает, когда в цепочке запросов/ответов присутствует один или несколько посредников. Существуют три основных разновидности посредников: прокси-сервера, шлюзы и туннели. Прокси-сервер является агентом-посредником, который получает запросы на некоторый URI в абсолютной форме, изменяет все сообщение или его часть и отсылает измененный запрос серверу, идентифицированному URI. Шлюз – это принимающий агент, действующий как бы на уровень выше некоторого другого сервера и при необходимости транслирующий запросы в протокол основного сервера. Туннель действует как реле между двумя соединениями, не изменяя сообщений; туннели используются, когда связь нужно производить через посредника (например, firewall), который не понимает содержание сообщений.

Фактически, имеется широкое разнообразие архитектур и конфигураций кэшей и прокси-серверов, разрабатываемых в настоящее время или развернутых в World Wide Web; эти системы включают национальные иерархии прокси-кэшей, которые сохраняют пропускную способность межокеанских каналов, системы, которые распространяют по многим адресам содержимое кэша, организации, которые распространяют подмножества кэшируемых данных на CD-ROM, и так далее. HTTP системы используются в корпоративных интранет-сетях с высокоскоростными линиями связи, и для доступа через PDA с маломощными радиолиниями и неустойчивой связью. Цель HTTP/1.1 состоит в поддержании широкого многообразия конфигураций, уже построенных при введении ранних версий протокола, а также в удовлетворении потребностей разработчиков web приложений, требующих все более высокой надежности.

HTTP соединение обычно происходит посредством TCP/IP соединений. HTTP также может быть реализован посредством любого другого протокола Интернет, или других сетей. HTTP необходима только надежная передача данных, следовательно, может использоваться любой протокол, который гарантирует надежную передачу данных; отображение структуры запроса и ответа HTTP/1.1 на транспортные модули данных рассматриваемого протокола – вопрос, не решается на уровне самого протокола.

9.4. ОПИСАНИЕ ПРОТОКОЛА HTTP/1.1

Протокол передачи гипертекста (HTTP) – протокол прикладного уровня для распределенных, совместных, многосредных информационных систем. HTTP используется в World Wide Web (WWW), начиная с 1990 года. Первой версией HTTP, известной как HTTP/0.9, был простой протокол для передачи необработанных данных через Интернет. По определению RFC 1945 HTTP/1.0 был улучшением этого протокола, допускал MIME-подобный формат сообщений, содержащий метаинформацию о передаваемых данных и имел модифицированную семантику запросов/ответов. Однако HTTP/1.0 недостаточно учитывал особенности работы с иерархическими прокси-серверами (hierarchical proxies), кэшированием, постоянными соединениями, и виртуальными хостами (virtual hosts). Кроме того, быстрый рост числа не полностью совместимых с протоколом HTTP/1.0 приложений, потребовал введения новой версии протокола, в которой были бы заложены дополнительные возможности, которые помогли бы привести эти приложения к единому стандарту.

9.4.1. Назначение

Протокол HTTP/1.1 содержит более строгие требования, чем HTTP/1.0, гарантирующие более надежную работу.

Большие информационные системы требуют большего количества функциональных возможностей, чем просто загрузку информации, включая поиск и модификацию данных при помощи внешних интерфейсов. HTTP предоставляет открытый (open-ended) набор методов, которые основаны на системе ссылок, которые обеспечиваются URI (Универсальными Идентификаторами Ресурсов). URI могут идентифицировать как расположение (URL), так и имя (URN) ресурса, к которому применяется данный метод. Сообщения передаются в формате, подобному используемому электронной почтой согласно определениям MIME (Многоцелевых Расширений Электронной Почты).

HTTP также используется как обобщенный протокол связи между агентами пользователей (user agents) и прокси-серверами/шлюзами (proxies/gateways) или другими Интернет-сервисами, включая такие как SMTP, NNTP, FTP, Gopher и WAIS. Таким образом, HTTP определяет основы многосредного доступа к ресурсам для разнообразных приложений. Более подробная терминология приведена в прил. 1.

9.4.2. ОБЩЕЕ ОПИСАНИЕ ПРОТОКОЛА HTTP

Протокол HTTP – это протокол запросов/ответов. Клиент посылает по соединению запрос серверу, содержащий: метод запроса, URI, версию протокола, MIME-подобное сообщение, включающее модификаторы запроса, клиентскую информацию и, возможно, тело запроса. Сервер отвечает строкой состояния, включающей версию протокола сообщения, кодом успешного выполнения или ошибки, MIME-подобным сообщением, содержащим информацию о сервере, метаинформацию объекта и, возможно, тело объекта.

Большинство HTTP соединений, инициализируется агентом пользователя и состоит из запроса, который нужно применить к ресурсу на некотором первоначальном сервере. В самом простом случае, он может быть выполнен посредством одиночного соединения между агентом пользователя и первоначальным сервером, как показано на рис. 9.2.

Более сложная ситуация возникает, когда в цепочке запросов/ответов присутствует один или несколько посредников. Существуют три основных разновидности посредников: прокси-сервера, шлюзы, и туннели. Прокси-сервер является агентом-посредником, который получает запросы на некоторый URI в абсолютной форме, изменяет все сообщение или его часть и отсылает измененный запрос серверу, идентифицированному URI. Шлюз – это принимающий агент, действующий как бы на уровень выше некоторого другого сервера(ов) и при необходимости транслирующий запросы в протокол основного сервера. Туннель действует как реле (relay) между двумя соединениями не изменяя сообщений; туннели используются, когда связь нужно производить через посредника (например firewall), который не понимает содержание сообщений.

Рис. 9.2. Схема простого соединения в протоколе HTTP

На рис. 9.3 показаны три посредника (A, B и C) между агентом пользователя и первоначальным сервером. Запросы и ответы передаются через четыре отдельных соединения. Это отличие важно, так как

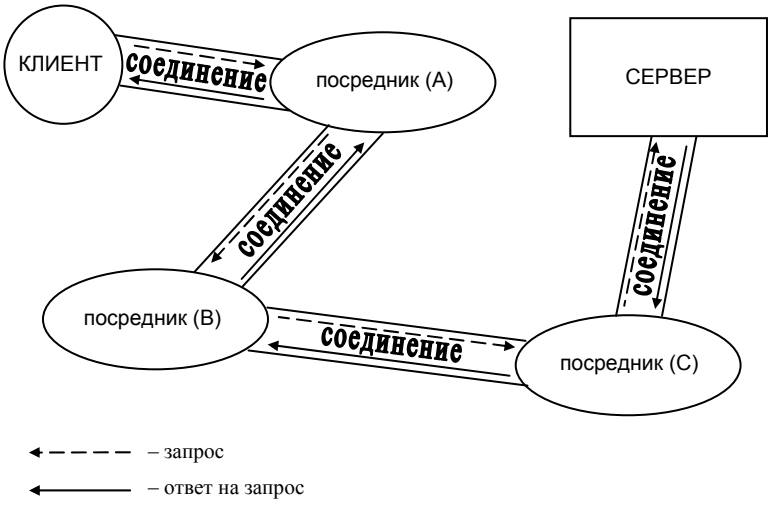

Рис. 9.3. Диаграмма соединения с использованием посредников

некоторые опции HTTP соединения применимы только к соединению с ближайшим не туннельным соседом, некоторые только к конечным точкам цепочки, а некоторые ко всем соединениям в цепочке. Хотя эта диаграмма линейна, каждый участник может быть задействован в нескольких соединениях одновременно. Например, B может получать запросы от других клиентов, а не только от A, и/или пересылать запросы серверам, отличным от C, в то же время, когда он обрабатывает запрос A.

Любая сторона соединения, которая действует не как туннель, может использовать внутренний кэш для обработки запросов. Эффект кэша заключается в том, что цепочка запросов/ответов сокращается, если один из участников в цепочке имеет кэшированный ответ, удовлетворяющий данному запросу. Далее на рис. 9.4 показана цепочка, возникающая в том случае, когда B имеет кэшированую копию раннего ответа O (полеченного через C) на запрос, и который не был кэширован ни UA, ни A.

Не все ответы полезно кэшировать, а некоторые запросы могут содержать модификаторы, которые указывают специальные требования, управляющие поведением кэша.

Фактически, имеется широкое разнообразие архитектур и конфигураций кэшей и прокси-серверов, разрабатываемых в настоящее время или развернутых в World Wide Web; эти системы включают национальные иерархии прокси-кэшей, которые сохраняют пропускную способность межокеанских каналов, системы, которые распространяют по

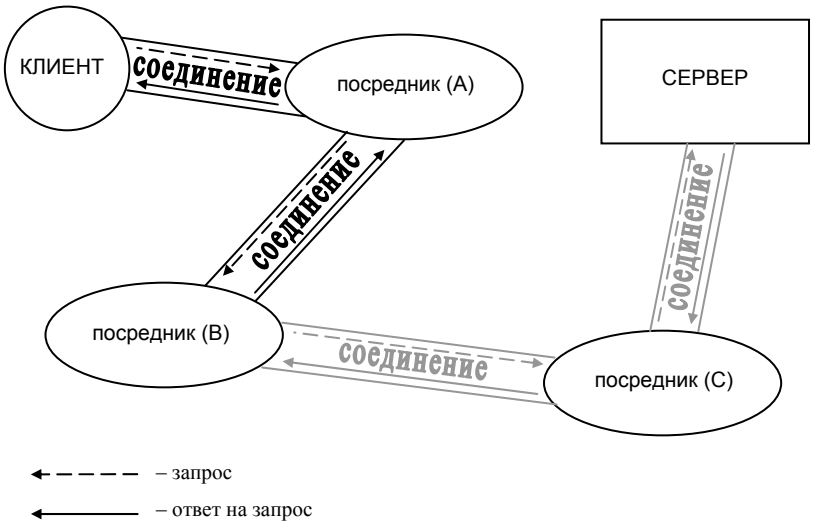

Рис. 9.4. Кэширование HTTP запросов

многим адресам содержимое кэша, организации, которые распространяют подмножества кэшируемых данных на CD-ROM, и так далее. HTTP системы используются в корпоративных интранет-сетях с высокоскоростными линиями связи, и для доступа через PDA с маломощными радиолиниями и неустойчивой связью. Цель HTTP/1.1 состоит в поддержании широкого многообразия конфигураций, уже построенных при введении ранних версий протокола, а также в удовлетворении потребностей разработчиков web приложений, требующих все более высокой надежности.

HTTP соединение обычно происходит посредством TCP/IP соединений. Заданный по умолчанию порт TCP-80, но могут использоваться и другие порты (например: 8080, 8081). HTTP также может быть реализован посредством любого другого протокола Интернет, или других сетей. HTTP необходима только надежная передача данных, следовательно может использоваться любой протокол, который гарантирует надежную передачу данных; отображение структуры запроса и ответа HTTP/1.1 на транспортные модули данных рассматриваемого протокола – вопрос, не решается на уровне самого протокола.

Большинство реализаций HTTP/1.0 использовало новое соединение для каждого обмена запросом/ответом. В HTTP/1.1, установленное соединение может использоваться для одного или нескольких обменов запросом/ответом, хотя соединение может быть закрыто по ряду причин.

9.4.3. ПАРАМЕТРЫ ПРОТОКОЛА HTTP

HTTP использует схему нумерации типа "<major>.<minor>", для указания версии протокола. Стратегия версификации протокола предназначена для того, чтобы позволить отправителю указать формат сообщения и свои способности понимания для дальнейшей HTTP связи, прежде чем он получит что-либо посредством этой связи. При добавлении компонентов сообщения, которые не воздействуют на процесс связи, или компонентов, которые добавляются только к расширяемым значениям поля, номер версии не меняется. Когда внесенные в протокол изменения добавляют возможности, которые не изменяют общий алгоритм анализа сообщений, но расширяют семантику сообщения и подразумевают дополнительные возможности отправителя, увеличивается <minor> номер. Когда изменяется формат сообщения протокола, увеличивается <major> номер.

Версия HTTP сообщения обозначается полем HTTP-version в первой строке сообщения:

HTTP-Version = "HTTP" "/" 1*DIGIT "." 1*DIGIT

Major и minor числа должны обрабатываться как отдельные целые числа и что каждое может состоять более чем из одной цифры. Таким образом, HTTP/2.4 – более низкая версия, чем

HTTP/2.13, которая, в свою очередь, ниже чем HTTP/12.3. Нули должны игнорироваться получателями и не должны посылаться.

Приложения, посылающие сообщения запросов или ответов, которые описывает спецификация HTTP/1.1, должны указывать версию HTTP (HTTP-version) "HTTP/1.1". Использование этого номера версии указывает, что посылающее приложение, по крайней мере, условно совместимо с этой спецификацией.

HTTP версия приложения – это самая высокая HTTP версия, с которой приложение является, по крайней мере, условно совместимым с ним.

Приложения, реализующие прокси-сервера и шлюзы, должны обрабатывать протокольные сообщения различных версий. Начиная с момента, когда версия протокола указывает возможности отправителя, прокси-сервер/шлюз никогда не должен посылать сообщения, версия которых больше, чем HTTP версия отправителя; если получена более высокая версия запроса, то прокси-сервер/шлюз должен или понизить версию запроса, вернув сообщение об ошибке, или переключиться на туннельное поведение. У запросов, версия которых ниже, чем HTTP версия прокси-сервера/шлюза, можно перед пересылкой увеличить версию; ответ прокси-сервера/шлюза на этот запрос должен иметь ту же самую major версию, что и запрос.

Преобразование версий HTTP может включать модификацию полей заголовка, требуемых или запрещенных этими версиями.

URI известны под многими именами: WWW адреса, Универсальные Идентификаторы Документов, Универсальные Идентификаторы Ресурсов (URI), и, в заключение, как комбинация Единообразных Идентификаторов Ресурсов (Uniform Resource Locators, URL) и Единообразных Имен Ресурсов (Uniform Resource Names, URN). HTTP определяет URL просто как строку определенного формата, которая идентифицирует ресурс посредством имени, расположения, или любой другой характеристики.

URI в HTTP могут представляться в абсолютной форме (absolute URI) или относительно некоторого известного основного URI (relative URI), в зависимости от контекста их использования. Эти две формы различаются тем, что абсолютные URI всегда начинаются с имени схемы с двоеточием:

URI = (absoluteURI | relativeURI) ["#" fragment]
absoluteURI = scheme ":" *(uchar | reserved)
relativeURI = net_path | abs_path | rel_path
net_path = "//" net_loc [abs_path] abs_path = "/" rel_path rel_path =
[path] [";" params] ["?" query]
path = fsegment *("/" segment) fsegment = 1*pchar segment = *pchar
params = param *(";" param) param = *(pchar | "/")
scheme = 1*(ALPHA | DIGIT | "+" | "-" | ".") net_loc = *(pchar | ";" | "?")
query = *(uchar | reserved) fragment = *(uchar | reserved)
pchar = uchar | ":" | "@" | "&" | "=" | "+" uchar = unreserved | escape
unreserved = ALPHA | DIGIT | safe | extra | national
escape = "%" HEX HEX reserved = ";" | "/" | "?" | ":" | "@" | "&" | "="
| "+" extra = "!" | "*" | "'" | "(" | ")" | "," safe =
"$" | "-" | "_" | "." unsafe = CTL | SP | <"> | "#" | "%" | "<" | ">"
national = <любой OCTET за исключением ALPHA,
DIGIT, reserved, extra, safe, и unsafe октетов>

Полная информация о синтаксисе и семантике URL содержится в RFC 1738 и RFC 1808. Нормальная запись Бекуса-Наура включает национальные символы, недозволенные в правильных URL, определенных RFC 1738, так как HTTP серверы позволяют использовать для представления части rel_path адресов набор не зарезервированных символов, и, следовательно, HTTP прокси-сервера могут получать запросы URI, не удовлетворяющие RFC 1738.

Протокол HTTP не накладывает никаких ограничений на длины URI. Серверы должны обрабатывать URI любого ресурса, любой длины, который они обслуживают, и им надлежит обрабатывать URI неограниченной длины, если они обслуживают сервера, основанные на методе GET, которые могут создавать такой URI. Серверу следует возвращать код состояния 414 (URI

запроса слишком длинный, Request-URI Too Long), если URI длиннее, чем сервер в состоянии обработать.

Серверы должны обращать внимание на URI, которые имеют длину более 255 байтов, потому что некоторые старые клиенты или прокси-сервера могут неправильно поддерживать эти длины.

"Http" схема используется для доступа к сетевым ресурсам при помощи протокола HTTP. Этот раздел определяет схемо-определенный синтаксис и семантику для HTTP URL:

http_URL = "http:" "//" host [":" port] [abs_path]
host = <допустимое доменное имя машины или IP адрес
(в точечно десятичной форме), как определено в разделе 2.1
RFC 1123>
port = *DIGIT

Если порт пуст или не задан – используется порт 80. Это означает, что идентифицированный ресурс размещен в сервере, ожидающем TCP соединений на специфицированном порте port, компьютера host, и запрашиваемый URI ресурса – abs_path. Использования IP адресов в URL следует избегать, насколько это возможно (RFC 1900). Если abs_path не представлен в URL, он должен рассматриваться как "/" при вычислении запрашиваемого URI (Request-URI) ресурса.

При сравнении двух URI, чтобы решить соответствуют ли они друг другу или нет, клиенту следует использовать чувствительное к регистру пооктетное (octet-by-octet) сравнение этих URI, со следующими исключениями:

- **порт, который пуст или не указан, эквивалентен заданному по умолчанию порту для этого URI;**
- **сравнение имен хостов должно производиться без учета регистра;**
- **сравнение имен схем должно производиться без учета регистра;**
- **пустой abs_path эквивалентен "/";**
- **символы, отличные от тех, что находятся в "зарезервированных" ("reserved") и "опасных" ("unsafe") наборах эквивалентны их представлению как ""%" HEX HEX ".**

Например, следующие три URI эквивалентны:

http://abc.com:80/~smith/home.html
http://ABC.com/%7Esmith/home.html
http://ABC.com:/%7esmith/home.html

HTTP использует то же самое определение термина "кодовая таблица", которое определено для MIME:

Термин "кодовая таблица" используется, чтобы сослаться на метод, использующий одну или несколько таблиц для преобразования последовательности октетов в последовательность символов. Стоит отметить, что однозначное преобразование в обратном направлении не требуется, и что не все символы могут быть доступны в данной кодовой таблице, и что кодовая таблица может обеспечивать более чем одну последовательность октетов для представления специфических символов. Это определение допускает различные виды кодирования символов, от простых однотабличных отображений типа US-ASCII до сложных методов, переключающих таблицы, наподобие тех, которые используют методики ISO 2022. Однако определение, связанное с именем кодовой таблицы MIME должно полностью определять отображение, которое преобразует октеты в символы. В частности, использование внешней информации профилирования для определения точного отображения не разрешается.

Кодовые таблицы HTTP идентифицируются лексемами, не чувствительными к регистру. Полный набор лексем определен реестром кодовых таблиц IANA [19]:

charset = token

Хотя HTTP позволяет использовать в качестве значения charset произвольную лексему, любая лексема, которая имеет предопределенное значение в реестре кодовых таблиц IANA, должна представлять набор символов, определенный в данном реестре. Приложениям следует ограничить использование символьных наборов теми, которые определены в реестре IANA.

Значение кодирования содержимого указывает какое преобразование кодирования было или будет применено к объекту. Кодирование содержимого используется, прежде всего, для сжатия

или другого полезного преобразования документа без потери идентификации основного медиатипа и информации. Часто объект сохраняется в кодированной форме, затем передается, а потом декодируется получателем:

content-coding = token

Все значения кодирования содержимого (content-coding) не чувствительны к регистру. HTTP/1.1 использует значения кодирования содержимого (content-coding) в полях заголовка Accept-Encoding и Content-Encoding. Хотя значение описывает кодирование содержимого, но, что более важно – оно указывает, какой механизм декодирования потребуется для обратного процесса.

Internet Assigned Numbers Authority (IANA) действует как реестр для значений лексем кодирования содержимого (content-coding). Первоначально реестр содержал следующие лексемы:

- **gzip – формат кодирования, производящий сжатие файла программой "gzip" (GNU zip), описанный в RFC 1952. Это формат Lempel-Ziv кодирования (LZ77) с 32 разрядным CRC.**
- **Compress – формат кодирования, производимый общей программой "compress" для сжатия UNIX файлов. Это формат адаптивного Lempel-Ziv-Welch кодирования (LZW).**

Конечно, использовать названия программ для идентификации форматов кодирования не желательно и может пересекаться с форматами, которые возникнут в последствии. Их использование объясняется исторической практикой. Для совместимости с предыдущими реализациями HTTP, приложения должны рассматривать "x-gzip" и "x-compress" как эквиваленты "gzip" и "compress", соответственно:

deflate – формат zlib, определенный в 1950, в комбинации с механизмом сжатия "deflate", описанным в RFC 1951.

Новая лексема значения кодирования содержимого (content-coding) должна быть зарегистрирована; чтобы обеспечить взаимодействие между клиентами и серверами, спецификация алгоритма кодирования содержимого, необходимого для определения нового значения, должна быть открыто опубликована и адекватна для независимой реализации, а также соответствовать цели кодирования содержимого определенного в этом разделе.

9.4.4. HTTP СООБЩЕНИЕ (HTTP MESSAGE)

HTTP сообщения делятся на запросы клиента серверу и ответы сервера клиенту:

HTTP-message = Request | Response ; сообщения HTTP/1.1

Сообщения запроса и ответа используют обобщенный формат сообщения RFC 822 для пересылки объектов (полезной нагрузки сообщения). Оба типа сообщений выглядят следующим образом: сначала идет начальная строка (start-line), затем один или несколько полей заголовка (называемых также просто "заголовки"), затем пустая строка (то есть строка, равная CRLF), указывающая конец полей заголовка, а затем, возможно, тело сообщения:

generic-message = start-line *message-header CRLF [message-body]
start-line = Request-Line | Status-Line

В интересах ошибкоустойчивости, серверам следует игнорировать все пустые строки, полученные перед строкой запроса (Request-Line). Другими словами, если сервер читает поток протокола и в самом начале сообщения получает CRLF, то ему следует этот CRLF игнорировать.

Некоторые ошибочные реализации HTTP/1.0 клиентов генерируют дополнительные CRLF после запроса POST. Стоит вновь повторить, что это явно запрещено нормальной записью Бекуса-Наура. HTTP/1.1 клиент не должен добавлять дополнительные CRLF перед запросом и после него.

Поля заголовков HTTP, которые включают поля общих заголовков (general-header), заголовков запроса (request-header), заголовков ответа (response-header), и заголовков объекта (entity-header), имеют такой же обобщенный формат, как тот, что описан в разделе 3.1 RFC 822. Каждое поле заголовка состоит из имени поля, двоеточия (":") и значения поля. Имена полей не чувствительны к регистру. Значению поля может предшествовать любое число LWS, хотя предпочтите-

лен одиночный SP. Поля заголовка могут занимать несколько строк. При этом каждая следующая строка продолжения начинается, по крайней мере, одним SP или HT. Приложениям следует придерживаться "общей формы" ("common form") при генерации HTTP конструкций, так как могут существовать реализации, которые не в состоянии принимать что-либо кроме общих форм:

message-header = field-name ":" [field-value] CRLF
field-name = token field-value = *(field-content | LWS)
field-content = <октеты, составляющие значение поля и состоящие или из *TEXT или из комбинаций лексем, tspecials, и quoted-string>

Порядок, в котором получены поля заголовка с различными именами не имеет значения. Однако "хорошей практикой" является то, что сначала посылаются поля общих заголовков, затем поля заголовков запроса или заголовков ответа, и, наконец, поля заголовков объекта.

Несколько полей заголовка с одинаковыми именами могут присутствовать в сообщении тогда и только тогда, когда все значения полей, входящих в заголовок, определяют разделенный запятыми список [то есть #(value)]. Должно быть возможно объединить несколько таких полей заголовка в одну пару "имя поля: значение поля" (не изменяя этим семантику сообщения) путем присоединения каждого последующего значения поля к первому через запятые. Порядок, в котором получены поля с одинаковыми именами, имеет значение для интерпретации объединенного значения поля, и, следовательно, прокси-сервер не должен изменять порядок значений этого поля при пересылке.

Тело HTTP сообщения (message-body), если оно присутствует, используется для передачи тела объекта, связанного с запросом или ответом. Тело сообщения (message-body) отличается от тела объекта (entity-body) только в том случае, когда применяется кодирование передачи, что указывается полем заголовка Transfer-Encodingы:

message-body = entity-body | <entity-body закодированно согласно Transfer-Encoding>

Поле Transfer-Encoding должно использоваться для указания любого кодирования передачи, примененного приложением в целях гарантирования безопасной и правильной передачи сообщения. Поле Transfer-Encoding – это свойство сообщения, а не объекта, и, таким образом, может быть добавлено или удалено любым приложением в цепочке запросов/ответов.

Правила, устанавливающие допустимость тела сообщения в сообщении, различаются для запросов и ответов.

Присутствие тела сообщения в запросе отмечается добавлением к заголовкам запроса поля заголовка Content-Length или Transfer-Encoding. Тело сообщения (message-body) может быть добавлено в запрос только тогда, когда метод запроса допускает тело объекта (entity-body).

Включать или не включать тело сообщения (message-body) в сообщение ответа зависит как от метода запроса, так и от кода состояния ответа. Все ответы на запрос с методом HEAD не должны включать тело сообщения (message-body), даже если присутствуют поля заголовка объекта (entity-header), заставляющие поверить в присутствие объекта. Никакие ответы с информационными кодами состояния 1xx, кодом 204 (Нет содержимого, No Content) и кодом 304 (Не модифицирован, Not Modified) не должны содержать тела сообщения (message-body). Все остальные ответы содержат тело сообщения, даже если оно имеет нулевую длину.

Когда тело сообщения (message-body) присутствует в сообщении, длина этого тела определяется одним из следующих методов (в порядке старшинства):

1 **Любое сообщение ответа, которое не должно включать тело сообщения (message-body) (например ответы с кодами состояния 1xx, 204, 304 и все ответы на запрос HEAD) всегда завершается пустой строкой после полей заголовка, независимо от полей заголовка объекта (entity-header fields), представленных в сообщении.**

2 **Если поле заголовка Transfer-Encoding присутствует и указывает на применение кодирования передачи "chunked", то длина определяется кодированием по кускам (chunked encoding).**

3 **Если поле заголовка Content-Length присутствует, то его значение представляет длину тела сообщения (message-body) в байтах.**

4 **Если сообщение использует медиатип "multipart/byteranges", который саморазграничен, то он и определяет длину. Этот медиа тип не должен использоваться, если отправитель не знает, способен ли получатель его обработать; присутствие в запросе заголовка Range с несколькими спецификаторами диапазонов байтов (byte-range) подразумевает, что клиент может анализировать multipart/byteranges ответы.**

5 **Длина определяется закрытием соединения сервером. (Закрытие соединения не может использоваться для указания конца тела запроса, так как в этом случае у сервера не остается никакой возможности послать обратно ответ).**

Для совместимости с HTTP/1.0 приложениями HTTP/1.1 запросы, содержащие тело сообщения (message-body) должны включать допустимое поле заголовка Content-Length, пока не известно, что сервер является HTTP/1.1 совместимым. Если запрос содержит тело сообщения (message-body), и Content-Length не указано, серверу следует послать ответ с кодом состояния 400 (Испорченный Запрос, Bad Request), если он не может определить длину сообщения, или с кодом состояния 411 (Требуется длина, Length Required), если он настаивает на получении Content-Length.

Все HTTP/1.1 приложения, которые получают объекты, должны понимать кодирование передачи типа "chunked"; таким образом, разрешается использование данного механизма для сообщений, длина которых не может быть определена заранее.

Сообщения не должны одновременно включать и поле заголовка Content-Length и применять кодирование передачи типа "chunked". Если поступило сообщение с полем Content-Length и закодированное с применением кодирования передачи "chunked", то поле Content-Length должно игнорироваться.

Если поле Content-Length присутствует в сообщении, которое допускает наличие тела сообщения (message-body), то значение поля должно точно соответствовать числу октетов в теле сообщения. HTTP/1.1 агенты пользователя должны информировать пользователя в случае получения и обнаружения недопустимой длины.

Имеется несколько полей заголовка, которые применяются как для сообщений запросов, так и для сообщений ответов, но которые не применяются к передаваемому объекту. Эти поля заголовка применяются только к передаваемому сообщению:

general-header = Cache-Control | Connection | Date | Pragma | Transfer-Encoding | Upgrade | Via

Имена общих полей заголовка (general-header fields) могут быть надежно расширены только в сочетании с изменением версии протокола. Однако, новые или экспериментальные поля заголовка могут получить семантику общих полей заголовка (general-header fields), если все стороны соединения распознают их как общие поля заголовка. Нераспознанные поля заголовка обрабатываются как поля заголовка объекта (entity-header).

9.4.5. ЗАПРОС (REQUEST)

Сообщение запроса сервера клиентом содержит в первой строке: метод, который нужно применить к ресурсу, идентификатор ресурса и используемую версию протокола:

Request = Request-Line*(general-header | request-header | entity-header) CRLF [message-body]

Строка запроса (Request-Line) начинается с лексемы метода, затем следует запрашиваемый URI (Request-URI), версия протокола и CRLF. Эти элементы разделяются SP. В строке запроса (Request-Line) не допустимы CR и LF, исключение составляет конечная последовательность CRLF:

Request-Line = Method SP Request-URI SP HTTP-Version CRLF

Лексема метода указывает метод, который нужно применить к ресурсу, идентифицированному запрашиваемым URI (Request-URI). Метод чувствителен к регистру:

Method = "OPTIONS" | "GET" | "HEAD" | "POST" | "PUT" | "DELETE" | "TRACE" | extension-method
extension-method = token

Список методов, применимых к ресурсу, может быть указан в поле заголовка Allow. Возвращаемый код состояния ответа всегда сообщает клиенту, допустим ли метод для ресурса в настоящее время, так как набор допустимых методов может изменяться динамически. Серверам следует возвратить код состояния 405 (Метод не допустим, Method Not Allowed), если метод известен серверу, но не применим для запрошенного ресурса, и 501 (Не реализовано, Not Implemented), если метод не распознан или не реализован сервером. Список методов, известных серверу, может быть указан в поле заголовка ответа Public.

Методы GET и HEAD должны поддерживаться всеми универсальными (general-purpose) серверами. Остальные методы опциональны.

URI запроса (Request-URI) – это Единообразный Идентификатор Ресурса (URL), который идентифицирует ресурс запроса:

Request-URI = "*" | absoluteURI | abs_path

Три опции для URI запроса (Request-URI) зависят от характера запроса. Звездочка "*" означает, что запрашивается не специфический ресурс, а сервер непосредственно, и допустим только в том случае, когда используемый метод не обязательно обращается к ресурсу. В качестве примера:

OPTIONS * HTTP/1.1

absoluteURI необходим, когда запрос производится через прокси-сервер. Прокси-сервер перенаправляет запрос на сервер или обслуживает его, пользуясь кэшем, и возвращает ответ. Прокси-сервер может переслать запрос другому прокси-серверу или непосредственно серверу, определенному absoluteURI. Чтобы избежать зацикливания запроса прокси-сервер должен быть способен распознавать все имена сервера, включая любые псевдонимы, локальные разновидности, и числовые IP адреса. Request-Line может быть, например, таким:

GET http://www.w3.org/pub/WWW/TheProject.html HTTP/1.1

Чтобы обеспечить переход к absoluteURI во всех запросах в будущих версиях HTTP, все HTTP/1.1 сервера должны принимать absoluteURI в запросах, хотя HTTP/1.1 клиенты будут генерировать их только в запросах к прокси-серверам.

Наиболее общая форма Request-URI используется для идентификации ресурса на первоначальном сервере или шлюзе. В этом случае абсолютный путь URI должен быть передан как Request-URI, а сетевое расположение URI (net_loc) должно быть передано в поле заголовка Host. Для последнего примера клиент, желающий получить ресурс непосредственно с первоначального сервера, должен создать TCP соединение на 80 порт хоста "www.w3.org" и послать строки:

GET /pub/WWW/TheProject.html HTTP/1.1 Host: www.w3.org

и далее остаток запроса. Абсолютный путь не может быть пустым; если оригинальный URI пуст, то он должен запрашиваться как "/" (корневой каталог сервера).

Если прокси-сервер получает запрос без пути в Request-URI, и метод запроса допускает форму запроса "*", то последний прокси-сервер в цепочке запросов должен передать запрос, в котором Request-URI равен "*". Например, запрос

OPTIONS http://www.ics.uci.edu:8001 HTTP/1.1

был бы передан прокси-сервером в виде

OPTIONS * HTTP/1.1 Host: www.ics.uci.edu:8001

после соединения с портом 8001 хоста "www.ics.uci.edu".

Первоначальный сервер должен декодировать Request-URI, чтобы правильно интерпретировать запрос. Серверам следует отвечать на недопустимые Request-URI соответствующим кодом состояния.

В запросах, пересылаемых прокси-сервером, часть "abs_path" URI запроса (Request-URI) никогда не должна перезаписываться, за исключением случая, отмеченного выше, когда пустой abs_path заменяется на "*", независимо от внутренней реализации прокси-сервера.

Правило "ничто не перезаписывать" предохраняет прокси-сервера от изменения значения запроса, в котором первоначальный сервер неправильно использует не зарезервированные символы URL для своих целей.

Первоначальные HTTP/1.1 сервера должны учитывать, что точный ресурс, идентифицируемый интернет-запросом определяется путем исследования запрашиваемого URI (Request-URI) и поля заголовка Host.

Первоначальный сервер, который не различает ресурсы по запрошенному хосту (host), может игнорировать значение поля заголовка Host.

Первоначальный сервер, который различает ресурсы на основании запрошенного хоста (host) (иногда называемые виртуальными хостами или vanity hostnames) должен пользоваться следующими правилами для определения ресурса, запрошенного в HTTP/1.1 запросе:

1 Если Request-URI – это absoluteURI, то хост – это часть Request-URI. Любые значения поля заголовка Host в запросе должны игнорироваться.

2 Если Request-URI – не absoluteURI, а запрос содержит поле заголовка Host, то хост определяется значением поля заголовка Host.

3 Если хоста, определенного правилами 1 или 2 не существует на сервере, кодом состояния ответа должен быть 400 (Испорченный Запрос, Bad Request).

Получатели HTTP/1.0 запроса, в котором отсутствует поле заголовка Host, могут попытаться использовать эвристику (например, исследовать путь в URI на предмет уникальности на каком-либо из хостов) для определения какой именно ресурс запрашивается.

Поля заголовка запроса позволяют клиенту передать серверу дополнительную информацию о запросе и о самом клиенте. Эти поля действуют как модификаторы запроса с семантикой, эквивалентной параметрам вызова методов в языках программирования:

request-header = Accept | Accept-Charset | Accept-Encoding | Accept-Language | Authorization | From | Host | If-Modified-Since | If-Match | If-None-Match | If-Range | If-Unmodified-Since | Max-Forwards | Proxy-Authorization | Range | Referer | User-Agent

Множество имен полей заголовка запроса (Request-header) может быть надежно расширено только в сочетании с изменением версии протокола. Однако, новые или экспериментальные поля заголовка могут получить семантику полей заголовка запроса (Request-header), если все стороны соединения распознают их как поля заголовка запроса (Request-header). Нераспознанные поля заголовка обрабатываются как поля заголовка объекта (entity-header).

9.4.6. ОТВЕТ (RESPONSE)

После получения и интерпретации сообщения запроса, сервер отвечает сообщением HTTP ответа:

Response = Status-Line *(general-header | response-header | entity-header) CRLF [message-body]

Первая строка ответа – это строка состояния (Status-Line). Она состоит из версии протокола (HTTP-Version), числового кода состояния (Status-Code) и поясняющей фразы (Reason-Phrase)

разделенных символами SP. CR и LF не допустимы в Status-Line, за исключением конечной последовательности CRLF:

Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF

Элемент код состояния (Status-Code) – это целочисленный трехразрядный код результата попытки понять и выполнить запрос. Эти коды полностью определены в разделе 10. Поясняющая фраза (Reason-Phrase) предназначена для короткого текстового описания кода состояния. Код состояния (Status-Code) предназначен для использования автоматами, а поясняющая фраза предназначена для живых пользователей. От клиента не требуется исследовать или отображать поясняющую фразу (Reason-Phrase).

Первая цифра кода состояния определяет класс ответа. Последние две цифры не имеют определенной роли в классификации. Имеется пять значений первой цифры:

- **1xx: Информационные коды – запрос получен, продолжается обработка.**
- **2xx: Успешные коды – действие было успешно получено, понято и обработано.**
- **3xx: Коды перенаправления – для выполнения запроса должны быть предприняты дальнейшие действия.**
- **4xx: Коды ошибок клиента – запрос имеет плохой синтаксис или не может быть выполнен.**
- **5xx: Коды ошибок сервера – сервер не в состоянии выполнить правильный запрос.**

Конкретные значения числовых кодов состояния, определенных в HTTP/1.1, и примерный набор соответствующих поясняющих фраз (Reason-Phrase) приводятся ниже. Поясняющие фразы (Reason-Phrase), перечисленные здесь являются рекомендуемыми, но могут быть заменены на эквивалентные, не влияя на протокол.

Status-Code = "100" ; Продолжать, Continue |
"101" ; Переключение протоколов ; Switching Protocols |
"200" ; OK |
"201" ; Создан, Created |
"202" ; Принято, Accepted |
"203" ; Не авторская информация ; Non-Authoritative Information |
"204" ; Нет содержимого, No Content |
"205" ; Сбросить содержимое, Reset ; Content |
"206" ; Частичное содержимое, Partial ; Content |
"300" ; Множественный выбор, Multiple ; Choices |
"301" ; Постоянно перемещен, Moved ; Permanently |
"302" ; Временно перемещен, Moved ; Temporarily |
"303" ; Смотреть другой, See Other |
"304" ; Не модифицирован, Not Modified |
"305" ; Используйте прокси-сервер, Use ; Proxy |
"400" ; Испорченный запрос, Bad Request |
"401" ; Несанкционированно, Unauthorized |
"402" ; Требуется оплата, Payment ; Required |
"403" ; Запрещено, Forbidden |
"404" ; Не найден, Not Found |
"405" ; Метод не допустим, Method Not ; Allowed |
"406" ; Не приемлем, Not Acceptable |
"407" ; Требуется установление ; подлинности через прокси-сервер, ; Proxy Authentication Required |
"408" ; Истекло время ожидания запроса ; Request Timeout |
"409" ; Конфликт, Conflict |
"410" ; Удален, Gone |
"411" ; Требуется длина, Length Required |
"412" ; Предусловие неверно ; Precondition Failed |
"413" ; Объект запроса слишком большой ; Request Entity Too Large |
"414" ; URI запроса слишком длинный ; Request-URI Too Long |

"415" ; Неподдерживаемый медиатип ; Unsupported Media Type |
"500" ; Внутренняя ошибка сервера ; Internal Server Error |
"501" ; Не реализовано, Not Implemented |
"502" ; Ошибка шлюза, Bad Gateway |
"503" ; Сервис недоступен, Service ; Unavailable |
"504" ; Истекло время ожидания от шлюза ; Gateway Timeout |
"505" ; Не поддерживаемая версия HTTP ; HTTP Version Not Supported | extension-code
extension-code = 3DIGIT
Reason-Phrase = *<TEXT не включающий CR, LF>

Коды состояния HTTP расширяемы. HTTP приложениям не требуется понимать значение всех зарегистрированных кодов состояния, хотя их понимание очень желательно. Приложения должны понимать класс любого кода состояния, который обозначается первой цифрой, и обрабатывать любой нераспознанный ответ как эквивалентный коду состояния x00 этого класса, за исключением тех случаев, когда нераспознанный ответ не должен кэшироваться. Например, если клиентом получен и не был распознан код состояния 431, то он может безопасно считать, что в запросе что-то было неправильно и обрабатывать ответ, как если бы был получен код состояния 400. В таких случаях агентам пользователя следует представить пользователю объект, возвращенный в ответе, так как этот объект, вероятно, включает читабельную для человека информацию, которая поясняет необычное состояние.

Поля заголовка ответа (response-header fields) позволяют серверу передавать дополнительную информацию об ответе, которая не может быть помещена в строку состояния Status-Line. Эти поля заголовка дают информацию о сервере и о дальнейшем доступе к ресурсу, указанному этим Request-URI:

response-header = Age | Location | Proxy-Authenticate | Public | Retry-After | Server | Vary | Warning | WWW-Authenticate

Множество имен полей заголовка ответа (Response-header) может быть надежно расширено только в сочетании с изменением версии протокола. Однако, новые или экспериментальные поля заголовка могут получить семантику полей заголовка ответа (Response-header), если все стороны соединения распознают их как поля заголовка ответа (Response-header). Нераспознанные поля заголовка обрабатываются как поля заголовка объекта (entity-header).

9.4.7. ОБЪЕКТ (ENTITY)

Сообщения запросов и ответов могут передать объект, если это не запрещено методом запроса или кодом состояния ответа. Объект состоит из полей заголовка объекта (entity-header) и тела объекта (entity-body), хотя некоторые ответы могут включать только заголовки объекта (entity-headers).

Этот раздел относится как к отправителю, так и к получателю, то есть к клиенту или серверу, в зависимости от того, кто посылает, а кто получает объект.

Поля заголовка объекта (Entity-header fields) определяют *опциональную метаинформацию* о теле объекта (entity-body) или, если тело отсутствует, о *ресурсе*, идентифицированном запросом:

entity-header = Allow | Content-Base | Content-Encoding | Content-Language | Content-Length | Content-Location | Content-MD5 | Content-Range | Content-Type | ETag | Expires | Last-Modified | extension-header
extension-header = message-header

Механизм расширения полей заголовка позволяет вводить дополнительные поля заголовка объекта (entity-header fields), не изменяя протокол, но эти поля могут быть и не распознаны получателем. Получатель должен игнорировать нераспознанные поля заголовка, а прокси-сервер должен просто пересылать их без изменений.

Тело объекта (если оно присутствует) посылается с HTTP запросом или ответом и имеет формат и кодирование, определяемое полями заголовка объекта (entity-header fields):

entity-body = *OCTET

Тело объекта (entity-body) представлено в сообщении только тогда, когда присутствует тело сообщения (message-body). Тело объекта (entity-body) получается из тела сообщения (message-body) декодированием любого кодирования передачи, указанного в поле Transfer-Encoding, которое может быть применено для гарантирования безопасной и правильной передачи сообщения.

Когда в сообщении содержится тело объекта (entity-body), тип данных этого тела определяется полями заголовка Content-Type и Content-Encoding. Они определяют двухуровневую упорядоченную модель кодирования:

entity-body := Content-Encoding(Content-Type(data))

Тип содержимого (Content-Type) определяет медиатип лежащих в основе данных. Кодирование содержимого (Content-Encoding) может использоваться для указания любых дополнительных кодирований содержимого, примененных к данным (обычно с целью сжатия). Кодирование содержимого (Content-Encoding) является свойством запрошенного ресурса. По умолчанию никакого кодирования не задано.

В любое HTTP/1.1 сообщение, содержащее тело объекта (entity-body) включает поле заголовка Content-Type, определяющее медиатип этого тела. В том и только в том случае, когда медиатип не указан в поле Content-Type, получатель может попытаться самостоятельно определить медиатип, проверяя содержимое и/или расширение (расширения) в URL, используемого для идентификации ресурса. Если медиатип остался нераспознан, получателю следует обрабатывать его как тип "application/octet-stream".

Длина тела объекта (entity-body) – это длина тела сообщения (message-body), полученного после декодирования всех кодирований передачи.

9.4.8. СОЕДИНЕНИЯ (CONNECTIONS)

До введения в протокол постоянных соединений для запроса каждого URL устанавливалось отдельное TCP соединение, что увеличивало нагрузку на HTTP сервера и вызывало перегрузку сетей. Использование встроенных изображений и других связанных данных часто требует от клиента инициировать несколько запросов к одному серверу за короткий промежуток времени.

Постоянные HTTP соединения имеют ряд преимуществ:

- **открытие и закрытие меньшего количества TCP соединений экономит время центрального процессора и память, используемую для управляющих блоков протокола TCP;**
- **HTTP запросы и ответы может быть конвейеризованы в соединении. Конвейерная обработка позволяет клиенту делать несколько запросов не ожидая ответа на каждый, позволяет пользоваться единственным TCP соединением более эффективно, с меньшими затратами времени;**
- **загрузка сети уменьшается с уменьшением числа пакетов, необходимых для открытия TCP соединений, и, следовательно, предоставляет протоколу TCP достаточно времени для определения состояния перегрузки сети;**
- **HTTP может развиваться более элегантно, так как ошибки можно сообщать без закрытия TCP соединения в качестве штрафа. Клиенты, использующие будущие версии HTTP могли бы оптимистично пробовать новые возможности, а при связи со старым сервером, повторять запрос, используя старую семантику после сообщения об ошибке.**

Значительное отличие HTTP/1.1 от ранних версий HTTP состоит в том, что постоянные соединения являются заданным по умолчанию поведением любого HTTP соединения. То есть если не обозначено иного, клиент может считать, что сервер поддержит постоянное соединение.

Постоянные соединения обеспечивают механизм, согласно которому клиент и сервер могут сообщить о разрыве TCP соединения. Это сигнализируется путем использования поля заголовка Connection.

HTTP/1.1 сервер в праве считать, что HTTP/1.1 клиент не поддерживает постоянное соединение, если посланный в запросе заголовок Connection содержит лексему соединения (connection-token) "close". Если сервер решает закрыть соединение немедленно после посылки ответа, то ему необходимо послать заголовок Connection, который содержит лексему соединения (connection-token) "close".

HTTP/1.1 клиент должен ждать закрытие соединения, но должен держать его открытым на основании того, содержит ли ответ сервера заголовок Connection с лексемой соединения "close". В случае, если клиент не хочет поддерживать соединение для последующих запросов, ему надлежит послать заголовок Connection, содержащий лексему соединения "close".

Если клиент или сервер посылает лексему закрытия соединения "close" в заголовке Connection, то запрос становится последним в соединении.

Чтобы соединение оставалось постоянным, все сообщения, передаваемые по нему должны иметь самоопределенную (self-defined) длину (то есть, не определяемую закрытием соединения).

Клиент, который поддерживает постоянные соединения, умеет производить "конвейерную обработку" запросов (то есть посылать несколько запросов не ожидая ответа на каждый из них). Сервер должен послать ответы на эти запросы в том же самом порядке, в каком были получены запросы.

Клиенты, которые поддерживают постоянные соединения и производят конвейерную обработку немедленно после установления соединения, должны быть готовы повторить соединение, если первая попытка конвейерной обработки дала сбой. Если клиент производит такой повтор, он не должен производить конвейерную обработку прежде чем узнает, что соединение постоянно. Также клиенты должны быть готовы повторить запросы, если сервер закрывает соединение перед посылкой всех соответствующих ответов.

Очень важно, чтобы прокси-сервера правильно реализовывали свойства полей заголовка Connection.

Прокси-сервер должен сообщать о постоянных соединениях отдельно своим клиентам и отдельно первоначальным серверам (или другим прокси-серверам), которые с ним соединены. Каждое постоянное соединение применяется только к одной транспортной связи.

Прокси-сервер не должен устанавливать постоянных соединений с HTTP/1.0 клиентом.

Сервера обычно имеют некоторое значение времени ожидания, после которого они не поддерживают неактивное соединение. Прокси-сервера могут выставлять его значение более высоким, так как, вероятно, клиент сделает больше соединений через этот же сервер. Использование постоянных соединений не вводит никаких ограничений на продолжительность времени ожидания как для клиента, так и для сервера.

Когда у клиента или сервера истекло время ожидания, ему необходимо произвести закрытие транспортного соединения. Как клиентам, так и серверам надлежит постоянно наблюдать за другой стороной на предмет закрытия соединения и отвечать соответственно. Если клиент или сервер не сразу обнаруживает закрытие соединения другой стороной, то это вызывает неоправданную трату ресурсов сети.

Клиент, сервер или прокси-сервер в праве закрыть транспортное соединение в любое время. Например, клиент может начать посылать новый запрос в то время, когда сервер решает закрыть "бездействующее" соединение. С точки зрения сервера, соединение закрывается, в то время как оно было неактивно, но с точки зрения клиента, запрос произошел.

Это означает, что клиенты, серверы и прокси-серверы должны быть в состоянии обрабатывать асинхронные события закрытия. Программному обеспечению клиента следует вновь открыть транспортное соединение и повторно передать прерванный запрос без взаимодействия с пользователем, если метод запроса идемпотентен; другие методы не должны быть повторены автоматически, хотя агенты пользователя могут предложить оператору повторить запрос.

Однако этого автоматического повтора производить не следует, если сбой происходит уже во втором запросе.

Серверам всегда следует отвечать, по крайней мере, на один запрос в соединении, если это возможно. Серверам не следует разрывать соединение в середине передачи ответа, если не предполагается сетевого или клиентского отказа.

Общие требования:

– **HTTP/1.1 серверам следует поддерживать постоянные соединения и использовать механизмы управления потоком данных TCP в целях уменьшения временных перегрузок, вместо закрытия соединений, которые, как ожидается, могут быть повторно использованы клиентами. Последняя методика может усиливать сетевую загрузку.**

– **HTTP/1.1 (или более поздним) клиентам, посылающим тело сообщения (message-body) следует контролировать сетевое соединение на предмет ошибок во время передачи запроса. Если клиент обнаруживает ошибку, ему следует немедленно прекратить передачу тела сообщения. Если тело посылается с использованием кодирования "по кускам" ("chunked"), то кусок нулевой длины и пустой завершитель могут использоваться для индикации преждевременного конца сообщения. Если телу предшествовал заголовок Content-Length, клиент должен закрыть соединение.**

– **HTTP/1.1 (или более поздний) клиент должен быть готов принять ответ с кодом состояния 100 (Продолжать, Continue), предшествующий основному ответу.**

– **HTTP/1.1 (или более поздний) сервер, который получает запрос от HTTP/1.0 (или более раннего) клиента не должен отвечать кодом состояния 100 (Продолжать, Continue); ему следует либо ждать пока запрос будет выполнен обычным образом (то есть без использования прерванного запроса), либо преждевременно закрыть соединение.**

После получения метода, подчиненного этим требованиям, от HTTP/1.1 (или более позднего) клиента, HTTP/1.1 (или более поздний) сервер должен либо ответить кодом состояния 100 (Продолжать, Continue) и продолжать чтение входного потока, либо ответить кодом состояния ошибки. Если сервер ответил кодом состояния ошибки, то он может либо закрыть транспортное соединение (TCP), либо продолжать читать и отбрасывать оставшуюся часть запроса. Он не должен выполнять запрошенный метод, если возвратил код состояния ошибки.

Клиентам следует помнить номер версии HTTP, используемой сервером, по крайней мере, в последний раз; если HTTP/1.1 клиент встречал HTTP/1.1 или более поздний ответ от сервера и видит закрытие соединения перед получением какого-либо кода состояния от сервера, клиенту следует повторить запрос без взаимодействия с пользователем, если метод запроса идемпотентен; другие методы не должны быть повторены автоматически, хотя агенты пользователя могут предложить оператору повторить запрос. Если клиент повторяет запрос, то он должен сначала послать поля заголовка запроса, а затем должен либо ожидать ответа сервера с кодом 100 (Продолжать, Continue) и затем продолжать, либо с кодом состояния ошибки.

Если HTTP/1.1 клиент не встречал ответа сервера версии HTTP/1.1 или более поздней, то ему следует считать, что сервер реализует HTTP/1.0 или более старый протокол и не использовать ответы с кодом состояния 100 (Продолжать, Continue). Если в такой ситуации клиент видит закрытие соединения перед получением какого-либо ответа с кодом состояния от сервера, то ему следует повторить запрос. Если клиент повторяет запрос к этому HTTP/1.0 серверу, то он должен использовать следующий алгоритм "двоичной экспоненциальной задержки" ("binary exponential backoff"), чтобы гарантировать получение надежного ответа:

1) **инициировать новое соединение с сервером;**
2) **передать заголовки запроса (request-headers);**
3) **инициализировать переменную *R* примерным временем передачи информации на сервер и обратно (например на основании времени установления соединения), или постоянным значение в пять секунд, если время передачи не доступно;**
4) **вычислить *T = R * (2**N)*, где *N* – число предыдущих повторов этого запроса;**
5) **либо дождаться от сервера ответа с кодом ошибки, либо просто выждать *T* секунд (смотря что произойдет раньше);**
6) **если ответа с кодом ошибки не получено, после *T* секунд передать тело запроса;**
7) **если клиент обнаруживает, что соединение было закрыто преждевременно, то ему нужно повторять начиная с шага 1, пока запрос не будет принят, либо пока не будет получен ошибочный ответ, либо пока у пользователя не кончится терпение и он не завершит процесс повторения.**

Независимо от того, какая версия HTTP реализована сервером, если клиент получает код состояния ошибки, то он не должен продолжать и должен закрыть соединение, если он не завершил посылку сообщения.

HTTP/1.1 (или более позднему) клиенту, который обнаруживает закрытие соединения после получения ответа с кодом состояния 100 (Продолжать, Continue), но до получения ответа с другим

кодом состояния, следует повторить запрос, но уже не ожидать ответа с кодом состояния 100 (Продолжать, Continue).

9.4.9. ОПРЕДЕЛЕНИЯ МЕТОДОВ

Множество общих для HTTP/1.1 методов (OPTIONS, GET, PUT и т.д.) приводится ниже. Хотя это множество может быть расширено, нельзя считать, что дополнительные методы имеют одинаковую семантику, если они являются расширениями разных клиентов и серверов.

Поле заголовка запроса Host должно сопровождать все HTTP/1.1 запросы.

9.4.9.1. Безопасные и идемпотентные методы

Программистам следует понимать, что программное обеспечение при взаимодействии с Интернетом представляет пользователя, и программе следует информировать пользователя о любых действиях, которые он может произвести, но которые могут иметь непредсказуемое значение для него или других лиц.

В частности, было принято соглашение, что методы GET и HEAD никогда не должны иметь иного значения, кроме загрузки (retrieval). Указанные методы следует рассматривать как "безопасные". Это позволяет агенту пользователя представлять другие методы, такие как POST, PUT и DELETE, таким образом, чтобы пользователь был проинформирован о том, что он запрашивает выполнение потенциально опасного действия.

Естественно невозможно гарантировать, что сервер не генерирует побочных эффектов в результате выполнения запроса GET; фактически, некоторые динамические ресурсы содержат такую возможность. Важное различие здесь в том, что не пользователь запрашивает побочные эффекты, и, следовательно, пользователь не может нести ответственность за них.

Методы могут также обладать свойством "идемпотентности" ("idempotence") в том смысле, что побочные эффекты от $N > 0$ идентичных запросов такие же, как от одиночного запроса (за исключением ошибок и проблем устаревания). Методы GET, HEAD, PUT и DELETE обладают данным свойством.

9.4.9.2. OPTIONS

Метод OPTIONS представляет запрос информации об опциях соединения, доступных в цепочке запросов/ответов, идентифицируемой запрашиваемым URI (Request-URI). Этот метод позволяет клиенту определять опции и/или требования, связанные с ресурсом, или возможностями сервера, но не производя никаких действий над ресурсом и не инициируя его загрузку.

Если ответ сервера – это не сообщение об ошибке, то ответ не должен содержать иной информации объекта, кроме той, которую можно рассматривать как опции соединения (например Allow – можно рассматривать как опцию соединения, а Content-Type – нет). Ответы на этот метод не кэшируются.

Если запрашиваемый URI (Request-URI) – звездочка ("*"), то запрос OPTIONS предназначен для обращения к серверу в целом. Если код состояния ответа – 200, то ответу следует содержать любые поля заголовка, которые указывают опциональные возможности, реализуемые сервером (например, Public), включая любые расширения, не определенные данной спецификацией, в дополнение к соответствующим общим полям или полям заголовка ответа (response-header). Как описано в разделе 5.1.2, запрос "OPTIONS *" может быть применен через прокси-сервер с определением адресуемого сервера в запрашиваемом URI (Request-URI) с пустым путем.

Если запрашиваемый URI (Request-URI) не звездочка ("*"), то запрос OPTIONS применяется к опциям, которые доступны при соединении с указанным ресурсом. Если код состояния ответа – 200, то ответу следует содержать любые поля заголовка, которые указывают опциональные возможности, реализуемые сервером и применимые к указанному ресурсу (например Allow), включая любые расширения, не определенные данной спецификацией, в дополнение к соответствующим общим полям или полям заголовка ответа (response-header). Если запрос OPTIONS передается через прокси-сервер, то последний редактирует ответ, исключая те опции, которые не предусмотрены возможностями этого прокси-сервера.

9.4.9.3. GET

Метод GET позволяет получать любую информацию (в форме объекта), идентифицированную запрашиваемым URI (Request-URI). Если запрашиваемый URI (Request-URI) обращается к процессу, производящему данные, то в качестве объекта ответа должны быть возвращены произведенные данные, а не исходный текст процесса, если сам процесс не выводит исходный текст.

Различается "условный GET" ("conditional GET"), при котором сообщение запроса включает поля заголовка If-Modified-Since, If-Unmodified-Since, If-Match, If-None-Match, или If-Range. Условный метод GET запрашивает передачу объекта, только если последний удовлетворяет условиям, описанным в условных полях заголовка. Условный метод GET предназначен для уменьшения неоправданной загрузки сети и позволяет обновлять кэшированные объекты без использования нескольких запросов или пересылки данных, уже сохраненных клиентом.

Различается также "частичный GET" ("partial GET"), при котором сообщение запроса включает поле заголовка Range. Частичный GET запрашивает передачу только части объекта. Частичный метод GET предназначен для уменьшения ненужной загрузки сети, и позволяет собирать объекты из частей, без передачи частей данных, уже сохраненных клиентом.

Ответ на запрос GET кэшируем тогда и только тогда, когда он отвечает требованиям кэширования в HTTP, описанным ниже.

9.4.9.4. HEAD

Метод HEAD идентичен GET, за исключением того, что сервер не должен возвращать в ответе тело сообщения (message-body). Метаинформации, содержащейся в HTTP заголовках ответа на запрос HEAD следует быть идентичной информации, представляемой в ответ на запрос GET. Этот метод может использоваться для получения метаинформации об объекте запроса без непосредственной пересылки тела объекта (entity-body) и часто используется для тестирования гипертекстовых связей в целях проверки достоверности, достижимости и времени модификации.

Ответ на запрос HEAD может быть кэшируемым в том смысле, что информация, содержащаяся в ответе, может использоваться для модифицикации предварительно кэшированного объекта из этого ресурса. Если новые значения поля указывают, что кэшируемый объект отличается от текущего объекта (по таким параметрам, как Content-Length, Content-MD5, ETag или Last-Modified), то кэш должен обрабатывать содержимое как просроченное.

9.4.9.5. POST

Метод POST используется для запроса, при котором адресуемый сервер принимает объект, включенный в запрос, как новое подчинение (subordinate) ресурса, идентифицированного запрашиваемым URI (Request-URI) в строке запроса (Request-Line). POST разработан для того, чтобы общим методом реализовать следующие функции:

- **аннотация существующих ресурсов;**
- **регистрация сообщения на электронной доске объявлений (bulletin board), в конференции новостей (newsgroup), списке рассылки (mailing list) или подобной группе статей;**
- **передача блока данных, например результат ввода в форме, процессу обработки;**
- **расширение базы данных посредством конкатенирующей операции (append operation).**

Фактически функция, выполняемая методом POST, определяется сервером и обычно зависит от запрашиваемого URI (Request-URI). Объект, передаваемый методом POST, относится к этому URI так же, как файл относится к каталогу, в котором он находится, статья относится к конференции новостей (newsgroup), в которой она зарегистрирована, а запись относится к базе данных.

Действие, выполняемое методом POST, может не давать в качестве результата ресурс, который можно было бы идентифицировать URI.

В этом случае, в зависимости от того, включает ли ответ объект, описывающий результат, или нет, код состояния в ответе может быть как 200 (OK), так и 204 (Нет содержимого, No Content).

Если ресурс был создан на первоначальном сервере, ответу следует содержать код состояния 201 (Создан, Created) и включать объект, который описывает состояние запроса и ссылается на новый ресурс, а также заголовок Locatio.

Ответы на этот метод не кэшируемы, если ответ не включает соответствующие поля заголовка Cache-Control или Expires. Однако ответ с кодом состояния 303 (Смотреть другой, See Other) может использоваться для перенаправления агента пользователя для загрузки кэшируемого ресурса.

Запросы POST должны отвечать требованиям передачи сообщения.

9.4.9.6. PUT

Запросы с методом PUT, которые содержат объект, сохраняются под запрашиваемым URI (Request-URI). Если Request-URI обращается к уже существующему ресурсу, включенный объект следует рассматривать как модифицированную версию объекта, находящегося на первоначальном сервере. Если Request-URI не указывает на существующий ресурс, и может интерпретироваться агентом пользователя как новый ресурс для запросов, первоначальный сервер может создать ресурс с данным URI. Если новый ресурс создан, то первоначальный сервер должен сообщить агенту пользователя об этом посредством ответа с кодом состояния 201 (Создан, Created). Если существующий ресурс модифицирован, то для указания успешного завершения запроса следует послать ответ с кодом состояния либо 200 (OK), либо 204 (Нет содержимого, No Content). Если ресурс не может быть создан или изменен для запрашиваемого URI (Request-URI), то следует послать ответ, отражающий характер проблемы. Получатель объекта не должен игнорировать заголовков Content-* (например Content-Range), которых не понимает или не реализует, а должен в данном случае возвратить ответ с кодом состояния 501 (Не реализовано, Not Implemented).

Если запрос передается через кэш и запрашиваемый URI (Request-URI) идентифицирует один или несколько кэшированных в настоящее время объектов, то вхождения в кэш этих объектов должны обрабатываться как просроченные. Ответы на этот метод не кэшируемы.

Фундаментальное различие между запросами POST и PUT отражено в различном значении запрашиваемого URI (Request-URI). URI в запросе POST идентифицирует ресурс, который обрабатывает включенный объект. Этим ресурсом может быть процесс, принимающий данные, шлюз к некоторому другому протоколу, или отдельный объект, который принимает аннотации (accepts annotations). Напротив, URI в запросе PUT идентифицирует объект включенный в запрос – агент пользователя назначает данный URI включенному ресурсу, а сервер не должен пытаться применить запрос к некоторому другому ресурсу. Если сервер желает применить запрос к другому URI, он должен послать ответ с кодом 301 (Перемещен постоянно, Moved Permanently); агент пользователя может затем принять собственное решение относительно переназначения запроса.

Один ресурс может быть идентифицирован несколькими различными URI. Например статья может иметь URI идентифицирующий "текущую версию", который отличен от URI, идентифицирующего каждую специфическую версию. В этом случае, запрос PUT на общий URI может отразиться на некоторых других URI, определенных первоначальным сервером.

HTTP/1.1 не определяет каким образом метод PUT воздействует на состояние первоначального сервера.

Запросы PUT должны подчиняться требованиям передачи сообщений.

9.4.9.7. DELETE

Метод DELETE запрашивает первоначальный сервер об удалении ресурса, идентифицируемого запрашиваемым URI (Request-URI). Этот метод может быть отменен человеческим вмешательством (или другими средствами) на первоначальном сервере. Клиенту нельзя гарантировать, что операция была выполнена, даже если код состояния, возвращенный первоначальным сервером указывает на то, что действие было завершено успешно. Однако, серверу не следует отвечать об успешном выполнении, если во время формирования ответа он только собирается удалить ресурс или переместить его в недоступное положение.

Успешному ответу следует иметь код состояния 200 (OK), если он включает объект, описывающий состояние; иметь код состояния 202 (Принято, Accepted), если действие еще не было про-

изведено; либо иметь код состояния 204 (Нет содержимого, No Content), если ответ сообщает об успехе (OK), но не содержит объекта.

Если запрос передается через кэш и запрашиваемый URI (Request-URI) идентифицирует один или несколько кэшированных в настоящее время объектов, то вхождения их должны обрабатываться как просроченные. Ответы на этот метод не кэшируемы.

9.4.9.8. TRACE

Метод TRACE используется для вызова удаленного возврата сообщения запроса на уровне приложения. Конечному получателю запроса следует отразить полученное сообщение обратно клиенту как тело объекта ответа с кодом состояния 200 (OK). Конечным получателем является либо первоначальный сервер, либо первый прокси-сервер/шлюз, получивший нулевое значение (0) в поле Max-Forwards в запросе. Запрос TRACE не должен содержать объекта.

TRACE позволяет клиенту увидеть, что получается на другом конце цепочки запросов и использовать эти данные для тестирования или диагностической информации. Значение поля заголовка Via представляет особый интерес, так как оно действует как след цепочки запросов. Использование поля заголовка Max-Forwards позволяет клиенту ограничивать длину цепочки запросов, что полезно при тестировании бесконечных циклов в цепочке прокси-серверов, пересылающих сообщения.

Если запрос выполнен успешно, то ответу следует содержать все сообщение запроса в теле объекта (entity-body), а Content-Type следует быть равным "message/http". Ответы на этот метод не должны кэшироваться.

10. ТРЕБОВАНИЯ К КОМПЬЮТЕРНЫМ СЕТЯМ

Соответствие стандартам – это только одно из многих требований, предъявляемых к современным сетям. В этом разделе мы остановимся на некоторых других, не менее важных.

Самое общее пожелание, которое можно высказать в отношении работы сети – это выполнение сетью того набора услуг, для оказания которых она предназначена: например, предоставление доступа к файловым архивам или страницам публичных Web-сайтов Internet, обмен электронной почтой в пределах предприятия или в глобальных масштабах, интерактивный обмен голосовыми сообщениями IP-телефонии и т.п.

Все остальные требования – *производительность*, надежность, совместимость, управляемость, защищенность, *расширяемость* и *масштабируемость* – связаны с качеством выполнения этой основной задачи. И хотя все перечисленные выше требования весьма важны, часто понятие "качество обслужива-

ния" (Quality of Service, QoS) компьютерной сети трактуется более узко: в него включаются только две самые важные характеристики сети – производительность и надежность.

10.1. Производительность

Потенциально высокая производительность – это одно из основных преимуществ распределенных систем, к которым относятся компьютерные сети. Это свойство обеспечивается принципиальной, но, к сожалению, не всегда практически реализуемой возможностью распределения работ между несколькими компьютерами сети.

Основные характеристики производительности сети:

- время реакции;
- скорость передачи трафика;
- пропускная способность;
- задержка передачи и вариация задержки передачи.

Время реакции сети является интегральной характеристикой производительности сети с точки зрения пользователя. Именно эту характеристику имеет в виду пользователь, когда говорит: "Сегодня сеть работает медленно".

В общем случае время реакции определяется как интервал между возникновением запроса пользователя к какой-либо сетевой службе и получением ответа на него.

Очевидно, что значение этого показателя зависит от типа службы, к которой обращается пользователь, от того, какой пользователь и к какому серверу обращается, а также от текущего состояния элементов сети – загруженности сегментов, коммутаторов и маршрутизаторов, через которые проходит запрос, загруженности сервера и т.п.

Поэтому имеет смысл использовать также и средневзвешенную оценку времени реакции сети, усредняя этот показатель по пользователям, серверам и времени дня (от которого в значительной степени зависит загрузка сети).

Время реакции сети обычно складывается из нескольких составляющих. В общем случае в него входит:

– время подготовки запросов на клиентском компьютере;

– время передачи запросов между клиентом и сервером через сегменты сети и промежуточное коммуникационное оборудование;

– время обработки запросов на сервере;

– время передачи ответов от сервера клиенту и время обработки получаемых от сервера ответов на клиентском компьютере.

Очевидно, что разложение времени реакции на составляющие пользователя не интересует – ему важен конечный результат. Однако для сетевого специалиста очень важно выделить из общего времени реакции составляющие, соответствующие этапам собственно сетевой обработки данных, – передачу данных от клиента к серверу через сегменты сети и коммуникационное оборудование.

Знание сетевых составляющих времени реакции позволяет оценить производительность отдельных элементов сети, выявить узкие места и при необходимости выполнить модернизацию сети для повышения ее общей производительности.

Производительность сети может характеризоваться также скоростью передачи трафика.

Скорость передачи трафика может быть мгновенной, максимальной и средней.

– мгновенная скорость отличается от средней тем, что для усреднения выбирается очень маленький промежуток времени – например, 10 мс или 1 с;

– максимальная скорость – это наибольшая скорость, зафиксированная в течение периода наблюдения;

– средняя скорость вычисляется путем деления общего объема переданных данных на время их передачи, причем выбирается достаточно длительный промежуток времени – час, день или неделя.

Чаще всего при проектировании, настройке и оптимизации сети используются такие показатели, как средняя и максимальная скорость. Средняя скорость, с которой обрабатывает трафик, отдельный элемент или сеть в целом, позволяет оценить работу сети на протяжении длительного времени, в течение которого в силу закона больших чисел пики и спады интенсивности трафика компенсируют друг друга. Максимальная скорость позволяет оценить, как сеть будет справляться с пиковыми нагрузками, характерными для особых периодов работы, например в утренние часы, когда сотрудники предприятия почти одновременно регистрируются в сети и обращаются к разделяемым файлам и базам данных. Обычно при определении скоростных характеристик некоторого сегмента или устройства в передавае-

мых данных не выделяется трафик какого-то определенного пользователя, приложения или компьютера – подсчитывается общий объем передаваемой информации. Тем не менее, для более точной оценки качества обслуживания такая детализации желательна, и в последнее время системы управления сетями все чаще позволяют ее выполнять.

Пропускная способность – максимально возможная скорость обработки трафика, определенная стандартом технологии, на которой построена сеть. Пропускная способность отражает максимально возможный объем данных, передаваемый сетью или ее частью в единицу времени.

Пропускная способность уже не является, подобно времени реакции или скорости прохождения данных по сети, пользовательской характеристикой, так как она говорит о скорости выполнения внутренних операций сети – передачи пакетов данных между узлами сети через различные коммуникационные устройства. Зато она непосредственно характеризует качество выполнения основной функции сети – транспортировки сообщений – и поэтому чаще используется при анализе производительности сети, чем время реакции или скорость.

Пропускная способность измеряется либо в битах в секунду, либо в пакетах в секунду.

Пропускная способность сети зависит как от характеристик физической среды передачи (медный кабель, оптическое волокно, витая пара) так и от принятого способа передачи данных (технология Ethernet, FastEthernet, ATM). Пропускная способность часто используется в качестве характеристики не столько сети, сколько собственно технологии, на которой построена сеть. Важность этой характеристики для сетевой технологии показывает, в частности, и то, что ее значение иногда становится частью названия, например, 10 Мбит/с Ethernet, 100 Мбит/с Ethernet.

В отличие от времени реакции или скорости передачи трафика пропускная способность не зависит от загруженности сети и имеет постоянное значение, определяемое используемыми в сети технологиями.

На разных участках гетерогенной сети, где используется несколько разных технологий, пропускная способность может быть различной. Для анализа и настройки сети очень полезно знать данные о пропускной способности отдельных ее элементов. Важно отметить, что из-за последовательного характера передачи данных различными элементами сети *общая пропускная способность* любого составного пути в сети будет равна минимальной из пропускных способностей составляющих элементов маршрута. Для повышения пропускной способности составного пути необходимо в первую очередь обратить внимание на самые медленные элементы. Иногда полезно оперировать общей пропускной способностью сети, которая определяется как среднее количество информации, переданной между всеми узлами сети за единицу времени. Этот показатель характеризует качество сети в целом, не дифференцируя его по отдельным сегментам или устройствам.

Задержка передачи определяется как задержка между моментом поступления данных на вход какого-либо сетевого устройства или части сети и моментом появления их на выходе этого устройства.

Этот параметр производительности по смыслу близок ко времени реакции сети, но отличается тем, что всегда характеризует только сетевые этапы обработки данных, без задержек обработки конечными узлами сети.

Обычно качество сети характеризуют величинами максимальной задержки передачи и вариацией задержки. Не все типы трафика чувствительны к задержкам передачи, во всяком случае, к тем величинам задержек, которые характерны для компьютерных сетей, – обычно задержки не превышают сотен миллисекунд, реже – нескольких секунд. Такого порядка задержки пакетов, порождаемых файловой службой, службой электронной почты или службой печати, мало влияют на качество этих служб с точки зрения пользователя сети. С другой стороны, такие же задержки пакетов, переносящих голосовые или видеоданные, могут приводить к значительному снижению качества предоставляемой пользователю информации – возникновению эффекта "эха", невозможности разобрать некоторые слова, вибрации изображения и т.п.

Все указанные характеристики производительности сети достаточно независимы. В то время как пропускная способность сети является постоянной величиной, скорость передачи трафика может варьироваться в зависимости от загрузки сети, не превышая, конечно, предела, устанавливаемого пропускной способностью. Так в односегментной сети 10 Мбит/с Ethernet компьютеры могут обмениваться данными со скоростями 2 Мбит/с и 4 Мбит/с, но никогда – 12 Мбит/с.

Пропускная способность и задержки передачи также являются независимыми параметрами, так что сеть может обладать, например, высокой пропускной способностью, но вносить значительные задержки при передаче каждого пакета. Пример такой ситуации дает канал связи, образованный геостационарным спутником. Пропускная способность этого канала может быть весьма высокой, например 2 Мбит/с, в то

время как задержка передачи всегда составляет не менее 0,24 с, что определяется скоростью распространения электрического сигнала (около 300 000 км/с) и длиной канала (72 000 км).

10.2. Надежность и безопасность

Одна из первоначальных целей создания распределенных систем, к которым относятся и вычислительные сети, состояла в достижении большей надежности по сравнению с отдельными вычислительными машинами.

Важно различать несколько аспектов надежности.

Для сравнительно простых технических устройств используются такие *показатели надежности*, как:

- среднее время наработки на отказ;
- вероятность отказа;
- интенсивность отказов.

Однако эти показатели пригодны для оценки надежности простых элементов и устройств, которые могут находиться только в двух состояниях – работоспособном или неработоспособном. Сложные системы, состоящие из многих элементов, кроме состояний работоспособности и неработоспособности, могут иметь и другие промежуточные состояния, которые эти характеристики не учитывают.

Для оценки надежности сложных систем применяется другой набор характеристик:

- готовность или коэффициент готовности;
- сохранность данных;
- согласованность (непротиворечивость) данных;
- вероятность доставки данных;
- безопасность;
- отказоустойчивость.

Готовность или коэффициент готовности (availability) означает период времени, в течение которого система может использоваться. Готовность может быть повышена путем введения избыточности в структуру системы: ключевые элементы системы должны существовать в нескольких экземплярах, чтобы при отказе одного из них функционирование системы обеспечивали другие.

Чтобы компьютерную систему можно было считать высоконадежной, она должна как минимум обладать высокой готовностью, но этого недостаточно. Необходимо обеспечить сохранность данных и защиту их от искажений. Кроме того, должна поддерживаться согласованность (непротиворечивость) данных, например если для повышения надежности на нескольких файловых серверах хранится несколько копий данных, то нужно постоянно обеспечивать их идентичность.

Так как сеть работает на основе механизма передачи пакетов между конечными узлами, одной из характеристик надежности является вероятность доставки пакета узлу назначения без искажений. Наряду с этой характеристикой могут использоваться и другие показатели: вероятность потери пакета (по любой из причин – из-за переполнения буфера маршрутизатора, несовпадения контрольной суммы, отсутствия работоспособного пути к узлу назначения и т.д.), вероятность искажения отдельного бита передаваемых данных, соотношение количества потерянных и доставленных пакетов.

Другим аспектом общей надежности является безопасность (security), то есть способность системы защитить данные от несанкционированного доступа. В распределенной системе это сделать гораздо сложнее, чем в централизованной. В сетях сообщения передаются по линиям связи, часто проходящим через общедоступные помещения, в которых могут быть установлены средства прослушивания линий. Другим уязвимым местом могут стать оставленные без присмотра персональные компьютеры. Кроме того, всегда имеется потенциальная угроза взлома защиты сети от неавторизованных пользователей, если сеть имеет выходы в глобальные общедоступные сети.

Еще одной характеристикой надежности является *отказоустойчивость* (fault tolerance). В сетях под отказоустойчивостью понимается способность системы скрыть от пользователя отказ отдельных ее элементов. Например, если копии таблицы базы данных хранятся одновременно на нескольких файловых серверах, пользователи могут просто не заметить отказа одного из них. В отказоустойчивой системе выход из строя одного из ее элементов приводит к некоторому снижению качества ее работы (деградации), а не к полному останову. Так, при отказе одного из файловых серверов в предыдущем примере увеличивается только время доступа к базе данных из-за уменьшения степени распараллеливания запросов, но в целом система будет продолжать выполнять свои функции.

10.3. Расширяемость и масштабируемость

Термины "расширяемость" и "масштабируемость" иногда используют как синонимы, но это неверно – каждый из них имеет четко определенное самостоятельное значение.

Расширяемость (extensibility) означает возможность сравнительно легкого добавления отдельных элементов сети (пользователей, компьютеров, приложений, служб), наращивания длины сегментов сети и замены существующей аппаратуры более мощной. При этом принципиально важно, что легкость расширения системы иногда может обеспечиваться в весьма ограниченных пределах. Например, локальная сеть Ethernet, построенная на основе одного сегмента толстого коаксиального кабеля, обладает хорошей расширяемостью, в том смысле, что позволяет без труда подключать новые станции. Однако такая сеть имеет ограничение на число станций – оно не должно превышать 30–40. Хотя сеть допускает физическое подключение к сегменту и большего числа станций (до 100), но при этом чаще всего резко снижается производительность сети. Наличие такого ограничения и является признаком плохой масштабируемости системы при хорошей расширяемости.

Масштабируемость (scalability) означает, что сеть позволяет наращивать количество узлов и протяженность связей в очень широких пределах, при этом производительность сети не ухудшается. Для обеспечения масштабируемости сети приходится применять дополнительное коммуникационное оборудование и специальным образом структурировать сеть. Например, хорошей масштабируемостью обладает многосегментная сеть, построенная с использованием коммутаторов и маршрутизаторов и имеющая иерархическую структуру связей. Такая сеть может включать несколько тысяч компьютеров и при этом обеспечивать каждому пользователю сети нужное качество обслуживания.

Прозрачность (transparency) сети достигается в том случае, когда сеть представляется пользователям не как множество отдельных компьютеров, связанных между собой сложной системой кабелей, а как единая традиционная вычислительная машина с системой разделения времени. Известный лозунг компании Sun Microsystems "Сеть – это компьютер" – говорит именно о такой прозрачной сети.

Прозрачность может быть достигнута на двух различных уровнях – на уровне пользователя и на уровне программиста. На уровне пользователя прозрачность означает, что для работы с удаленными ресурсами он использует те же команды и привычные процедуры, что и для работы с локальными ресурсами. На программном уровне прозрачность заключается в том, что приложению для доступа к удаленным ресурсам требуются те же вызовы, что и для доступа к локальным ресурсам. Прозрачности на уровне пользователя достичь проще, так как все особенности процедур, связанные с распределенным характером системы, скрываются от пользователя программистом, который создает приложение. Прозрачность на уровне приложения требует сокрытия всех деталей распределенности средствами сетевой операционной системы.

Прозрачность – свойство сети скрывать от пользователя детали своего внутреннего устройства, что упрощает работу в сети.

Сеть должна скрывать все особенности операционных систем и различия в типах компьютеров. Пользователь компьютера Macintosh должен иметь возможность обращаться к ресурсам, поддерживаемым UNIX-системой, а пользователь UNIX – разделять информацию с пользователями Windows 95. Подавляющее большинство пользователей ничего не хочет знать о внутренних форматах файлов или о синтаксисе команд UNIX. Пользователь терминала IBM 3270 должен иметь возможность обмениваться сообщениями с пользователями сети персональных компьютеров без необходимости вникать в секреты трудно запоминаемых адресов.

Концепция прозрачности применима к различным аспектам сети. Например, прозрачность расположения означает, что от пользователя не требуется знать местонахождение программных и аппаратных ресурсов, таких как процессоры, принтеры, файлы и базы данных. Имя ресурса не должно включать информацию о месте его расположения, поэтому имена типа mashine1:prog.c или \\ftp_serv\pub прозрачными не являются. Аналогично, прозрачность перемещения означает, что ресурсы могут свободно перемещаться из одного компьютера в другой без изменения имен. Еще одним из возможных аспектов прозрачности является прозрачность параллелизма, которая заключается в том, что процесс распараллеливания вычислений происходит автоматически, без участия программиста, при этом система сама распределяет параллельные ветви приложения по процессорам и компьютерам сети. В настоящее время нельзя сказать, что свойство прозрачности в полной мере присуще многим вычислительным сетям, это скорее цель, к которой стремятся разработчики современных сетей.

10.4. Поддержка разных видов трафика

Компьютерные сети изначально предназначались для совместного доступа к ресурсам компьютеров: файлам, принтерам и т.п. Трафик, создаваемый этими традиционными службами компьютерных сетей, имеет свои особенности и существенно отличается от трафика сообщений в телефонных сетях или, например, в сетях кабельного телевидения. Однако в 90-е годы в компьютерные сети проник трафик мультимедийных данных, представляющих в цифровой форме речь и видеоизображение. Компьютерные сети стали использоваться для организации видеоконференций, обучения на основе видеофильмов и т.п. Естественно, что для динамической передачи *мультимедийного трафика* требуются иные алгоритмы и протоколы, и, соответственно, другое оборудование. Хотя доля мультимедийного трафика пока невелика, он уже начал проникать как в глобальные, так и в локальные сети, и этот процесс, очевидно, будет активно продолжаться.

Главной особенностью трафика, образующегося при динамической передаче голоса или изображения, является наличие жестких требований к *синхронности* передаваемых сообщений. Для качественного воспроизведения непрерывных процессов, которыми являются звуковые колебания или изменения интенсивности света в видеоизображении, необходимо получение измеренных и закодированных амплитуд сигналов с той же частотой, с которой они были измерены на передающей стороне. При запаздывании сообщений будут наблюдаться искажения.

В то же время трафик компьютерных данных характеризуется крайне неравномерной интенсивностью поступления сообщений в сеть при отсутствии жестких требований к синхронности доставки этих сообщений. Например, доступ пользователя, работающего с текстом на удаленном диске, порождает случайный поток сообщений между удаленным и локальным компьютерами, зависящий от действий пользователя, причем задержки при доставке в некоторых (достаточно широких с компьютерной точки зрения) пределах мало влияют на качество обслуживания пользователя сети. Все алгоритмы компьютерной связи, соответствующие протоколы и коммуникационное оборудование были рассчитаны именно на такой "пульсирующий" характер трафика, поэтому необходимость передавать мультимедийный трафик требует внесения принципиальных изменений, как в протоколы, так и в оборудование. Сегодня практически все новые протоколы в той или иной степени предоставляют поддержку мультимедийного трафика.

Особую сложность представляет совмещение в одной сети традиционного компьютерного и мультимедийного трафика. Передача исключительно мультимедийного *трафика компьютерной* сетью хотя и связана с определенными сложностями, но доставляет меньше хлопот. А вот сосуществование двух типов трафика с противоположными требованиями к качеству обслуживания является намного более сложной задачей. Обычно протоколы и оборудование компьютерных сетей относят мультимедийный трафик к факультативному, поэтому качество его обслуживания оставляет желать лучшего. Сегодня затрачиваются большие усилия по созданию сетей, которые не ущемляют интересы одного из типов трафика. Наиболее близки к этой цели сети на основе технологии ATM, разработчики которой изначально учитывали случай сосуществования разных типов трафика в одной сети.

10.5. Управляемость

В идеале средства управления сетями представляют собой систему, осуществляющую наблюдение, контроль и управление каждым элементом сети – от простейших до самых сложных устройств, при этом такая система рассматривает сеть как единое целое, а не как разрозненный набор отдельных устройств.

Управляемость сети подразумевает возможность централизованно контролировать состояние основных элементов сети, выявлять и решать проблемы, возникающие при работе сети, выполнять анализ производительности и планировать развитие сети.

Хорошая система управления наблюдает за сетью и, обнаружив проблему, активизирует определенное действие, исправляет ситуацию и уведомляет администратора о том, что произошло и какие шаги предприняты. Одновременно с этим система управления должна накапливать данные, на основании которых можно планировать развитие сети. Наконец, система управления должна быть независимой от производителя и обладать удобным интерфейсом, позволяющим выполнять все действия с одной консоли.

Решая тактические задачи, администраторы и технический персонал сталкиваются с ежедневными проблемами обеспечения работоспособности сети. Эти задачи требуют быстрого решения, обслуживающий сеть персонал должен оперативно реагировать на сообщения о неисправностях, поступающих от пользователей или автоматических средств управления сетью. Постепенно становятся заметны общие проблемы производительности, конфигурирования сети, обработки сбоев и безопасности данных,

требующие стратегического подхода, то есть *планирования сети*. Планирование, кроме этого, включает прогноз изменений требований пользователей к сети, вопросы применения новых приложений, новых сетевых технологий и т.п.

Необходимость в системе управления особенно ярко проявляется в больших сетях: корпоративных или глобальных. Без системы управления в таких сетях требуется присутствие квалифицированных специалистов по эксплуатации в каждом здании каждого города, где установлено оборудование сети, что в итоге приводит к необходимости содержания огромного штата обслуживающего персонала.

В настоящее время в области систем управления сетями много нерешенных проблем. Явно недостаточно действительно удобных, компактных и многопротокольных средств управления сетью. Большинство существующих средств вовсе не управляют сетью, а всего лишь осуществляют наблюдение за ее работой. Они следят за сетью, но не выполняют активных действий, если с сетью что-то произошло или может произойти. Мало масштабируемых систем, способных обслуживать как сети масштаба отдела, так и сети масштаба предприятия, – очень многие системы управляют только отдельными элементами сети и не анализируют способность сети выполнять качественную передачу данных между конечными пользователями.

10.6. Совместимость

Совместимость или интегрируемость означает, что сеть может включать в себя разнообразное программное и аппаратное обеспечение, то есть в ней могут сосуществовать различные операционные системы, поддерживающие разные стеки коммуникационных протоколов, и работать аппаратные средства и приложения от разных производителей. Сеть, состоящая из разнотипных элементов, называется неоднородной или гетерогенной, а если гетерогенная сеть работает без проблем, то она является интегрированной. Основной путь построения интегрированных сетей – использование модулей, выполненных в соответствии с открытыми стандартами и спецификациями.

Качество обслуживания (Quality of Service, QoS) определяет количественные оценки вероятности того, что сеть будет передавать определенный поток данных между двумя узлами в соответствии с потребностями приложения или пользователя.

Например, при передаче голосового трафика через сеть под качеством обслуживания чаще всего понимают гарантии того, что голосовые пакеты будут доставляться сетью с задержкой не более N мс, при этом вариация задержки не превысит M мс, и эти характеристики станут выдерживаться сетью с вероятностью 0,95 на определенном временном интервале. То есть приложению, которое передает голосовой трафик, важно, чтобы сеть гарантировала соблюдение именно этого приведенного выше набора характеристик качества обслуживания. Файловому сервису нужны гарантии средней полосы пропускания и расширения ее на небольших интервалах времени до некоторого максимального уровня для быстрой передачи пульсаций. В идеале сеть должна гарантировать особые параметры качества обслуживания, сформулированные для каждого отдельного приложения. Однако по понятным причинам разрабатываемые и уже существующие механизмы QoS ограничиваются решением более простой задачи – гарантированием неких усредненных требований, заданных для основных типов приложений.

Чаще всего параметры, фигурирующие в разнообразных определениях качества обслуживания, регламентируют следующие показатели работы сети:

- пропускную способность;
- задержки передачи пакетов;
- уровень потерь и искажений пакетов.

Качество обслуживания гарантируется для некоторого потока данных. Напомним, что поток данных – это последовательность пакетов, имеющих некоторые общие признаки, например адрес узла-источника, информация, идентифицирующая тип приложения (номер порта TCP/UDP) и т.п. К потокам применимы такие понятия, как агрегирование и дифференцирование. Так, поток данных от одного компьютера может быть представлен как совокупность потоков от разных приложений, а потоки от компьютеров одного предприятия агрегированы в один поток данных абонента некоторого провайдера услуг.

Механизмы поддержки качества обслуживания сами по себе не создают пропускной способности. Сеть не может дать больше того, что имеет. Так что фактическая пропускная способность каналов связи и транзитного коммуникационного оборудования – это ресурсы сети, являющиеся отправной точкой для работы механизмов QoS. Механизмы QoS только управляют распределением имеющейся пропускной способности в соответствии с требованиями приложений и настройками сети. Самый очевидный способ перераспределения пропускной способности сети состоит в управлении очередями пакетов.

Поскольку данные, которыми обмениваются два конечных узла, проходят через некоторое количество промежуточных сетевых устройств, таких как концентраторы, коммутаторы и маршрутизаторы, то поддержка QoS требует взаимодействия всех сетевых элементов на пути трафика, то есть "из-конца-в-конец" ("end-to-end", "e2e"). Любые гарантии QoS настолько соответствуют действительности, насколько их обеспечивает наиболее "слабый" элемент в цепочке между отправителем и получателем. Поэтому нужно четко понимать, что поддержка QoS только в одном сетевом устройстве, пусть даже и магистральном, может лишь весьма незначительно улучшить качество обслуживания или же совсем не повлиять на параметры QoS.

Реализация в компьютерных сетях механизмов поддержки QoS является сравнительно новой тенденцией. Долгое время компьютерные сети существовали без таких механизмов, и это объясняется в основном двумя причинами.

Во-первых, большинство приложений, выполняемых в сети, были "нетребовательными", то есть для таких приложений задержки пакетов или отклонения средней пропускной способности в достаточно широком диапазоне не приводили к значительной потере функциональности. Примерами "нетребовательных" приложений являются наиболее распространенные в сетях 80-х годов приложения электронной почты или удаленного копирования файлов.

Во-вторых, сама пропускная способность 10-мегабитных сетей Ethernet во многих случаях не была дефицитом. Так, разделяемый сегмент Ethernet, к которому было подключено 10–20 компьютеров, изредка копирующих небольшие текстовые файлы, объем которых не превышает несколько сотен килобайт, позволял трафику каждой пары взаимодействующих компьютеров пересекать сеть так быстро, как требовалось породившим этот трафик приложениям.

В результате большинство сетей работало с тем качеством транспортного обслуживания, которое обеспечивало потребности приложений. Правда, никаких гарантий относительно контроля задержек пакетов или пропускной способности, с которой пакеты передаются между узлами, в определенных пределах эти сети не давали. Более того, при временных перегрузках сети, когда значительная часть компьютеров одновременно начинала передавать данные с максимальной скоростью, задержки и пропускная способность становились такими, что работа приложений давала сбой – шла слишком медленно, с разрывами сессий и т.п.

Существует два основных подхода к обеспечению качества работы сети. Первый состоит в том, что сеть гарантирует пользователю соблюдение некоторой числовой величины показателя качества обслуживания. Например, сети frame relay и ATM могут гарантировать пользователю заданный уровень пропускной способности. При втором подходе (best effort) сеть старается по возможности более качественно обслужить пользователя, но ничего при этом не гарантирует.

Транспортный сервис, который предоставляли такие сети, получил название "best effort", то есть сервис "с максимальными усилиями" (или "по возможности"). Сеть старается обработать поступающий трафик как можно быстрее, но при этом никаких гарантий относительно результата не дает. Примерами может служить большинство технологий, разработанных в 80-е годы: Ethernet, Token Ring, IP, X.25. Сервис "с максимальными усилиями" основан на некотором справедливом алгоритме обработки очередей, возникающих при перегрузках сети, когда в течение некоторого времени скорость поступления пакетов в сеть превышает скорость продвижения этих пакетов. В простейшем случае алгоритм обработки очереди рассматривает пакеты всех потоков как равноправные и продвигает их в порядке поступления (First In – First Out, FIFO). В том случае, когда очередь становится слишком большой (не умещается в буфере), проблема решается простым отбрасыванием новых поступающих пакетов.

Очевидно, что сервис "с максимальными усилиями" обеспечивает приемлемое качество обслуживания только в тех случаях, когда производительность сети намного превышает средние потребности, то есть является избыточной. В такой сети пропускная способность достаточна даже для поддержания трафика пиковых периодов нагрузки. Также очевидно, что такое решение не экономично – по крайней мере, по отношению к пропускным способностям сегодняшних технологий и инфраструктур, особенно для глобальных сетей. Тем не менее, построение сетей с избыточной пропускной способностью, будучи самым простым способом обеспечения нужного уровня качества обслуживания, иногда применяется на практике. Например, некоторые провайдеры услуг сетей TCP/IP предоставляют гарантию качественного обслуживания, постоянно поддерживая определенный уровень превышения пропускной способности своих магистралей по сравнению с потребностями клиентов.

В условиях, когда многие механизмы поддержки качества обслуживания только разрабатываются, использование для этих целей избыточной пропускной способности часто оказывается единственно возможным, хотя и временным решением.

ЗАКЛЮЧЕНИЕ

Специалисты в области сетевых технологий утверждают, что половина знаний в этой динамичной области полностью устаревают за пять лет.

Авторы надеются, что учебное пособие, которое студенты прочитали, создало запас знаний, с помощью которого они могли бы обновлять переменную "половину" знаний о постоянно изменяющемся мире компьютерных сетей.

СПИСОК ЛИТЕРАТУРЫ

1. Вычислительные системы, сети и телекоммуникации: Учебник / А.П. Пятибратов, Л.П. Гудыно, А.А. Кириченко; Под ред. А.П. Пятибратова. М.: Финансы и статистика, 2001. 400 с.
2. Компьютерные сети. Принципы, технологии, протоколы / В.Г. Оли-фер, Н.А. Олифер. СПб.: Питер, 2003. 864 с.
3. Якубайтис Э.А. Информационные сети и системы. Справочная книга. М.: Финансы и статистика, 1996. 368 с.
4. Толковый словарь сетевых терминов и аббревиатур. СПб.: Изд-во Вильямс, 2002. 368 с.
5. Олифер В.Г., Олифер Н.А. Основы сетей передачи данных: Курс лекций. М.: ИНТУИТ.РУ "Интернет Ун-т Информационных технологий", 2003. 248 с.
6. Олифер В.Г., Олифер Н.А. Новые технологии и оборудование IP- сетей. СПб.: БХВ-Петербург, 2003. 512 с.
7. Гук М. Аппаратные средства локальных сетей. Энциклопедия. – Теоретические и практические вопросы построения сетей. СПб.: Изд-во Питер, 2002. 576 с.
8. Вито А. Основы организации сетей Cisco. М.: Изд-во Вильямс, 2002. 512 с.
9. Ретана А., Слайс Д., Уайт Р. Принципы проектирования корпоративных IP-сетей. Экзамен на получение квалификации сертифицированного специалиста по межсетевому обмену CISCO. СПб.: Изд-во Вильямс, 2002. 368 с.
10. Кульгин М.В. Компьютерные сети. Практика построения. СПб.: Питер, 2003. 464 с.
11. Шалин П.А. Компьютерная сеть своими руками. СПб.: Питер, 2003. 176 с.
12. Смит Р. Сетевые средства Linux. СПб.: Изд-во Вильямс, 2003. 672 с.
13. Альбитц П., Ли К. DNS и BIND. М.: Изд-во "Символ-Плюс", 2004. 688 с.
14. Дуглас Э. Камер, Дэвид Л. Стивенс Сети TCP/IP. Разработка приложений типа клиент/сервер.: Пер. с англ. М.: Изд-во Вильямс, 2002. 592 с.

ПРИЛОЖЕНИЕ

1. ТЕРМИНОЛОГИЯ ПРОТОКОЛА HTTP

Соединение (connection) **– виртуальный канал транспортного уровня, установленный между двумя программами с целью связи.**

Сообщение (message) **– основной модуль HTTP связи, состоящей из структурной последовательности октетов, соответствующих синтаксису протокола и передаваемых по соединению.**

Запрос (request) – **любое HTTP сообщение, содержащее запрос.**

Ответ (response) – **любое HTTP сообщение, содержащее ответ.**

Ресурс (resource) – **сетевой объект данных или сервис, который может быть идентифицирован URI. Ресурсы могут быть доступны в нескольких представлениях (например на нескольких языках, в разных форматах данных, иметь различный размер или различную разрешающую способность) или различаться по другим параметрам.**

Объект (entity) – **информация, передаваемая в качестве полезной нагрузки запроса или ответа. Объект состоит из метаинформации в форме полей заголовка объекта и содержания в форме тела объекта.**

Представление (representation) – **объект включенный в ответ и подчиняющийся обсуждению содержимого (Content Negotiation). Может существовать несколько представлений, связанных со специфическими состояниями ответа.**

Обсуждение содержимого (content negotiation) – **механизм для выбора соответствующего представления во время обслуживания запроса. Представление объектов в любом ответе может быть обсуждено (включая ошибочные ответы).**

Вариант (variant) – **ресурс может иметь одно, или несколько представлений, связанных с ним в данный момент. Каждое из этих представлений называется "вариант". Использование термина "вариант" не обязательно подразумевает, что ресурс подчинен обсуждению содержимого.**

Клиент (client) – **программа, которая устанавливает соединения с целью посылки запросов.**

Агент пользователя (user agent) – **клиент, который инициирует запрос. Как правило браузеры, редакторы, роботы (spiders), или другие инструментальные средства пользователя.**

Сервер (server) – **приложение, которое слушает соединения, принимает запросы на обслуживание и посылает ответы. Любая такая программа способна быть как клиентом, так и сервером; наше использование данного термина относится скорее к роли, которую программа выполняет, создавая специфические соединения, нежели к возможностям программы вообще. Аналогично, любой сервер может действовать как первоначальный сервер (origin server), прокси-сервер (proxy), шлюз (gateway) или туннель (tunnel), изменяя поведение, основываясь на характере каждого запроса.**

Первоначальный сервер (origin server) – **сервер, на котором данный ресурс находится постоянно или должен быть создан.**

Прокси-сервер (proxy) – **программа-посредник, которая действует и как сервер, и как клиент с целью создания запросов от имени других клиентов. Запросы обслуживаются прокси-сервером, или пересылаются им, возможно с изменениями. Прокси-сервер, согласно этой спецификации, должен удовлетворять требованиям клиента и сервера.**

Шлюз (gateway) – **сервер, который действует как посредник для некоторого другого сервера. В отличие от прокси-сервера, шлюз получает запросы в качестве первоначального сервера для запрошенного ресурса; клиент запроса может не знать, что он соединяется со шлюзом.**

Туннель (tunnel) – **программа-посредник, которая поддерживает соединение. Один раз созданный, туннель не рассматривается как часть HTTP связи, хотя туннель, возможно, был инициализирован запросом HTTP. Туннель прекращает существовать, когда оба конца соединения закрываются.**

Кэш (cache) – **локальная память, в которой программа хранит сообщения-ответы, и в которой располагается подсистема, управляющая хранением, поиском и удалением сообщений. Кэш сохраняет ответы, которые могут быть сохранены, чтобы уменьшить время ответа и загрузку сети (траффик) при будущих эквивалентных запросах. Любой клиент или сервер может иметь кэш, но кэш не может использоваться сервером, который действует как туннель.**

Кэшируемый (cachable) – **ответ является кэшируемым, если кэшу разрешено сохранить копию ответного сообщения для использования при ответе на последующие запросы. Даже если ресурс кэшируем, могут существовать дополнительные ограничения на использование кэшем сохраненной копии для исходного запроса.**

Непосредственный (first-hand) – **ответ считается непосредственным, если он приходит непосредственно от первоначального сервера без ненужной задержки, возможно через один или несколько прокси-серверов. Ответ также является непосредственным, если его достоверность только что была установлена непосредственно первоначальным сервером.**

Точное время устаревания (explicit expiration time) **– время определенное первоначальным сервером и показывающее кэшу когда объект больше не может быть возвращен клиенту без дополнительной проверки достоверности.**

Эвристическое время устаревания (heuristic expiration time) **– время устаревания, назначенное кэшем, если не указано точное время устаревания.**

Возраст (age) **– возраст ответа – время, прошедшее с момента отсылки, или успешной проверки ответа первоначальным сервером.**

Время жизни (freshness lifetime) **– отрезок времени между порождением ответа и моментом устаревания.**

Свежий (fresh) **– ответ считается свежим, если его возраст еще не превысил время жизни.**

Просроченнный (stale) **– ответ считается просроченным, если его возраст превысил время жизни.**

Семантически прозрачный (semantically transparent) **– говорят, что кэш ведет себя "семантически прозрачным" образом в отношении специфического ответа, когда использование кэша не влияет ни на клиента запроса, ни на первоначальный сервер, но повышает эффективность. Когда кэш семантически прозрачен, клиент получает точно такой же ответ (за исключением промежуточных (hop-by-hop) заголовков), который получил бы, запрашивая непосредственно первоначальный сервер, а не кэш.**

Указатель достоверности (validator) **– элемент протокола (например, метка объекта или время последней модификации (Last-Modified time)), который используется, чтобы выяснить, является ли находящаяся в кэше копия эквивалентом объекта.**