КУРС ЛЕКЦИЙ
«Сети и телекоммуникации»

Оглавление

1. ПРОФИЛИ ПРОТОКОЛОВ INTERNET

1.1. Определения

Вычислительная сеть (ВС)– ассоциация территориально распределенных компьютеров, объединенных средствами коммутации и каналами связи. (см. рис.1.1)

Сети передачи данных (СПД) – обобщенное понятие, определяющее сеть связи и включающее в свой состав средства коммутации и каналы связи. (см.рис.1.2)

Средства коммутации – технические устройства, обеспечивающие подключение и взаимодействие компьютеров и каналов связи.

Абонент (пользователь) ВС – человек-оператор или прикладная программа, которые являются источниками и/или потребителями информации вычислительной сети.

Узел ВС – обобщенное понятие технического устройства, выполняющего определенные сетевые функции.

Терминальный узел – техническое устройство, часто персональный компьютер с необходимым программным обеспечением, подключенный к КВ и используемый абонентом для ввода и вывода информации.

Коммутирующий узел – техническое средство, обычно на базе специализированного компьютера, которое осуществляет сопряжение компьютеров и каналов связи и поддерживает обмен данными в требуемых режимах.

Сервер (server) – сетевой компьютер, предоставляющий абонентам ВС услуги по хранению, поиску и передаче информации.

Хостмашина (host) (рабочая машина) – сетевой компьютер, предоставляющий абонентам ВС услуги по информационно-вычислительной обработке данных.

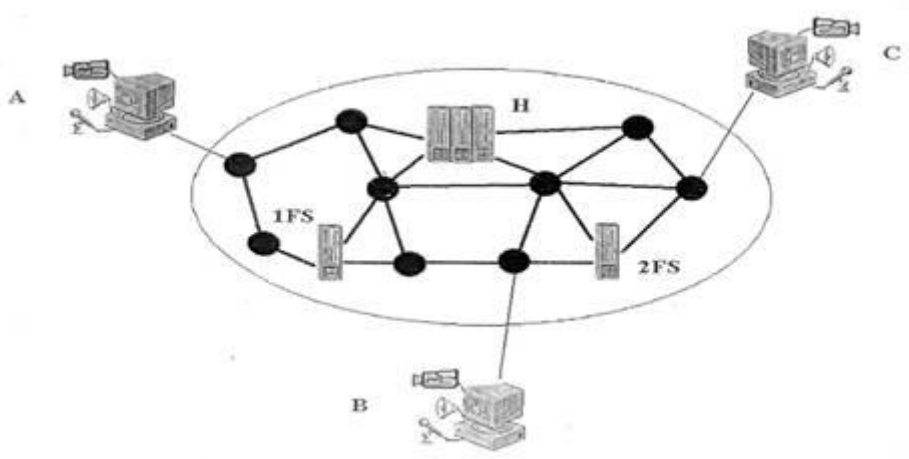


Рис. 1.1.Схема вычислительной сети

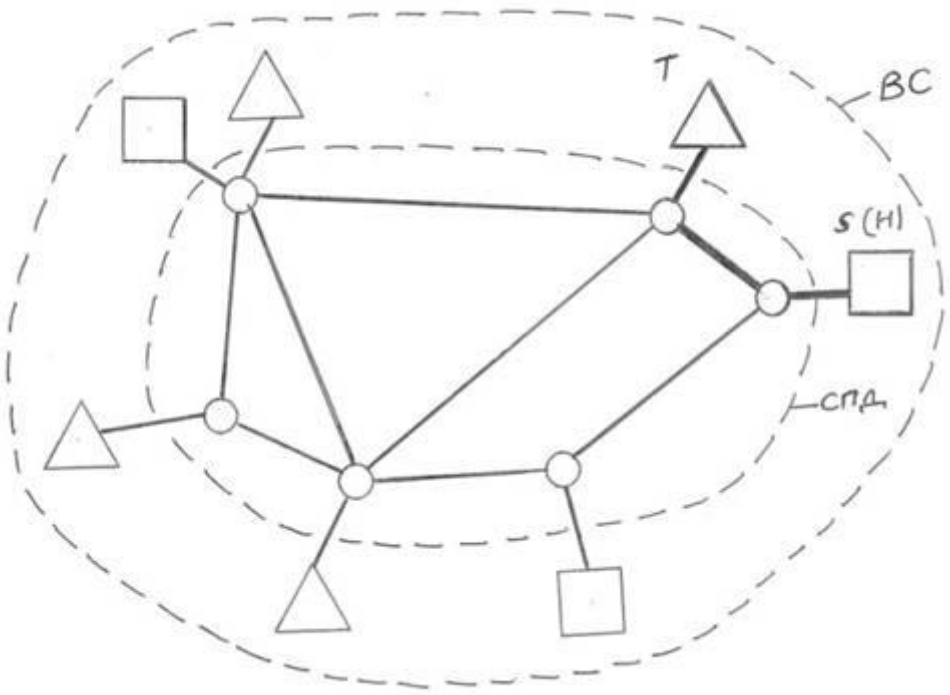


Рис. 1.2 Схема ВС и СПД

1.2. Тракт телеобработки данных (ТОД).

Типовой тракт телеобработки данных (ТОД) представляет фрагмент вычислительной сети (см.рис. 1.2), который состоит из следующих узлов (см.рис. 1.3):

Хостмашины (Н) или сервера (S) (см.рис. 1.4),
коммутирующего узла (Com),
аппаратуры передачи данных (АПД),
линии связи (ЛС),
терминального узла (Т).

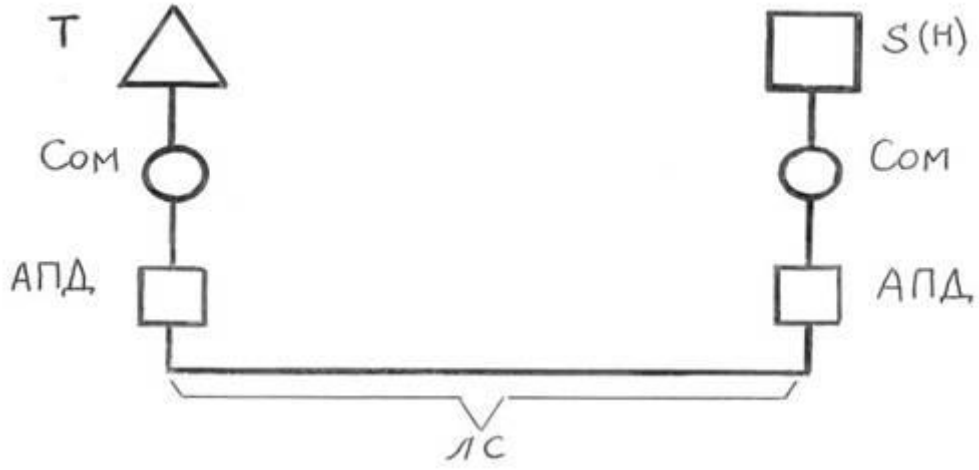


Рис. 1.3 Схема типового тракта телеобработки данных(ТОД)

Любую, даже самую сложную структуру ВС можно рассматривать как множество трактов телеобработки данных, обеспечивающих взаимодействие терминальных узлов Т с соответствующими хостмашинами (Н) или серверами (S), которые предоставляют информационно-вычислительные ресурсы.

Следовательно, для понимания функционирования любой ВС достаточно знать технические и функциональные характеристики типового тракта телеобработки данных.

Кратко напомним особенности технической реализации узлов, входящих в состав ТОД.

Хостмашины (Н), сервера (S) и терминальные узлы (Т) (см. рис.1.4) представляют собой компьютер, который содержат в своем составе:

ПР- процессор

ОЗУ – оперативное запоминающее устройство, хранящее: данные в регистрах **(Рег),** программы операционной системы**(ОС),** канальные программы**(КПр)**

НЖМД – накопитель на жестком магнитном диске

ОШ – общая шина

К1, К2 - многопортовые контроллеры внешних устройств

КЛ – клавиатура

Мон – монитор

М – мышь

П – принтер

НГМД – накопитель на гибком магнитном диске

Современные компьютеры оснащены многопортовыми контроллерами (КК), которые выступают в качестве ***коммутирующих узлов*** (**Com**) и обеспечивают взаимодействие с АПД при построении территориально распределенной сети (WAN), либо с контроллерами локальной вычислительной сети (КЛВС), при построении локальной вычислительной сети (LAN).

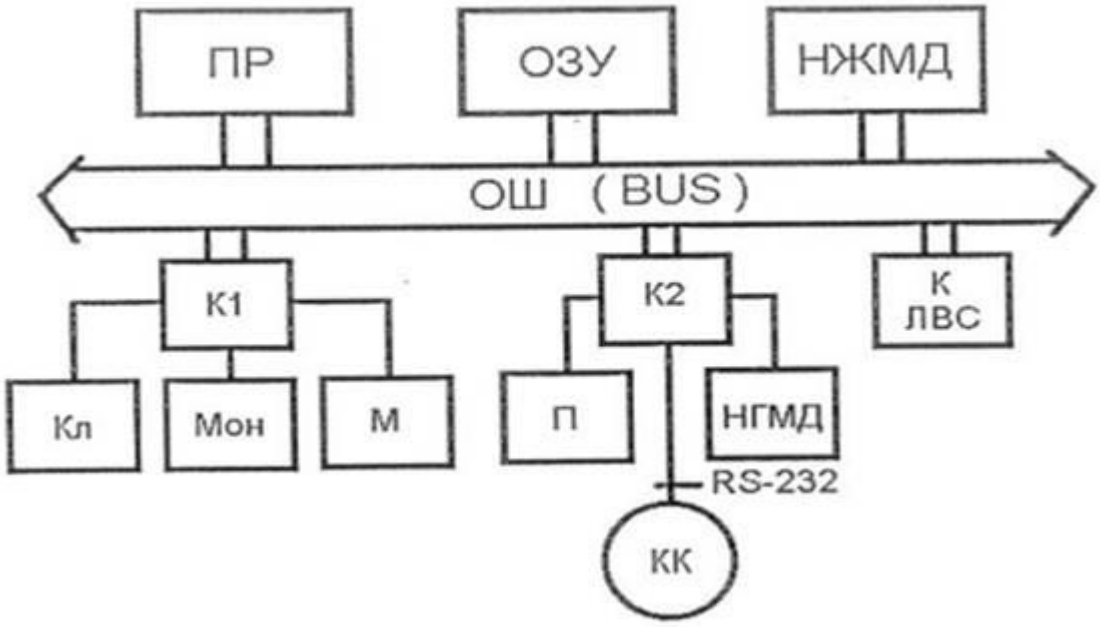


Рис. 1.4 Схема компьютера ВС (хост, сервер, терминал)

Одним из часто реализуемых представителей АПД при построении WAN является модем.

Модем (модулятор-демодулятор) является устройством, преобразующим последовательные цифровые сигналы в аналоговые и наоборот. Схема подключения модема представлена на рис.

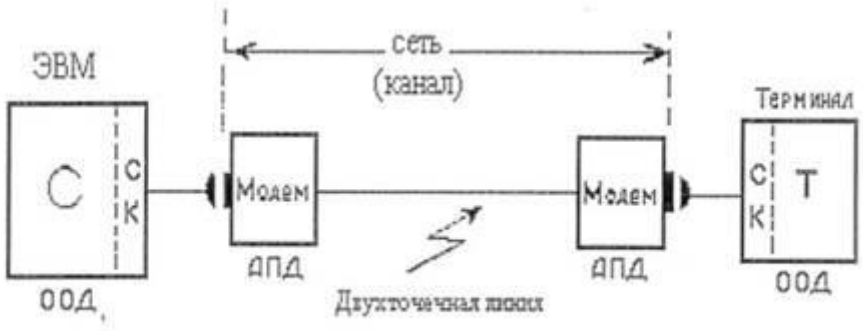


Рис.1.5. Схема подключения модема

Модуляция передаваемого сигнала – это изменение одного или нескольких параметров несущего синусоидального колебания (амплитуды, частоты, фазы) в соответствии со значением передаваемой двоичной информации.

Линии связи образуют физическую среду передачи сигналов, к которой относятся телефонная пара, витая пара, коаксиальный кабель, волоконно-оптический кабель, «эфир» электромагнитных полей.

АПД, включающие аппаратные средства и необходимое программное обеспечение, вместе с линиями связи линии связи обеспечивают реализацию каналов связи, которые по функциональным особенностям разделяются на симплексные, дуплексные и полудуплексные.

Симплексный канал связи обеспечивает передачу информации только в одном направлении (телевидение).

Дуплексный канал связи обеспечивает передачу информации одновременно и независимо в двух направлениях (поддержка диалога).

Полудуплексный канал связи обеспечивает передачу информации поочередно: то в одном, то в другом направлении (обычно спецсвязь).

При передаче данных могут использоваться два принципа связи: коммутация линий связи и коммутация пакетов.

При коммутации линий на время передачи данных организуется физическое соединение нескольких линий связи.

На рис. 1.6 приведен пример, трех коммутируемых каналов, состоящих из трех участков. При этом по линии, соединяющей два центра коммутации линий (ЦКД-Л) организовано три канала связи. На время установления соединения каждому коммутируемому каналу выделен ресурс производительности:

- каналу терминал Т 1 - хост Н1 выделена производительность 9600 бит/ с;
- каналу принтер Пр - хост Н1 выделена производительность 300 бит/ с;
- каналу хост Н2 - терминал Т 2 выделена производительность 2400 бит/ с;

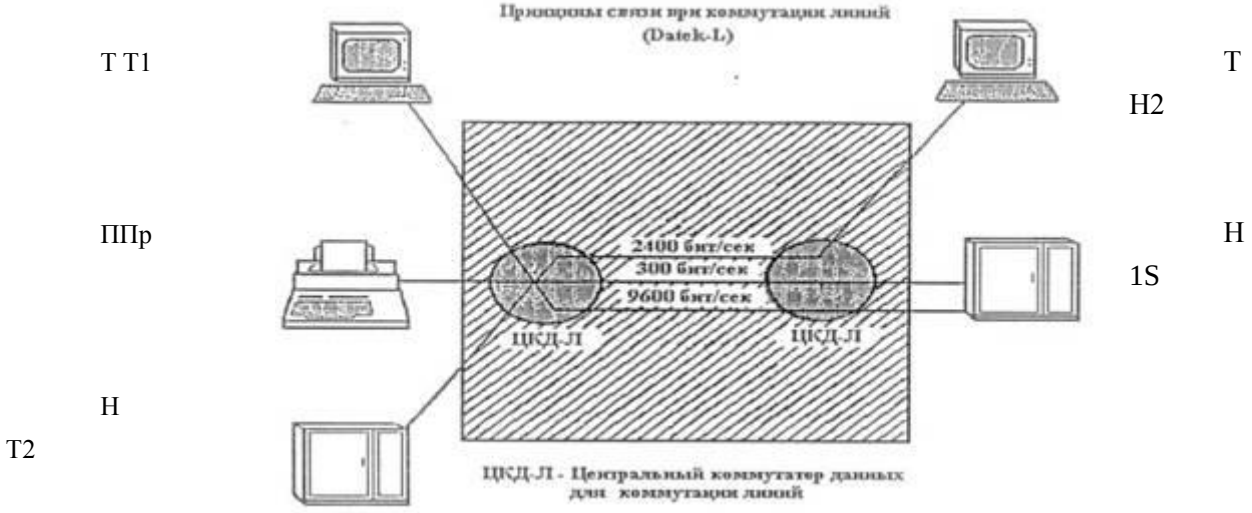


Рис.1.6 Коммутация линий связи

При коммутации пакетов в каждом коммутирующем узле имеется буфер, в котором каждое передаваемое сообщение (пакет данных), может ожидать освобождения очередного канала для передачи данных .

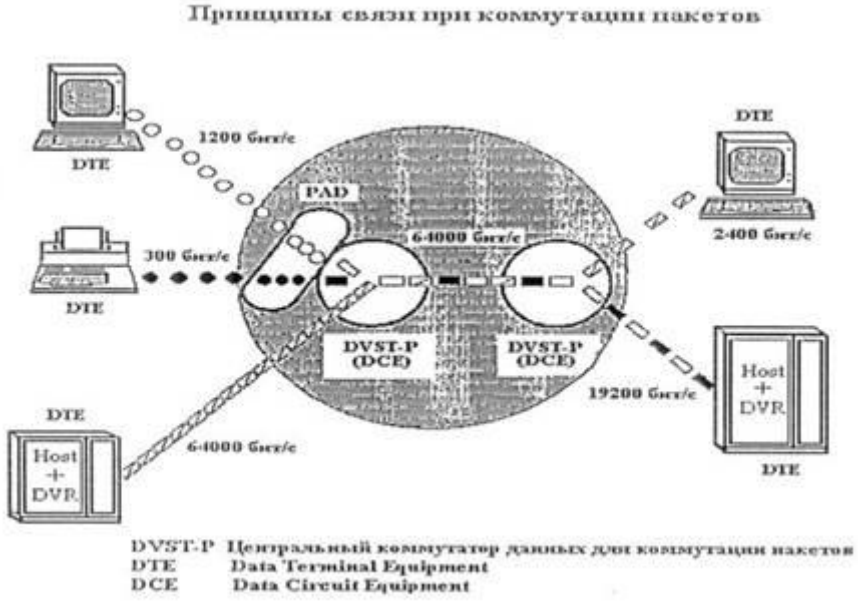


Рис. 1.7. Коммутация пакетов

На примере, представленном на рис.1.7, показано, что можно сократить число каналов за счет объединения в одном канале потоков данных различных групп пользователей.

1.3 Функции тракта ТОД

При функционировании вычислительной сети абоненты используют 2 режима работы: локальный и удаленный

При *локальном* режиме работы абонент посредством клавиатуры и мыши осуществляет взаимодействие с прикладной программой (ПП1), записанной в накопителе на жестком магнитном диске НЖМД. При изменении ПП1 на ПП2 выполняются следующие упрощенно представленные процедуры:

Исходное состояние: взаимодействуют **ПР - Рег(ОЗУ) – ОС –ПП1**

Команда М изменения **ПП: М – КВВ – ПР**

ПР - Рег(ОЗУ) – ОС –ПП2

При режиме *удаленного подключения* к **ПП2**, расположенной на удаленном сервере, абонент посредством клавиатуры и мыши осуществляет выполнение следующих упрощенно представленных процедур:

Исходное состояние: взаимодействуют **ПР - Рег(ОЗУ) – ОС –ПП 1**.

Команда М изменения **ПП: М – КВВ – ПР.**

Ввод данных: **ОС**(модуль прерываний) – **Рег(ОЗУ)** - канальные программы**(КПр).**

Формирование параметров сетевой программы: **ОС**(модуль прерываний) – Сетевая программа **(СетП).**

Передача управления **КК: ПР – К2**.

Передача пакета: **КК - АПД1.**

Доставка пакета: **АПД1 – АПД2.**

Прием пакета: **АПД2 – КК.**

Прерывание удаленного сервера: **К2 – ПР.**

10) Формирование параметров сетевой программы: **ОС**(модуль прерываний) – Сетевая программа **(СетП).**

11) Загрузка **ПП2: ОС**(модуль прерываний) – **Рег(ОЗУ)** - канальные программы **(КПр).**

12) Запуск удаленной программы: **ПР - Рег(ОЗУ) – ОС –ПП 1**.

Главная проблема, возникающая при функционировании ВС, состоит в обеспечении для узлов ВС такой согласованности, инвариантности команд и выполняемых функций, которая бы не зависела от технических характеристик устройств и особенностей применяемого программного обеспечения.

2. БАЗОВЫЕ ПРОФИЛИ ПРОТОКОЛОВ INTERNET И СЕМИУРОВНЕВАЯ МОДЕЛЬ ОТКРЫТЫХ СИСТЕМ

2.1 Наборы функций тракта ТОД

В процессе создания первых вычислительных сетей фирмы, производящие и разрабатывающие компьютеры, активно подключились к созданию узлов ВС, реализующих функции тракта ТОД. Однако, корпоративные разработки не позволяли обеспечить в полной мере реализацию тракта ТОД продуктами (аппаратными и программными) одной корпорации. Остро встала задача обеспечения согласованной работы различного сетевого оборудования.

Для решения задачи обеспечения совместного функционирования продуктов различных фирм потребовалось:

провести анализ функций, выполняемых узлами ВС в тракте ТОД;

разработать унифицированную архитектуру ВС.

Анализ функций, выполняемых узлами ВС в тракте ТОД позволил установить следующее.

Каналы связи и АПД (См. Рис. 2.1) выполняют набор функций ***{Д}***, обеспечивающий доставку блока цифровых данных на любые (даже удаленные) расстояния.

Коммутирующий узел (Com) выполняет набор функций ***{М}***, обеспечивающий выбор маршрута при реализации режимов коммутации каналов и коммутации пакетов для обмена блоками цифровых данных между терминальным узлом (Т) и хостмашиной (Н) или сервером (S).

Хостмашина (Н), сервер (S), терминальный узел (Т) выполняют для обмена блоками цифровых данных между терминальным узлом (Т) и хостмашиной (Н) или сервером (S) два набора функций:

{Т} – обеспечение транспортировки больших объемов данных;

{П} – запуск удаленных прикладных программ и получение результатов выполнения обработки.

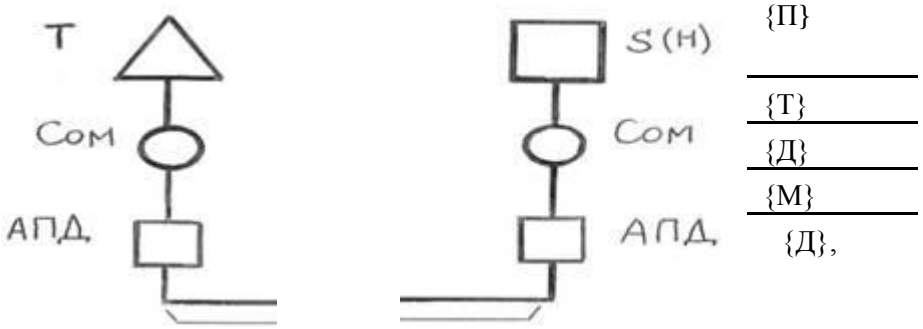


Рис.2.1. Наборы функций тракта ТОД.

2.2 Тракт ТОД и модель OSI

В 1984 году с целью упорядочения описания принципов взаимодействия устройств в сетях Международная организация по стандартизации (International Organization of Standardization — ISO) предложила семиуровневую эталонную коммуникационную модель «Взаимодействие Открытых Систем» (Open System Interconnection, OSI). Модель OSI стала основой для разработки стандартов на взаимодействие систем. Она определяет только схему выполнения необходимых задач, но не дает конкретного описания их выполнения. Это описывается конкретными протоколами или правилами, разработанными для определенной технологии с учетом модели OSI. Уровни OSI могут реализовываться как аппаратно, так и программно.

Существует семь основных уровней модели OSI:

7-й Прикладной (Application)

6-й Представления (Presentation)

5-й Сеансовый (Session)

4-й Транспортный (Transport)

3-й Сетевой (Network)

2-й Канальный (Data Link)

1-й Физический (Physical)

Они начинаются с физического уровня и заканчиваются прикладным. Каждый уровень предоставляет услуги для более высокого уровня. Седьмой уровень обслуживает непосредственно пользователей.

Сравнение семиуровневой эталонной модели OSI с функциями тракта ТОД приведено на рис. 2.2.

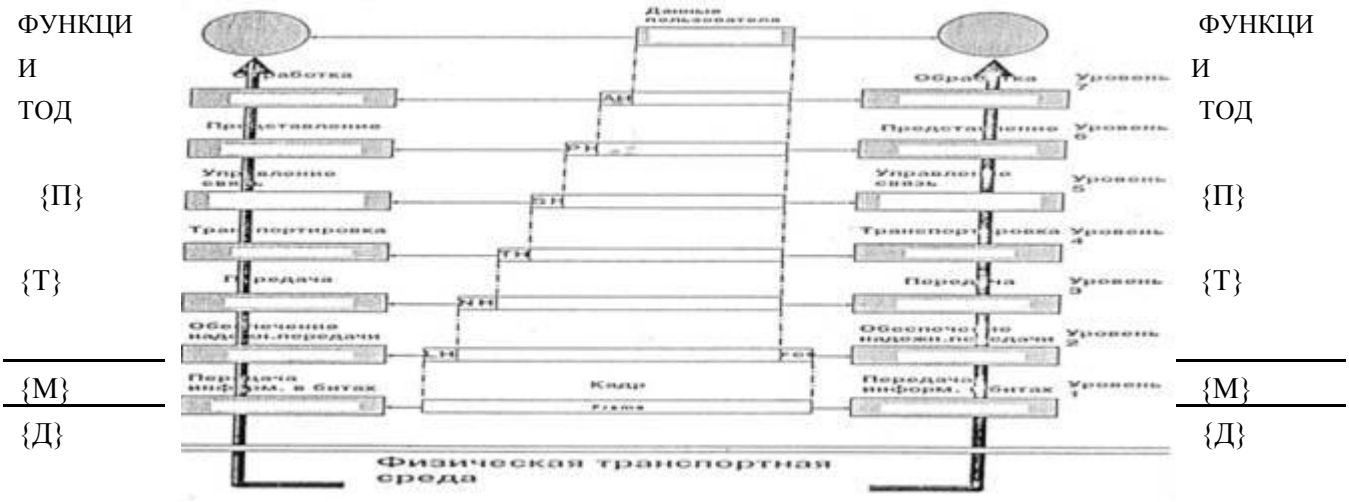


Рис.2.2. Функции тракта ТОД и модели OSI

Модель OSI послужила основой для стандартизации всей сетевой индустрии. Кроме того, модель OSI является хорошей методологической основой для изучения сетевых технологий. Несмотря на то, что были разработаны и другие модели (в основном, патентованные), большинство поставщиков сетевого оборудования определяет свои продукты в терминах эталонной модели OSI.

Эталонная модель OSI сводит передачу информации в сети к семи относительно простым подзадачам. Каждая из них соответствует своему строго определенному уровню модели OSI. Тем не менее, в реальной жизни некоторые аппаратные и программные средства отвечают сразу за несколько уровней. Два самых низких уровня модели OSI реализуются как аппаратно, так и программно. Остальные пять уровней, в основном, — программные.

Эталонная модель OSI определяет назначение каждого уровня и правила взаимодействия уровней (табл. 2.1).

Таблица 2.1. Уровни модели OSI

Уровень	Ключевое слово	Данные	Ответственность
Прикладной	Разделение	Сообщение	Предоставляет сетевой сервис
Представления	Форматирование	Пакет	Трансляция данных и файлов Шифрование данных Сжатие данных
Сеансовый	Диалог	Пакет	Управление сессией Диалог Контроль за ошибками Обработка транзакций Поддержка вызовов удаленных процедур RPC
Транспортный	Надежность	Сегмент, дейтаграмма, пакет	Надежность передачи Гарантированная доставка Мультиплексирование сессий верхнего уровня
Сетевой	Дейтаграмма	Дейтаграмма	Маршрутизация логических адресов Создание и ведение таблиц маршрутизации Фрагментация и сборка данных Неориентированная на соединение и ненадежная доставка
Канальный	Кадр	Кадр, пакет	Окончательная доставка по физическому адресу устройства Синхронизация кадров Доступ к среде передачи
Физический	Биты	Биты	Синхронизация битов Сигнализация, аналоговая или цифровая Электрическая и механические спецификации

Модель OSI описывает путь информации через сетевую среду от одной прикладной программы на одном компьютере до другой программы на другом компьютере. При этом пересылаемая информация проходит вниз через все уровни системы. Уровни на разных системах не могут общаться между собой напрямую. Это умеет только физический уровень. По мере прохождения информации вниз внутри системы она преобразуется в вид, удобный для передачи по физическим каналам связи. Для указания адресата к этой преобразованной информации добавляется заголовок с адресом. После получения адресатом этой информации, она проходит через все уровни наверх. По мере прохождения информация преобразуется в первоначальный вид. Каждый уровень системы должен полагаться на услуги, предоставляемые ему смежными уровнями.

Основная идея модели OSI в том, что одни и те же уровни на разных системах, не имея возможности связываться непосредственно, должны работать абсолютно одинаково. Одинаковым должен быть и сервис между соответствующими уровнями различных систем. Нарушение этого принципа может привести к тому, что информация, посланная от одной системы к другой, после всех преобразований будет не похожа на исходную.

Проходящие через уровни данные имеют определенный формат. Сообщение, как правило, делится на заголовок и информационную часть. Конкретный формат зависит от функционального назначения уровня, на котором информация находится в данное время. Например, на сетевом уровне информационный блок состоит из сетевого адреса и следующими за ним данными. Данные сетевого уровня, в свою очередь, могут содержать заголовки более высоких уровней — транспортного, сеансового, уровня представления и прикладного. И, наконец, не все уровни нуждаются в присоединении заголовков. Некоторые уровни просто выполняют преобразование получаемых физических данных к формату, подходящему для смежных уровней.

2.2 Модель OSI и Internet

Корпоративная сеть — это сложная система, состоящая из большого числа разнообразных компонентов: компьютеров, концентраторов, маршрутизаторов, коммутаторов, системного прикладного программного обеспечения и т. д. Основная задача системных интеграторов и администраторов сетей состоит в том, чтобы эта система как можно лучше справлялась с обработкой потока информации и позволяла пользователям решать их прикладные задачи. Прикладное программное обеспечение часто обращается к службе, обеспечивающей связь с другими прикладными программами по сети. Этой службой является механизм межсетевого обмена.

Парадигмы обработки корпоративной информации не постоянны. Примером резкого изменения подхода к обработке корпоративной информации стал беспрецедентный рост популярности глобальной сетиInternet за последнее десятилетие. Сеть Internet изменила способ представления информации, объединив на своих узлах все ее виды — текст, графику и звук. Транспортная система сети Internetсущественно облегчила задачу построения распределенной корпоративной сети.

Основой сети Internet является набор протоколов, называемый стеком протоколов TCP/IP (Transmission Control Protocol/Internet Protocol, протокол управления передачей/протокол сети Internet). Он реализует межсетевой обмен.

Основное достоинство стека протоколов TCP/IP в том, что он обеспечивает надежную связь между сетевым оборудованием от различных производителей. Протоколы TCP/IP предоставляют механизм передачи сообщений, описывают формат сообщений и указывают, как обрабатывать ошибки. Протоколы позволяют описать и понять процессы передачи данных независимо от типа оборудования, на котором эти процессы происходят.

История создания TCP/IP ведет свое начало с момента, когда министерства обороны США столкнулось с проблемой объединения большого числа компьютеров с различными операционными системами. В 1970 году был разработан необходимый набор стандартов. Протоколы, разработанные на базе этих стандартов, получили обобщенное название TCP/IP.

Стек TCP/IP был изначально разработан для сети Advanced Research Project Agency Network (ARPANET). ARPANET рассматривалась как экспериментальная распределенная сеть с коммутацией пакетов. Эксперимент применению TCP/IP в этой сети закончился положительно. В результате с протоколов был принят в промышленную эксплуатацию, а в дальнейшем расширялся и совершенствовался в течение нескольких лет. Позже стек адаптировали для использования в локальных сетях.

В начале 1980 года протокол с составной частью операционной системы Berkley UNIX v4.2. В том же году появилась объединенная сеть Internet. Переход к технологии Internet б завершен в 1983 году, когда министерство обороны США решило, что все компьютеры, присоединенные к глобальной сети, будут использовать стек протоколов TCP/IP.

Стек TCP/IP предоставляет пользователям две основные службы, которые используют прикладные программы:

Дейтаграммное средство доставки пакетов. Это означает, что протоколы стека TCP/IP определяют - маршрут передачи небольшого сообщения, основываясь только на адресной информации, находящейся в

этом сообщении. Доставка осуществляется без установки логического соединения. Такой тип доставки делает протоколы TCP/IP адаптируемыми к широкому диапазону сетевого оборудования;

Надежное потоковое транспортное средство. Большинство приложений требуют от коммуникационного программного обеспечения автоматического восстановления при ошибках передачи, потери пакетов или сбоев промежуточных маршрутизаторах. Надежное транспортное средство позволяет устанавливать логическое соединение между приложениями, а затем посылать большие объемы данных по этому соединению.

Основными преимуществами стека протоколов TCP/IP являются:

Независимость от сетевой технологии. TCP/IP не зависит от оборудования, так как он только определяет элемент передачи — дейтаграмму - описывает способ ее движения по сети;

Всеобщая связанность. Стек позволяет любой паре компьютеров, которые его поддерживают, взаимодействовать друг с другом. Каждому компьютеру назначается логический адрес, а каждая передаваемая дейтаграмма содержит логические адреса отправителя и получателя. Промежуточные маршрутизаторы используют адрес получателя для принятия решения маршрутизации;

Подтверждения. Протоколы стека TCP/IP обеспечивают подтверждения правильности прохождения информации при обмене между отправителем и получателем;

Стандартные прикладные протоколы. Протоколы TCP/IP включают в свой состав средства поддержки основных приложений, таких как электронная почта, передача файлов, удаленный доступ и т. д.

Резкий рост сети Internet и все большее распространение стека протоколов TCP/IP привело к выпуску серии документов, которые призваны способствовать дальнейшему упорядоченному развитию протоколов. Организация Internet Activities Board (IAB) выпускает документы, называемые RFC (Request For Comments, обращение за разъяснением). Некоторые RFC описывают сетевые службы или протоколы и их реализацию, другие документы описывают условия применения. В RFC опубликованы стандарты TCP/IP. При этом следует иметь в виду, что стандарты TCP/IP всегда публикуются в виде документов RFC, но не все RFC определяют стандарты.

Документы RFC первоначально публиковались в электронном виде. Документ мог претерпевать несколько изменений до тех пор, пока не достигалось общее соглашение по его содержанию. Если документ при этом регламентировал какой-то аспект протокола, то ему присваивался номер. При этом каждому новому документу присваивается статус, указывающий на необходимость его внедрения. Выход в свет нового документа RFC вовсе не означает, что все производители оборудования и программного обеспечения должны внедрять его в своей продукции. В приложении 2 приведено описание некоторых документов RFC и их статусов.

Содержание документов RFC делится на две части:

Состояние стандартизации. Протокол может иметь несколько состояний:

Стандарт на протокол утвержден;

Стандарт на протокол предлагается к рассмотрению;

Предлагается экспериментальный протокол;

Протокол устарел и в настоящее время не используется;

Статус протокола. Протокол может иметь несколько статусов:

Протокол должен быть внедрен;

Протокол рекомендуется для внедрения;

Протокол может внедряться производителем по выбору;

Протокол не рекомендуется для внедрения.

В сложной корпоративной сети при ее эксплуатации возникает масса проблем. Решить их функциональными возможностями одного протокола практически невозможно. Такой протокол должен был бы:

Распознавать сбои в сети и восстанавливать ее работоспособность;

Распределять пропускную способность сети и уменьшать поток данных при перегрузке;

Распознавать задержки и потери пакетов и принимать контрмеры;

Распознавать ошибки в данных и информировать о них прикладное программное обеспечение;

Упорядочивать движение пакетов в сети.

Такое количество функциональных возможностей не может вместить ни один протокол. Поэтому был создан набор взаимодействующих протоколов, названный *стеком*. Так как стек протоколов TCP/IP был разработан до появления эталонной модели OSI, то соответствие его уровней с уровнями OSI достаточно условно. В табл. 2.2 показаны структура стека протоколов TCP/IP и соответствие уровней OSI и стека TCP/IP уровням модели ATM.

Таблица 2.2. Соответствие уровней стека TCP/IP уровням моделей OSI

Уровни модели OSI	Уровни модели TCP/IP	Протоколы
7 Прикладной	Уровень: прикладной	Telnet FTP, SMTP, DNS,SNMP и т. д.
6 Представления	Уровень: представления	HTTP, UNIX, Windows
5 Сеансовый	Уровень: сеансовый	TCP,
4 Транспортный	Уровень: транспортный	TCP, UDP
3 Сетевой	Уровень: сетевой	ARP, Classical ARP, IP, Classical IP, ICMP и т.д.
2 Канальный	Уровень: сетевого интерфейса	Ethernet FDOI, Token Ring, ATM и т. д.
1 Физический	Уровень: физический	SMT, 10BASE-X

Теоретически, посылка сообщения от одной прикладной программы к другой означает последовательную передачу сообщения вниз по уровням стека у отправителя, передачу сообщений по уровню сетевого интерфейса (уровню 2) или, в соответствии с эталонной моделью OSI, по физическому уровню, прием сообщения получателем и передачу его вверх по уровням. Физический уровень может быть реализован транспортными механизмами ATM. На практике, взаимодействие уровней стека организовано гораздо

сложнее. Каждый уровень принимает решение о правильности сообщения и производит определенное действие на основании типа сообщения или адреса назначения. В структуре стека протоколов TCP/IP имеется явный «центр тяжести» — это сетевой уровень и его протокол IP. Протокол IPможет взаимодействовать с несколькими протоколами более высокого уровня и несколькими сетевыми интерфейсами. То есть, на практике процесс передачи сообщений от одной прикладной программы к другой будет выглядеть следующим образом: отправитель передает сообщение, которое на уровне протоколом IP помещается в дейтаграмму и посылается в сеть (сеть 1). На промежуточных устройствах, например маршрутизаторах, дейтаграмма передается вверх до уровня протокола IP, который отправляет ее обратно вниз, в другую сеть (сеть 2). Когда дейтаграмма достигает получателя, протокол IPвыделяет сообщение и передает его на верхние уровни.

Структура стека протоколов TCP/IP в целом соответствует модели OSI (см. рис.2.3).

Самый нижний — уровень сетевого интерфейса — соответствует физическому и канальному уровням модели OSI. В стеке протоколов TCP/IP этот уровень не регламентирован.

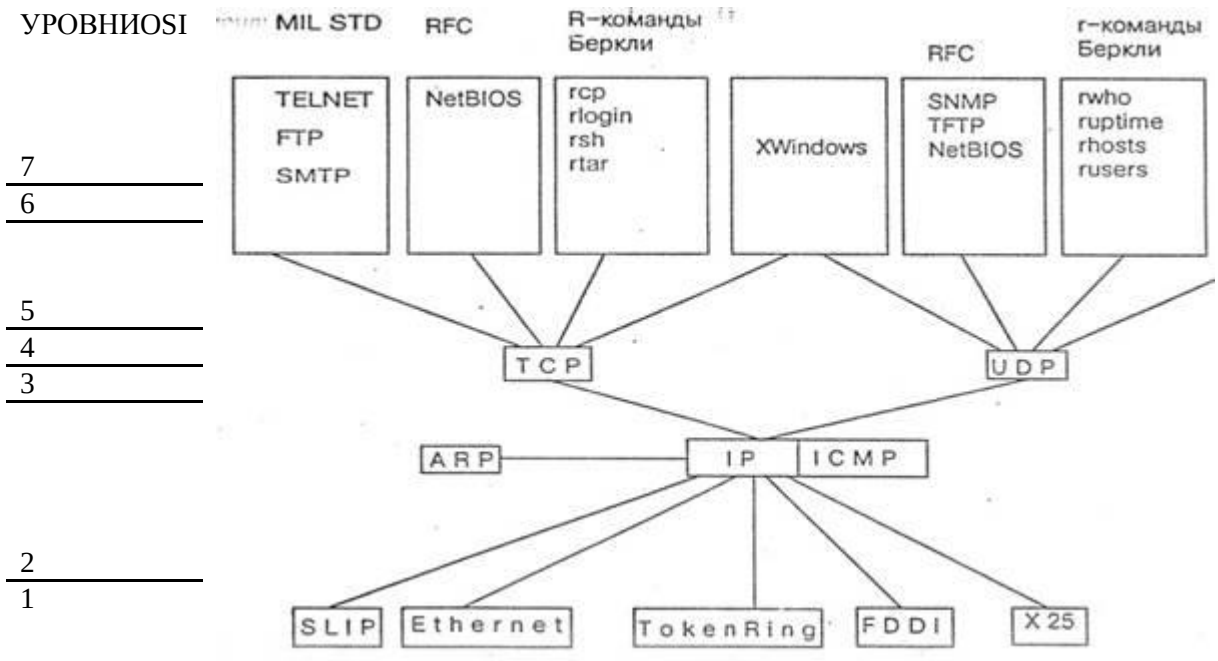


Рис 2.3 Структура стека протоколов TCP/IP

Уровень сетевого интерфейса отвечает за прием дейтаграмм и передачу их по конкретной сети. Интерфейс с сетью может быть реализован драйвером устройства или сложной системой, которая использует свой протокол канального уровня (коммутатор, маршрутизатор). Он поддерживает стандарты физического и канального уровней популярных локальных сетей: Ethernet, Token Ring, FDDI и т. д. Для распределенных сетей поддерживаются протоколы соединений PPP и SLIP, а для глобальных сетей — протокол X.25. Предусмотрена поддержка технологии коммутации ячеек — ATM. Обычной практикой стало включение в стек протоколов TCP/IP новых технологий локальных или распределенных сетей и регламентация их новыми документами RFC.

Сетевой уровень — это уровень межсетевого взаимодействия. Уровень управляет взаимодействием между пользователями в сети. Он принимает запрос на посылку пакета от транспортного уровня вместе с указанием адреса получателя. Уровень инкапсулирует пакет в дейтаграмму, заполняет ее заголовок и при необходимости использует алгоритм маршрутизации. Уровень обрабатывает приходящие дейтаграммы и проверяет правильность поступившей информации. На стороне получателя дейтаграммы программное обеспечение сетевого уровня удаляет заголовок дейтаграммы и определяет, какой из транспортных протоколов будет обрабатывать пакет.

В качестве основного протокола сетевого уровня в стеке протоколов TCP/IP используется протокол IP, который создавался как раз с целью передачи информации в распределенных сетях. Достоинством протокола IP является возможность его эффективной работы в сетях со сложной топологией. При этом

протокол рационально использует пропускную способность низкоскоростных линий связи. В основе протокола IP заложен дейтаграммный метод, который не гарантирует доставку пакета, но «стремится» к этому. Для поддержки совместной работы протокола IP и технологии ATM комитетом IETF был разработан стандарт «Классический IP и ARP поверх ATM», который позволяет преобразовывать IP-адреса сетевого уровня в адреса ATM и передавать пакеты TCP/IP по сети ATM. К этому уровню относятся все протоколы, которые создают, поддерживают и обновляют таблицы маршрутизации. Кроме того, на этом уровне функционирует протокол обмена информацией об ошибках между маршрутизаторами в сети и отправителями.

Следующий уровень — транспортный. Основной его задачей является взаимодействие между удаленными прикладными программами. Транспортный уровень управляет потоком информации и обеспечивает надежность передачи. Для этого использован механизм подтверждения правильного приема с дублированием передачи утерянных или пришедших с ошибками пакетов. Транспортный уровень принимает данные от нескольких прикладных программ и посылает их более низкому уровню. При этом он добавляет дополнительную информацию к каждому пакету, в том числе контрольную сумму. На этом уровне функционирует протокол управления передачей данных TCP (Transmission Control Protocol) и протокол передачи прикладных пакетов дейтаграммным методом UDP (User DatagramProtocol).

Протокол TCP обеспечивает гарантированную доставку данных за счет образования логических соединений между удаленными прикладными процессами. Работа протокола UDP аналогична IP, но основной его задачей является связь сетевого протокола и различных приложений.

Самый верхний уровень— прикладной. На этом уровне реализованы широко используемые сервисы прикладного уровня. К ним относятся: протокол передачи файлов между удаленными системами (FTP), протокол эмуляции удаленного терминала (telnet), почтовые протоколы, протокол разрешения имен (DNS) и т. д. Каждая прикладная программа выбирает тип транспортировки — либо непрерывный поток сообщений, либо последовательность отдельных сообщений. Прикладная программа передает данные транспортному уровню в требуемой форме.

Стек протоколов TCP/IP включает большое число протоколов прикладного уровня. Они выполняют различные функции, в том числе: управление сетью, передачу файлов, оказание распределенных услуг при использовании файлов, эмуляцию терминалов, передачу электронной почты и т. д. Протокол передачи файлов (File Transfer Protocol — FTP) обеспечивает пересылку файлов между компьютерными системами. Протокол Telnet эмулирует терминал. Простой протокол управления сетью (Simple Network Management Protocol — SNMP) используется для сообщений об аномальных условиях в сети и установления значений допустимых порогов в сети. Простой протокол передачи почты (Simple Mail Transfer Protocol — SMTP) обеспечивает пересылку электронной почты. Эти протоколы и другие приложения используют TCP/IP.

Стек протоколов TCP/IP еще долгое время будет базовым в корпоративных сетях. Это связано с практически полным отсутствием новых приложений, способных работать самостоятельно поверх сетейATM. В этой связи в последние годы разработано и предложено к внедрению несколько различных вариантов использования стека протоколов TCP/IP в сетях ATM, иногда даже несмотря на некоторое снижение эффективности и пропускной способности сетей ATM. Например, эффективность TCP достаточно серьезно снижается из-за необходимости проведения фрагментации при поступлении пакета TCP/IP на уровень адаптации ATM (AAL). Необходимость преобразования пакета связана со значительно большим его размером по отношению к ячейке. Резкое снижение эффективности сети может произойти при потере хотя бы одной ячейки. В этом случае неизбежной будет повторная передача всего пакета, что приведет к дополнительной загрузке сети.

Стек TCP/IP — это тот сетевой и протокольный базис, на котором построен Internet. В стеке протоколов TCP/IP схема идентификации абонентов в сети аналогична физической адресации. Каждому устройству в сети присваивается уникальный 32-битный адрес, который называется IP-адресом. Он имеет вполне определенную структуру. В адрес входит идентификатор сети, к которой подсоединено устройство, и идентификатор самого устройства, уникальный в данной сети.

Компьютеры или другие сложные сетевые устройства, подсоединенные к нескольким физическим сетям, имеют несколько IP-адресов — по одному на каждый сетевой интерфейс. Можно сказать, что адрес в

сети Internet назначается не отдельному устройству, а сетевому интерфейсу. Схема адресации позволяет проводить единичную (unicast), широковещательную (broadcast) и групповую (multicast) адресацию.

Широковещательная адресация позволяет обращаться ко всем устройствам в сети. В этих адресах поле идентификатора устройства заполнено единицами. IP-адресация допускает широковещательную передачу, но не гарантирует ее - эта возможность зависит от конкретной физической сети. Например, в сетях Ethernet широковещательная передача выполняется с той же эффективностью, что и обычная передача данных, но есть сети, которые вообще не поддерживают такой тип передачи или поддерживают его весьма ограниченно.

Групповая адресация используется для отправки сообщений определенным адресатам (multicasting). Поддержка групповой адресации обязательна для многих приложений, например, интерактивных конференций, электронной почты и групп новостей. Для групповой передачи рабочие станции и маршрутизаторы используют протокол IGMP (Internet Group Management Protocol), который предоставляет информацию о принадлежности устройств определенным группам.

IP-адреса больших сетей в Internet определяются специальной организацией
- Internet Network Information Center (InterNIC). Назначение идентификаторов устройств не входит в компетенцию InterNIC и находится в ведении системного администратора конкретной сети. До 1 апреля 1993 г. (дата создания InterNIC) назначение IP-адресов и имен доменов DNS выполнялось организацией Network InformationCenter (NIC). В настоящее время NIC обслуживает только сети министерства обороны США — DDN (Defense Data Network). Все остальные организации, входящие в Internet, обслуживаются InterNIC.

В больших корпоративных сетях управлять IP-адресами «вручную» практически невозможно. В частности, чрезвычайно сложно гарантировать уникальность адресов. А назначать IP-адреса приходится достаточно часто. Например, если рабочая станция перемещается в другую подсеть, необходимо сменить ее сетевой идентификатор. Для помощи сетевому администратору в этом нелегком деле разработаны различные программные продукты и сетевые технологии.

В стек TCP/IP входит множество протоколов, отвечающих различным уровням эталонной модели OSI.

2.4 Базовые профили протоколов Internet

2.4.1. Базовый Модуль Internet IP создает единую логическую сеть

Архитектура протоколов TCP/IP предназначена для объединенной сети, состоящей из соединенных друг с другом шлюзами отдельных разнородных пакетных подсетей, к которым подключаются разнородные машины. Каждая из подсетей работает в соответствии со своими специфическими требованиями и имеет свою природу средств связи. Однако предполагается, что каждая подсеть может принять пакет информации (данные с соответствующим сетевым заголовком) и доставить его по указанному адресу в этой конкретной подсети. Не требуется, чтобы подсеть гарантировала обязательную доставку пакетов и имела надежный сквозной протокол. Таким образом, две машины, подключенные к одной подсети могут обмениваться пакетами.

Когда необходимо передать пакет между машинами, подключенными к разным подсетям, то машина-отправитель посылает пакет в соответствующий шлюз (шлюз подключен к подсети также как обычный узел). Оттуда пакет направляется по определенному маршруту через систему шлюзов и подсетей, пока не достигнет шлюза, подключенного к той же подсети, что и машина-получатель; там пакет направляется к получателю. Объединенная сеть обеспечивает датаграммный сервис.

Проблема доставки пакетов в такой системе решается путем реализации во всех узлах и шлюзах межсетевого протокола IP. Межсетевой уровень является по существу базовым элементом во всей архитектуре протоколов, обеспечивая возможность стандартизации протоколов верхних уровней.

2.4.2. Структура связей протокольных модулей

Логическая структура сетевого программного обеспечения, реализующего протоколы семейства TCP/IP в каждом узле сети Internet, изображена на рис. 2.4 Прямоугольники

обозначают обработку данных, а линии, соединяющие прямоугольники, - пути передачи данных. Горизонтальная линия внизу рисунка обозначает кабель сети Ethernet, которая используется в качестве примера физической среды; "o" - это трансивер. Знак "*" – обозначает IP-адрес, а "@" - адрес узла в сети Ethernet (Ethernet-адрес). Понимание этой логической структуры является основой для понимания всей технологии Internet. В дальнейшем мы будем часто ссылаться на эту схему.

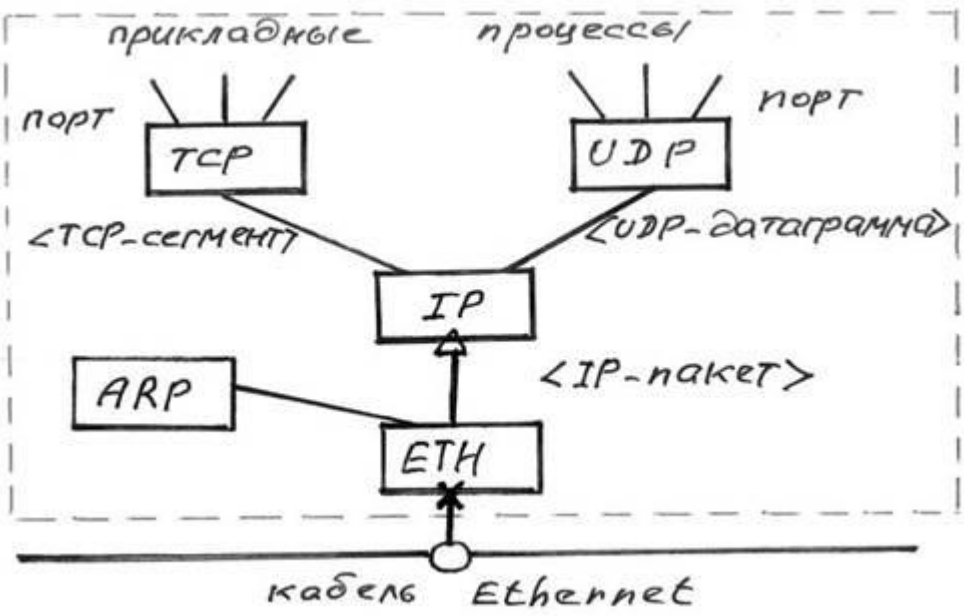


Рис.2.4. Структура протокольных модулей в узле сети TCP/IP

2.4.3. Терминология

Введем ряд базовых терминов, которые мы будем использовать в дальнейшем.

Драйвер - это программа, непосредственно взаимодействующая с сетевым адаптером.

Модуль - это программа, взаимодействующая с драйвером, сетевыми прикладными программами или другими модулями. Драйвер сетевого адаптера и, возможно, другие модули, специфичные для физической сети передачи данных, предоставляют сетевой интерфейс для протокольных модулей семейства TCP/IP.

Название блока данных, передаваемого по сети, зависит от того, на каком уровне стека протоколов он находится. Блок данных, с которым имеет дело сетевой интерфейс, называется кадром; если блок данных находится между сетевым интерфейсом и модулем IP, то он называется IP-пакетом; если он - между модулем IP и модулем UDP, то - UDP-датаграммой; если между модулем IP и модулем TCP, то - TCP-сегментом (или транспортным сообщением); наконец, если блок данных находится на уровне сетевых прикладных процессов, то он называется прикладным сообщением.

Эти определения, конечно, несовершенны и неполны. К тому же они меняются от публикации к публикации. Более подробные определения можно найти в RFC-1122.

2.4.4. Потоки данных

Рассмотрим потоки данных, проходящие через стек протоколов, изображенный на рис.2.4. В случае использования протокола TCP (Transmission Control Protocol - протокол управления передачей), данные передаются между прикладным процессом и модулем TCP. Типичным прикладным процессом, использующим протокол TCP, является модуль FTP (File Transfer Protocol - протокол передачи файлов). Стек протоколов в этом случае будет FTP/TCP/IP/ENET. При использовании протокола UDP (User Datagram Protocol

- протокол пользовательских датаграмм), данные передаются между прикладным процессом и модулем UDP. Например, SNMP (Simple Network Management Protocol - простой протокол управления сетью) пользуется транспортными услугами UDP. Его стек протоколов выглядит так: SNMP/UDP/IP/ENET.

Модули TCP, UDP и драйвер Ethernet являются мультиплексорами **n** x **1.**

Действуя как мультиплексоры, они переключают несколько входов на один выход. Они также являются демультиплексорами **1** x **n.** Как демультиплексоры, они переключают один вход на один из многих выходов в соответствии с полем типа в заголовке протокольного блока данных (рис.2.5).

Когда Ethernet-кадр попадает в драйвер сетевого интерфейса Ethernet, он может быть направлен либо в модуль ARP (Address Resolution Protocol - адресный протокол), либо в модуль IP (Internet Protocol - межсетевой протокол). На то, куда должен быть направлен Ethernet-кадр, указывает значение поля типа в заголовке кадра.

Если IP-пакет попадает в модуль IP, то содержащиеся в нем данные могут быть переданы либо модулю TCP, либо UDP, что определяется полем "протокол" в заголовке IP-пакета.

Если UDP-датаграмма попадает в модуль UDP, то на основании значения поля "порт" в заголовке датаграммы определяется прикладная программа, которой должно быть передано прикладное сообщение. Если TCP-сообщение попадает в модуль TCP, то выбор прикладной программы, которой должно быть передано сообщение, осуществляется на основе значения поля "порт" в заголовке TCP-сообщения.

Мультиплексирование данных в обратную сторону осуществляется довольно просто, так как из каждого модуля существует только один путь вниз. Каждый протокольный модуль добавляет к пакету свой заголовок, на основании которого машина, принявшая пакет, выполняет демультиплексирование.

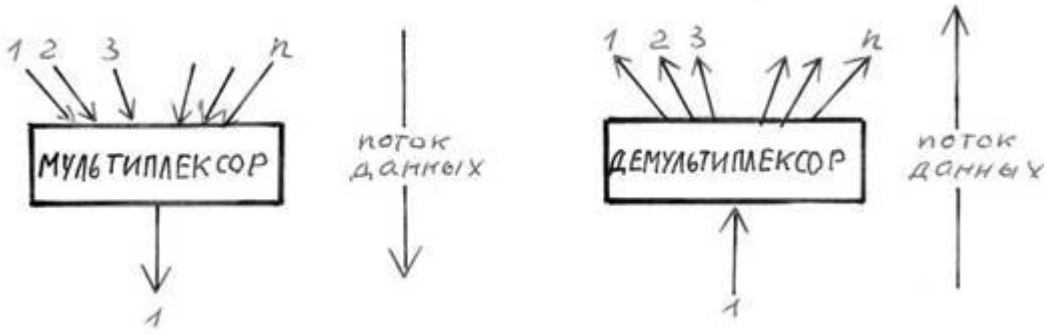


Рис.2.5 Мультиплексор **n** x **1** и демультиплексор **1** x **n**

Данные от прикладного процесса проходят через модули TCP или UDP,
после чего попадают в модуль IP и оттуда - на уровень сетевого интерфейса.

Хотя технология internet поддерживает много различных сред передачи данных, здесь мы будем предполагать использование Ethernet, так как именно эта среда чаще всего служит физической основой для IP-сети.

Машина на рис. 2.4 имеет одну точку соединения с Ethernet. Шестибайтный Ethernet-адрес является уникальным для каждого сетевого адаптера и распознается драйвером.

Машина имеет также четырехбайтный IP-адрес. Этот адрес обозначает точку доступа к сети на интерфейсе модуля IP с драйвером. IP-адрес должен быть уникальным в пределах всей сети Internet.

Работающая машина всегда знает свой IP-адрес и Ethernet-адрес.

2.4.5. Работа с несколькими сетевыми интерфейсами

Машина может быть подключена одновременно к нескольким средам передачи данных. На рис.2.6 показана машина с двумя сетевыми интерфейсами Ethernet. Заметим, что она имеет 2 Ethernet-адреса и 2 IP-адреса.

Из представленной схемы видно, что для машин с несколькими сетевыми интерфейсами модуль IP выполняет функции мультиплексора ***n*** x ***m*** и демультиплексора ***m*** x ***n*** (рис.4 2.7)

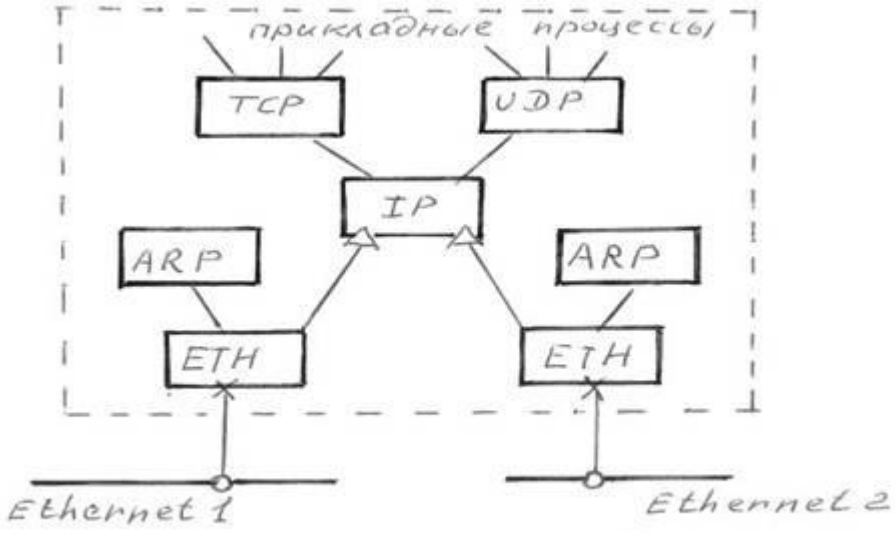


Рис. 2.6 Узел сети TCP/IP с двумя сетевыми интерфейсами

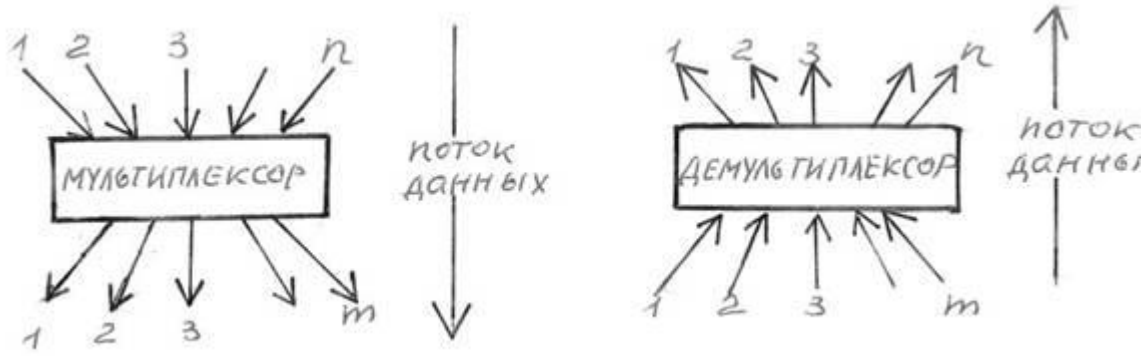


Рис. 2.7 Мультиплексор **n** x **m** и демультиплексор **m** x **n**

Таким образом, он осуществляет мультиплексирование входных и выходных данных в обоих направлениях. Модуль IP в данном случае сложнее, чем в первом примере, так как может передавать данные между сетями. Данные могут поступать через любой сетевой интерфейс и быть ретранслированы через любой другой сетевой интерфейс. Процесс передачи пакета в другую сеть называется ретрансляцией IP-пакета. Машина, выполняющая ретрансляцию, называется шлюзом. [1] Ретранслируемый пакет не передается модулям TCP или UDP. Некоторые шлюзы вообще могут не иметь модулей TCP и UDP.

3.КАНАЛЬНЫЕ ПРОТОКОЛЫ РАСПРЕДЕЛЕННЫХ СЕТЕЙ

В этом разделе кратко рассматривается технология Ethernet и на ее основе излагаются примеры, поясняющие взаимодействие протоколов Internet.

Кадр Ethernet содержит адрес назначения, адрес источника, поле типа и данные. Размер адреса в Ethernet - 6 байт. Каждый сетевой адаптер имеет свой Ethernet-адрес. Адаптер контролирует обмен информацией

3.1 Технология Ethernet

Технология Ethernet была разработана в исследовательском центре компании Xerox в 70-х годах и достигла своего нынешнего лидирующего положения в 80-х. Впервые термин Ethernet был использован Робертом Меткалфом в заметке, написанной им в этом исследовательском центре в мае 1973 года.

Технология Ethernet стала базой спецификации IEEE 802.3, которая была опубликована в 1980 году. Вскоре после этого компании Digital Equipment (DEC), Intel и Xerox совместно разработали и приняли вторую версию спецификации Ethernet, совместимую с IEEE 802.3. В настоящее время термин Ethernet чаще всего используют для описания всех локальных сетей, работающих в соответствии с принципами CSMA/CD (Carrier Sense Multiple Access/Collision Detection) — множественного доступа с контролем несущей и обнаружением коллизий, что соответствует спецификации Ethernet IEEE 802.3. В модели OSI протокол CSMA/CD относится к доступу к среде. На этом уровне определяется формат, в котором информация передается по сети, и способ, с помощью которого сетевое устройство получает доступ к сети (или управление сетью) для передачи данных.

CSMA/CD состоит из двух частей: Carrier Sense Multiple Access и Collision Detection. Первая часть определяет, каким образом рабочая станция с сетевым адаптером «ловит» момент, когда ей следует послать сообщение. В соответствии с протоколом CSMA, рабочая станция вначале слушает сеть, чтобы определить, не передается ли в данный момент какое-либо другое сообщение. Если слышится несущий сигнал (carrier tone), значит, в данный момент сеть занята другим сообщением — рабочая станция переходит в режим ожидания и находится в нем до тех пор, пока сеть не освободится. Когда в сети наступает молчание, станция начинает передачу. Вторая часть — Collision Detection — служит для разрешения ситуаций, когда две или более рабочие станции пытаются передавать сообщения одновременно. Если две станции начнут передавать свои пакеты одновременно, передаваемые данные належатся друг на друга и ни одно из сообщений не дойдет до получателя. Такую ситуацию называют конфликтом или коллизией (сигналы одной станции перемешаются с сигналами другой). Collision Detection требует, чтобы станция прослушала сеть также и после передачи пакета. Если обнаруживается конфликт, станция повторяет передачу пакета через случайным образом выбранный промежуток времени. Затем она вновь проверяет, не произошел ли конфликт. Термин «множественный доступ» подчеркивает тот факт, что все станции имеют одинаковое право на доступ к сети.

Если одна из станций обнаружит коллизию, она пошлет специальный сигнал, предупреждающий другие станции о произошедшем конфликте. При коллизии уничтожаются все данные в сети. После коллизии станции пытаются передать данные повторно. Для того чтобы предотвратить одновременную передачу, был разработан специальный механизм прерываний, который предписывает каждой станции выждать случайный промежуток времени перед повторной передачей. Станция, которой достался самый короткий период ожидания, первой получит право на очередную попытку передать данные, а остальные определят, что сеть занята и вновь будут ожидать. Единицей измерения времени ожидания является удвоенное время распространения сигнала из конца в конец отрезка кабеля, равное примерно 51.2 мс. После первого конфликта каждая станция ждет 0 или 1 единицу времени, прежде чем попытается возобновить передачу. Если снова произошел конфликт, что может быть, если две станции выбрали одно и то же число, то каждая из них выбирает одно из четырех случайных чисел: 0, 1, 2, 3. Если и в третий раз произошел конфликт, случайное число выбирается из интервала 0-7 и т. д. После десяти последовательных конфликтов интервал выбора случайных чисел фиксируется и становится равным 0-1023. После шестнадцати конфликтов контроллер отказывается от дальнейших попыток передать кадр и сообщает об этом компьютеру. Все дальнейшие действия по выходу из сложившейся ситуации осуществляются под руководством протоколов верхнего уровня. Такой алгоритм позволяет разрешить коллизии, когда конфликтующих станций немного.

Обнаружение конфликта основано на сравнении посланных сигналов и сигналов других рабочих станций. Аппаратное обеспечение станции должно во время передачи «прослушивать» кабель для определения факта коллизии. Если сигнал, который станция регистрирует, отличается от передаваемого ею, значит, произошла коллизия. Поэтому должен существовать механизм, позволяющий различать сигналы в кабеле. Этот механизм был найден — им стало манчестерское кодирование и дифференциальное манчестерское кодирование сигнала.

При манчестерском кодировании каждый интервал времени, в течение которого происходит передача одного бита, разделяется на две половинки. Единичный бит кодируется высоким напряжением в первой

половине и низким напряжением во второй. Нулевой бит кодируется противоположным образом. Изменение напряжения в середине интервала облегчает принимающей стороне синхронизацию с передающей станцией.

Дифференциальное манчестерское кодирование представляет собой разновидность обычного манчестерского кодирования. В этом случае единичный бит характеризуется отсутствием изменения напряжения (напряжения в обеих половинках равны). Изменение напряжения в начале бита означает, что это нулевой бит.

Недостатком схемы манчестерского кодирования является необходимость удвоения ширины полосы пропускания по сравнению с прямым кодированием. Однако вследствие своей простоты манчестерское кодирование используется в стандарте 802.3. Уровни высокого и низкого напряжения составляют +0.85 В и -0.85 В. Прямое двоичное кодирование построено на кодировании нулевого бита нулевым напряжением (0 В) и единичного бита ненулевым напряжением (5 В). Сеть Ethernet относится к категории широковещательных. В таких сетях все станции видят все кадры в независимости от того, являются ли они их получателями. Каждая станция должна проверять, не ей ли предназначаются передаваемые данные. Полученные данные передаются на следующий уровень.

В технологии Ethernet данные могут передаваться по коаксиальному или оптическому кабелю, а также через витую пару. Чаще всего при построении локальных сетей на основе этой технологии оптический кабель используется для формирования магистрали сети, в то время как витая пара применяется для подключения станций и серверов. Спецификации Ethernet были созданы в то время, когда для быстрой передачи данных требовались коаксиальные кабели. Необходимость перехода на менее дорогие телефонные кабели и попытка смягчить последствия разрыва коаксиального кабеля стали причинами появления спецификации 10Base-T IEEE 802.3. Эта спецификация определяет технологию Ethernet для сетей, построенных на базе неэкранированных витых пар и телефонных кабелей. При этом допускается звездообразная топология. Приведем основные спецификации Ethernet.

10Base5. Как и первая версия Ethernet, эта спецификация в качестве среды передачи предусматривает толстый коаксиальный кабель на 50 Ом с двумя оболочками. Из-за этого спецификацию называют «толстым Ethernet». Каждый коаксиальный кабель в сети образует отдельный сегмент. Протяженность сегмента не может превышать 500 м, а число узлов не должно превосходить 100. Отрезок кабеля между соседними узлами должен быть не менее 2.5 м. Это позволяет уменьшить вероятность отражения и появления стоячих волн. Как правило, производители размечают кабель с целью упростить нахождение мест, где станция может быть подключена к сегменту. Сетевой адаптер подключается к кабелю с помощью трансиверного кабеля и трансивера. Длина трансиверного кабеля не должна превышать 50 м.

10Base2. Эта спецификация предусматривает использование тонкого коаксиального кабеля, а также соединителей типа BNC-T, которые непосредственно связывают сетевой адаптер и кабель Ethernet. Такая схема исключает необходимость применения дорогостоящих трансиверов и трансиверных кабелей. Кроме того, значительно упрощается выполнение самой операции по подключению сетевого адаптера к кабелю. Этот стандарт известен как «тонкий Ethernet». Протяженность сегмента ограничена 185 м, а число узлов — 30.

10BaseT. Эта разновидность Ethernet получила наибольшее распространение. Буква T в названии означает, что средой передачи является неэкранированная витая пара (Unshielded Twisted Pair, UTP). Спецификация предусматривает использование концентратора для подключения пользователей по топологии «звезда». Применение дешевых кабелей UTP является одним из основных преимуществ l0BaseT по сравнению со спецификациями 10Base2 и 10Base5. Подключение узлов к сети осуществляется с помощью телефонных гнезд RJ-45 и RJ-11 и четырехпарного телефонного кабеля UTP. Вилка RJ-45 вставляется напрямую в сетевую плату. Протяженность отрезка кабеля от концентратора до станции не должна превышать 100 м (в случае UTP категории 3) или 150 м (в случае UTP категории 5).

10BaseF. Эта спецификация использует в качестве среды передачи оптоволоконный кабель. Применение оптоволоконной технологии приводит к высокой стоимости комплектующих. Однако нечувствительность к электромагнитным помехам позволяет использовать спецификацию в особо ответственных случаях и для связи далеко расположенных друг от друга объектов.

Каждая из разновидностей Ethernet предусматривает те или иные ограничения на протяженность сегмента кабеля. Для создания более протяженной сети несколько кабелей могут соединяться с помощью повторителей. Повторитель представляет собой устройство физического уровня. Он принимает, усиливает и передает сигнал дальше. С точки зрения программного обеспечения последовательность кабельных сегментов, связанных повторителями, ничем не отличается от единого кабеля. Сеть может содержать несколько сегментов кабеля и несколько повторителей.

Теоретическая производительность сети Ethernet составляет 10 Мбит/сек. Однако нужно учитывать, что из-за коллизий технология Ethernet никогда не сможет достичь своей максимальной производительности. При увеличении числа станций в сети временные задержки между посылками отдельных пакетов по сети возрастают, так как количество коллизий увеличивается. Поэтому реальная производительность Ethernet не превышает 70 % от теоретической.

Можно проследить некоторую зависимость: в сетях с совместным доступом к среде передачи общий объем трафика растет пропорционально квадрату числа станций, в то время как объем полезного трафика увеличивается линейно. Это приводит к тому, что эффективность использования сети оказывается обратно пропорциональной числу станций. Учитывая квадратичный рост объема трафика, любое быстродействие сети оказывается исчерпанным достаточно быстро.

Для снижения нагрузки на сеть ее разбивают на отдельные сегменты с помощью мостов, коммутаторов или маршрутизаторов. Это позволяет передавать между сегментами лишь необходимый трафик. Данные, посылаемые между двумя станциями в одном сегменте, не будут передаваться в другой и, следовательно, не вызовут в нем повышения нагрузки. Однако сегментация в традиционных сетях решает проблему, связанную с совместным доступом к среде передачи, только частично. Она позволяет сократить общий объем трафика между сегментами в число раз, примерно равное среднему количеству сегментов. Однако сегментация не устраняет саму тенденцию роста трафика. Линейный рост трафика достигается только в технологиях, ориентированных на установление соединения.

3.2 Форматы кадров Ethernet

Данные, передаваемые в сети Ethernet, разбиты на кадры. Напомним, что практически каждой сетевой технологии (независимо от ее уровня) соответствует единица передачи данных: Ethernet-кадр, ATM-ячейка, IP-дейтаграмма и т. д. Данные по сети в чистом виде не передаются. Как правило, к единице данных «пристраивается» заголовок. В некоторых сетевых технологиях добавляется также окончание. Заголовок и окончание несут служебную информацию и состоят из определенных полей.

Так как существует несколько типов кадров, для того, чтобы понять друг друга, отправитель и получатель должны использовать один и тот же тип кадров. Кадры могут быть четырех разных форматов, несколько отличающихся друг от друга. Базовых форматов кадров (raw formats) существует всего два — Ethernet II и Ethernet 802.3. Эти форматы отличаются назначением всего одного поля.

Для успешной доставки информации получателю каждый кадр должен кроме данных содержать дополнительную служебную информацию: длину поля данных, физические адреса отправителя и получателя, тип сетевого протокола и т. д.

Большинство сетевых администраторов не уделяет должного внимания типам кадров Ethernet, а это может явиться источником проблем. Например, если клиентское сетевое программное обеспечение настроено на неверный тип кадра, то пользователь не сможет взаимодействовать с сервером. За типом кадра приходится особенно внимательно следить в сетях Novell NetWare, так как в новых версиях этой операционной системы тип кадра по умолчанию был изменен с 802.3 на 802.2. Кроме того, в корпоративных сетях применяются устройства от нескольких поставщиков, базирующихся на разных протоколах взаимодействия и использующих различные типы кадров.

Для того чтобы рабочие станции имели возможность взаимодействовать с сервером в одном сегменте сети, они должны поддерживать единый формат кадра. Существуют четыре основных разновидности кадров Ethernet:

Ethernet Type II

Ethernet 802.3

- Ethernet 802.2

Ethernet SNAP (SubNetwork Access Protocol).

Рассмотрим поля, общие для всех четырех типов кадров (рис. 3.1).

Преамбула (56 бит)	Признак начала кадра (8 бит)	Адрес получателя (48 бит)	Адрес отправителя (48 бит)	Длина/тип (16 бит)	Данные (переменная длина)	Контрольная сумма (32 бит)

Рис. 3.1 Общий формат кадров Ethernet

Поля в кадре имеют следующие значения:

Поля «Преамбула» и **«Признак начала кадра»** предназначены для синхронизации отправителя и получателя. Преамбула представляет собой 7-байтовую последовательность единиц и нулей. Поле признака начала кадра имеет размер 1 байт. Эти поля не принимаются в расчет при вычислении длины кадра.

Поле «Адрес получателя» состоит из 6 байт и содержит физический адрес устройства в сети, которому адресован данный кадр. Значения этого и следующего поля являются уникальными. Каждому производителю адаптеров Ethernet назначаются первые три байта адреса, а оставшиеся три байта определяются непосредственно самим производителем. Например, для адаптеров фирмы 3Com физические адреса будут начинаться с 0020AF. Первый бит адреса получателя имеет специальное значение. Если он равен 0, то это адрес конкретного устройства (только в этом случае первые три байта служат для идентификации производителя сетевой платы), а если 1 — широковещательный. Обычно в широковещательном адресе все оставшиеся биты тоже устанавливаются равными единице (FF FF FFFF FF FF).

Поле «Адрес отправителя» состоит из 6 байт и содержит физический адрес устройства в сети, которое отправило данный кадр. Первый бит адреса отправителя всегда равен нулю.

Поле «Длина/тип» может содержать длину или тип кадра в зависимости от используемого кадра Ethernet. Если поле задает длину, она указывается в двух байтах. Если тип — то содержимое поля указывает на тип протокола верхнего уровня, которому принадлежит данный кадр. Например, при использовании протокола IPX поле имеет значение 8137, а для протокола IP - 0800.

Поле «Данные» содержит данные кадра. Чаще всего — это информация, нужная протоколам верхнего уровня. Данное поле не имеет фиксированной длины.

Поле «Контрольная сумма» содержит результат вычисления контрольной суммы всех полей за исключением преамбулы, признака начала кадра и самой контрольной суммы. Вычисление выполняется отправителем и добавляется в кадр. Аналогичная процедура вычисления выполняется и на устройстве получателя. В случае, если результат вычисления не совпадает со значением данного поля, предполагается, что произошла ошибка при передаче. В этом случае кадр считается испорченным и игнорируется.

Следует отметить, что минимальная допустимая длина для всех четырех типов кадров Ethernet составляет 64 байта, а максимальная — 1518 байт. Так как на служебную информацию в кадре отводится 18 байт, то поле «Данные» может иметь длину от 46 до 1500 байт. Если передаваемые по сети данные меньше

допустимой минимальной длины, кадр будет автоматически дополняться до 46 байт. Столь жесткие ограничения на минимальную длину кадра введены для обеспечения нормальной работы механизма обнаружения коллизий.

Рассмотрим более подробно форматы кадров разных типов. Тип кадра Ethernet II используется многими протоколами верхнего уровня, такими как TCP/IP, IPX и AppleTalk. Данный тип кадра был разработан фирмами DEC, Intel и Xerox. Необходимо учитывать, что хотя данный тип кадра является наиболее широко используемым, он не одобрен организациями IEEE и ISO. Формат данного типа кадра отличается от рассмотренного выше только тем, что в поле «Длина/тип» всегда указывается тип протокола.

Сетевые операционные системы Novell NetWare 2.x и 3.x (за исключением 3.12) по умолчанию используют кадры Ethernet 802.3. Хотя в названии этого типа кадра есть упоминание комитета IEEE, последний не имел никакого отношения к его разработке.

Данный тип кадра не содержит никакой информации о протоколе. Поле «Длина/тип» всегда указывает длину кадра. В результате нет стандартных методов идентификации сетевого протокола, которому принадлежит данный кадр. Однако в соответствии с концепцией фирмы Novell, только протокол IPX может использоваться с данным типом кадров. Разработана специальная последовательность действий для определения того, что именно протокол IPX был инкапсулирован в кадр данного типа.

Проверяется поле «Длина/тип». Если оно содержит значение между 0 и 1518 (05EE), то данное поле определяет длину кадра, а не тип протокола (то есть это кадр 802.3, в противном случае — кадр EthernetII).

Проверяются следующие два байта за полем «Длина/тип». Если они содержат FFFF, это означает, что кадр принадлежит протоколу IPX, так как заголовок этого протокола всегда начинается с FFFF.

В результате стандартизации сетей Ethernet подкомитетом IEEE 802.3 появился кадр Ethernet 802.2. Этот кадр является базовым для операционных систем Novell Netware версий 3.12 и 4-х. В данном типе кадра сразу за адресом отправителя следует поле длины, имеющее такое же назначение. Кроме того, этот тип кадра содержит несколько дополнительных полей, рекомендованных подкомитетом IEEE 802.3 Эти поля располагаются за полем «Длина/тип» и имеют следующее назначение:

Поле «DSAP» указывает на используемый получателем протокол сетевого уровня. Размер поля составляет 1 байт (один бит в нем зарезервирован). Для протокола IPX значение поля равно E0, для протоколов IP — 06, для NetBIOS – F0.

Поле «SSAP» указывает на используемый отправителем протокол сетевого уровня. Размер данного поля составляет 1 байт (один бит зарезервирован). Обычно значение данного поля совпадает со значением поля DSAP.

Поле «Контроль» указывает на тип сервиса, требуемый для сетевого протокола. Размер данного поля составляет 1 байт. Сетевая операционная система Novell NetWare устанавливает значение данного поля в 03.

Формат кадра Ethernet 802.2 имеет некоторые недостатки, в частности он содержит нечетное число байтов служебной информации. Это не совсем удобно для работы большинства сетевых устройств. Кроме того, для идентификации протокола сетевого уровня отводится 7 бит, что позволяет поддерживать «всего» 128 различных протоколов. Кадр Ethernet SNAP, являющийся дальнейшим развитиемEthernet 802.2, содержит следующие дополнительные поля (рис. 1.6):

Поле «Код организации» имеет длину три байта и указывает на код организации (фирмы), которая присвоила значения поля «Идентификатор протокола». Если значение поля равно 000000 (а это так практически всегда, за исключением сетей AplleTalk), то поле «Идентификатор протокола» содержит значение, которое обычно помещается в поле «Длина/ тип», то есть идентификатор протокола верхнего уровня.

Поле «Идентификатор протокола» имеет длину два байта и идентифицирует протокол верхнего уровня, инкапсулированный в поле «Данные» кадра. При использовании протокола IPX это поле содержит значение 8137.

В большинстве локальных и глобальных сетей есть ограничение на максимальный размер кадра. Эту величину называют ***максимальной единицей передачи*** **(MTU-** maximum Transmission Unit**).**

В совокупности эти два поля составляют дополнительное пятибайтовое поле для идентификации протокола. Это было сделано для увеличения числа поддерживаемых протоколов.

<table>
<tr><td colspan="2">Адрес получателя (48 бит)</td></tr>
<tr><td colspan="2">Адрес отправителя (48 бит)</td></tr>
<tr><td colspan="2">Длина (16 бит)</td></tr>
<tr><td>DSAP (AA, 8 бит)</td><td>SSAP (AA, 8 бит)</td></tr>
<tr><td colspan="2">Контрольная сумма (32 бита)</td></tr>
<tr><td>Контроль (8 бит)</td><td>Код организации (000000,24 бита)</td></tr>
<tr><td colspan="2">Идентификатор протокола (8137,16 бит)</td></tr>
<tr><td colspan="2">Данные (переменная длина)</td></tr>
<tr><td colspan="2">Контрольная сумма (32 бита)</td></tr>
</table>

Рис.3.2 Формат кадра Ethernet SNAP

Нужно отметить, что сетевой протокол IPX может использовать любой из рассмотренных выше четырех типов кадров, чего нельзя сказать об остальных сетевых протоколах. В таблице 3.1. приводятся протоколы, которые могут быть использованы с тем или иным типом кадра.

Таблица 3.1.

Совместимость кадров Ethernet с протоколами верхних уровней

Кадр	Протоколы
Ethernet II	IPX, IP, AplleTalk Phase I
Ethernet 802.3	IPX
Ethernet 802.2	IPX, FTAM
Ethernet SNAP	IPX, IP, AplleTalk Phase II

Дальнейшее развитие технологии Ethernet

В настоящее время самой распространенной сетевой технологией является именно Ethernet. По данным IDC, в 1997 году более 80 % всех сетей были построены на базе Ethernet. Все популярные операционные системы и стеки протоколов (TCP/IP, IPX, DECNet и т. д.) поддерживают Ethernet.

Причинами такого господства Ethernet в сетевом мире являются высокая надежность, доступность инструментов управления, масштабируемость, гибкость, низкая стоимость и легкость внедрения.

Технология Ethernet достаточно бурно эволюционировала с момента своего зарождения. В табл. 3.2 показана шкала эволюционного развития, представленная в формате nBASE-X (*n* — номинальная скорость передачи информации в Мбит/с, а X — среда передачи). В табл. 3.3 также приведена максимально допустимая длина кабеля.

Таблица 3.3.4.

Технологии и соответствующие скорости передачи

Тип	Скорость передачи	Длина
10BASE-5	10 Мбит/с, толстый коаксиал	500м
10BASE-2	10 Мбит/с, тонкий коаксиал	185м
10BASE-T	10 Мбит/с, неэкранированная витая пара	100м
10BASE-FL	10 Мбит/с, оптоволоконный кабель	2км
100BASE-TX	100 Мбит/с, неэкранированная витая пара (2 пары)	100м
100BASE-T4	100 Мбит/с, неэкранированная витая пара (4 пары)	100м
100BASE-FX	100 Мбит/с, оптоволоконный кабель	412 м/2 км
1000BASE-SX*	1000 Мбит/с (1 Гбит/с), многомодовый оптоволоконный кабель (62.5/125 мкм)	260м
1000BASE-SX	1000 Мбит/с (1 Гбит/с), многомодовый оптоволоконный кабель (50/125 мкм)	500м
1000BASE-LX	1000 Мбит/с (1 Гбит/с), многомодовый оптоволоконный кабель (62.5/125 мкм)	400м
1000BASE-LX	1000 Мбит/с (1 Гбит/с), многомодовый оптоволоконный кабель (50/125 мкм)	550м
1000BASE-LX	1000 Мбит/с (1 Гбит/с), одномодовый оптоволоконный кабель (9/126 мкм)	5000м
1000BASE-CX	1000 Мбит/с, экранированный сбалансированный медный кабель	25м

Протяженность кабеля для скоростей 1 Гбит/с приведена из текущего стандарта IEEE 802.3z, находящегося в стадии утверждения.

Изначально технология Ethernet была ограничена тем, что множество пользователей конкурировали за одну полосу пропускания в 10 Мбит/с. Однако со временем были найдены интересные решения, частично снимающие эту проблему. В их основе лежит использование коммутаторов, которые в отличие от традиционных мостов имеют большое количество портов и обеспечивают передачу кадров между несколькими портами одновременно. Это позволяет эффективно применять коммутаторы и для таких сетей, в которых трафик между сегментами практически не отличался от трафика, циркулирующего в самих сегментах. Технология Ethernet после появления коммутаторов перестала казаться совершенно бесперспективной, так как появилась возможность соединить низкую стоимость устройств Ethernet с высокой производительностью сетей, построенных на основе коммутаторов. Используя технологию коммутируемого Ethernet, каждое устройство получает выделенный канал между собой и портом коммутатора. Технология коммутации прижилась в сетях очень быстро. Обеспечивая передачу данных со скоростью канала связи между различными сегментами локальной сети (иными словами, между портами коммутатора), коммутация позволяет создавать крупные сети с эффективной системой управления. Кроме того, эта технология стала толчком к созданию концепции виртуальных локальных вычислительных сетей (ВЛВС).

Однако необходимость организации магистрали сети, к которой подключаются отдельные коммутаторы, не отпала. Если множество сегментов сети работают на скорости 10 Мбит/с, то магистраль должна иметь скорость значительно большую.

В начале 90-х годов начала ощущаться недостаточная пропускная способность Ethernet. Для компьютеров на процессорах Intel 80286 или 80386 с шинами ISA (8 Мбайт/с) или EISA (32 Мбайт/с) пропускная способность сегмента Ethernet составляла 1/8 или 1/32 часть канала «память-диск» и хорошо согласовывалась с соотношением между объемом локальных и внешних данных, циркулирующих в компьютере. Теперь же у мощных клиентских станций с процессорами Pentium или Pentium Pro и шиной PCI (133 Мбайт/с) эта доля упала до 1/133, что явно недостаточно. Поэтому многие сегментыEthernet на 10 Мбит/с стали перегруженными, время реакции серверов и частота возникновения коллизий в таких сегментах значительно возросли, еще более снижая реальную пропускную способность. В ответ на эти требования была разработана технология Fast Ethernet, являющаяся 100-мегабитной версией Ethernet.

Следует отметить, что увеличение скорости в 10 раз приводит к уменьшению максимального расстояния между узлами. Сначала было предложено простое решение задачи построения магистрали — несколько коммутаторов Ethernet связывались вместе по витой паре или волоконно-оптическому кабелю — так называемая коллапсированная магистраль. Но возникла проблема, когда потребовалось связать коммутаторы, находящиеся на больших расстояниях. Она была решена с помощью организации выделенного, свободного от коллизий оптоволоконного канала связи. В этом случае коммутаторы могли связываться напрямую на расстояния до 2 км. Как видно, технология Fast Ethernet обеспечила достаточно всеобъемлющее решение для построения сетей масштаба одного или нескольких зданий. Одобрение стандарта на технологию Fast Ethernet в 1995 году стало важным событием для сообщества производителей сетевого оборудования, так как появилась гибкая, быстрая и масштабируемая технология передачи данных.

До разработки технологий коммутации и Fast Ethernet среди специалистов по сетевым технологиям господствовало мнение, что технологии ATM и FDDI будут оптимальным решением для организации магистрали сети. Однако в настоящее время технология Fast Ethernet часто конкурирует с упомянутыми технологиями в этой области. Кроме того, активно разрабатывается и внедряется технология GigabitEthernet.

Fast Ethernet

Идея технологии Fast Ethernet родилась в 1992 году. В августе следующего года группа производителей объединилась в организацию, названную Альянсом Fast Ethernet (Fast Ethernet Alliance — FEA). Цель этого альянса заключалась в скорейшем одобрении стандарта Fast Ethernet комитетом IEEE. В июне 1995 года все процедуры стандартизации были успешно завершены, и технология Fast Ethernetбыла стандартизирована в документе 802.3и.

При рассмотрении стандарта много времени уделялось сохранению метода доступа CSMA/CD. Все предложенные решения опирались на этот метод, что вполне естественно, так как он позволяет сохранить преемственность с сетями l0Base-T и l00Base-T. CSMA/CD определяет способ передачи данных по сети от одного узла к другому через кабельную систему. В модели OSI протокол CSMA/CD является частью уровня управления доступом к среде (Media Access Control, MAC). На этом уровне определяется формат, в котором информация передается по сети, и способ получения доступа сетевого устройства к сети для передачи данных. Компании HP и AT&T предложили совершенно отличный от CSMA/CD метод доступа, который был назван Demand Priority. Однако он был поддержан гораздо меньшим числом сетевых производителей. Для его стандартизации был организован новый комитет IEEE 802.12.

Стандарт Fast Ethernet определяет три модификации для работы с разными видами кабелей: 100BaseTX, 100BaseT4 и 100BaseFX. Модификации 100BaseTX и 100BaseT4 рассчитаны на витую пару, а 100BaseFX был разработан для оптического кабеля.

Стандарт 100BaseTX требует применения двух пар неэкранированных или экранированных витых пар. Одна пара служит для передачи, другая — для приема. Этим требованиям отвечают два основных кабельных стандарта: на неэкранированную витую пару категории 5 и экранированную витую пару типа 1 от IBM.

Стандарт 100BaseT4 имеет менее ограничительные требования к кабелю, так как в нем задействуются все четыре пары восьмижильного кабеля: одна пара для передачи, другая для приема, а оставшиеся две пары работают как на передачу, так и на прием. В результате в стандарте 100BaseT4 и прием, и передача данных могут осуществляться по трем парам. Для реализации сетей 100BaseT4 подойдут кабели с неэкранированной витой парой категорий 3-5 и экранированный типа 1.

Технология Fast Ethernet включает в себя также стандарт для работы с многомодовым оптоволоконным кабелем. Этот стандарт (100BaseFX) ориентирован, в основном, на применение в магистрали сети или для организации связи удаленных объектов.

Преемственность технологий Fast Ethernet и Ethernet позволяет легко выработать рекомендации по применению: Fast Ethernet целесообразно применять в тех организациях, которые широко использовали классический Ethernet, но сегодня испытывают потребность в увеличении пропускной способности. При этом сохраняется весь накопленный опыт работы с Ethernet и, частично, сетевая инфраструктура.

Хотя Fast Ethernet и является развитием стандарта Ethernet, переход к 100BaseT требует некоторого изменения в топологии сети. Теоретический предел диаметра сегмента сети Fast Ethernet составляет 250 м. Это ограничение определено самой природой метода доступа CSMA/CD и скоростью передачи в 100 Мбит/с.

Для классического Ethernet время прослушивания сети определяется максимальным расстоянием, которое 512-битный кадр может пройти по сети за время, равное времени обработки этого кадра на рабочей станции. Для сети Ethernet это расстояние равно 2500 м. В сети Fast Ethernet этот же самый 512-битный кадр за время, необходимое на его обработку рабочей станцией, пройдет всего 250 м. Если принимающая станция будет удалена от передающей на расстояние свыше 250 м, то кадр может вступить в конфликт с другим кадром на линии, а передающая станция, завершив передачу, уже опоздала бы с реакцией на этот конфликт. Поэтому максимальный диаметр сети 100BaseT составляет 250 м.

Для увеличения допустимой дистанции необходимо использовать два повторителя для соединения всех узлов. В соответствии со стандартом Fast Ethernet расстояние между концентратором и рабочей станцией не должно превышать 100 м. Для установки Fast Ethernet потребуются сетевые адаптеры для рабочих станций и серверов, концентраторы 100BaseT и, возможно, некоторое количество коммутаторов 100BaseT. К моменту появления стандарта Fast Ethernet в построении локальных сетей масштаба здания сложился следующий подход — магистраль крупной сети строилась на технологииFDDI (высокоскоростной и отказоустойчивой, но весьма дорогой), а сети рабочих групп и отделов использовали Ethernet или Token Ring.

Основная область использования Fast Ethernet сегодня — это сети рабочих групп и отделов. Целесообразно совершать переход к Fast Ethernet постепенно, оставляя Ethernet там, где он хорошо справляется с поставленными задачами. Одним из очевидных случаев, когда Ethernet не следует заменять технологией Fast Ethernet, является подключение к сети старых персональных компьютеров с шиной ISA.

3.3 Схема протокола логического контроля соединения (LLC)

Стандарт на технологии OSI определяет канальный и физический уровни передачи данных. Канальный уровень ЛВС разбит на два подуровня: логического контроля соединения (LLC — Logical LinkControl) и контроля доступа к среде (MAC — Medium Access Control). На канальный уровень возлагается ответственность за обеспечение надежной передачи данных между двумя узлами сети. Получая пакет для передачи с более высокого сетевого уровня, канальный уровень присоединяет к этому пакету адреса получателя и отправителя, формирует из него набор кадров для передачи и обеспечивает выявление и исправление ошибок.

Нижний подуровень канального уровня — контроль доступа к среде — при передаче окончательно формирует кадр передачи в соответствии с тем протоколом, который реализован в данном сегменте (IEEE802.3 или 802.5). При получении пакета этот подуровень проверяет адрес, контрольную сумму и наличие ошибок при передаче.

Верхний подуровень канального уровня — логический контроль соединения — обеспечивает режимы передачи данных как с установлением, так и без установления соединения.

Рассмотрим работу протокола LLC на примере передачи данных в ЛВС, содержащих станции A,B ,C (см. рис. 3.4) соединенных каналом связи и использующим технологию Ethernet. В примере требуется доставить кадр от станции A к станции B.

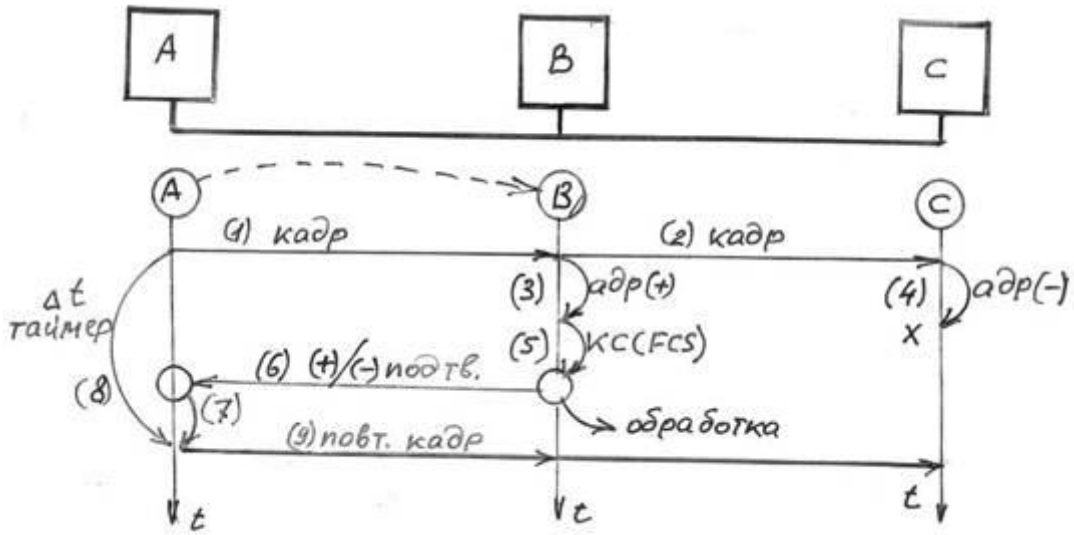


Рис 3.4 Схема протокола LLC

Станция A формирует и отправляет кадр к станции B (1), который по единой среде канала связи достигает станции C. Одновременно на станции A взводится таймер контроля длительности подтверждения приема. На станции B (3) при проверке адреса получателя устанавливается, что кадр доставлен правильно. На станции C (4) из-за несовпадения адреса назначения кадр удаляется. На станции B (5) проверяется контрольная сумма и, если контрольная сумма совпадает, то формируется (6) положительный кадр (+) подтверждения и кадр поступает на *обработку,* если контрольная сумма не совпадает, то формируется (6) отрицательный кадр (-) подтверждения и кадр удаляется. На станции A при поступлении отрицательного кадра (-) и при срабатывании таймера (8) осуществляется повторная передача (9) кадра.

4. ПРОТОКОЛ ARP И RARP

4.1 Протокол ARP

Протокол ARP позволяет решить задачу: «Как на стороне *отправителя* по известному IP-адресу *получателя* определить MAC-адрес ближайшего коммуникационного узла, которому известен маршрут к получателю?»

Протокол ARP (Address Resolution Protocol, протокол разрешения адреса) описан в документе RFC 826. Необходимость протокола ARP продиктована тем обстоятельством, что IP-адреса устройств в сети назначаются независимо от их физических адресов. Поэтому для доставки сообщений по сети необходимо определить соответствие между физическим адресом устройства и его IP-адресом — это называется *разрешением адресов*. В большинстве случаев прикладные программы используют именно IP-адреса. А так как схемы физической адресации устройств весьма разнообразны, то необходим специальный, универсальный протокол. Разрешение адресов может быть произведено двумя способами: с помощью прямого отображения и с помощью динамического связывания. Протокол ARP использует механизм динамического связывания.

Функционально протокол ARP состоит из двух частей. Одна часть протокола определяет физические адреса при посылке дейтаграммы, другая отвечает на запросы от других устройств в сети. Протокол ARP предполагает, что каждое устройство знает как свой IP-адрес, так и свой физический адрес.

Для того чтобы уменьшить количество посылаемых запросов ARP, каждое устройство в сети, использующее протокол ARP, должно иметь специальную буферную память. В этой памяти хранятся пары (IP-адрес, физический адрес) устройств в сети. Всякий раз, когда устройство получает ARP-ответ, оно сохраняет в этой памяти соответствующую пару. Если адрес есть в списке пар, то нет необходимости посылать APP - запрос. Эта буферная память называется ARP-*таблицей*.

4.2. ARP-таблица для преобразования адресов

В ARP-таблице могут быть как статические, так и динамические записи. Динамические записи добавляются» и удаляются автоматически. Статические записи могут быть добавлены пользователем. Кроме того, ARP-таблица всегда содержит запись с физическим широковещательным адресом (%FFFFFFFFFFF) для локальной сети. Эта запись позволяет устройству принимать широковещательные ARP-запросы.

Каждая запись в ARP-таблице имеет свое время жизни — обычно оно составляет 10 мин. После того как записи была добавлена в таблицу, ей присваивается таймер. Если запись не используется в первые две минуты, она удаляется. Если используется — время ее жизни (доставляет 10 мин. В некоторых реализациях протокола ARP новый таймер устанавливается после каждого использования записи в ARP-таблице. На рис.4.1 показан пример ARP-таблицы, сформированной на компьютере с двумя сетевыми интерфейсами, работающим под управлением операционной системы Micrcosoft Windows NT. Это таблица выводится по команде **arp -a**.

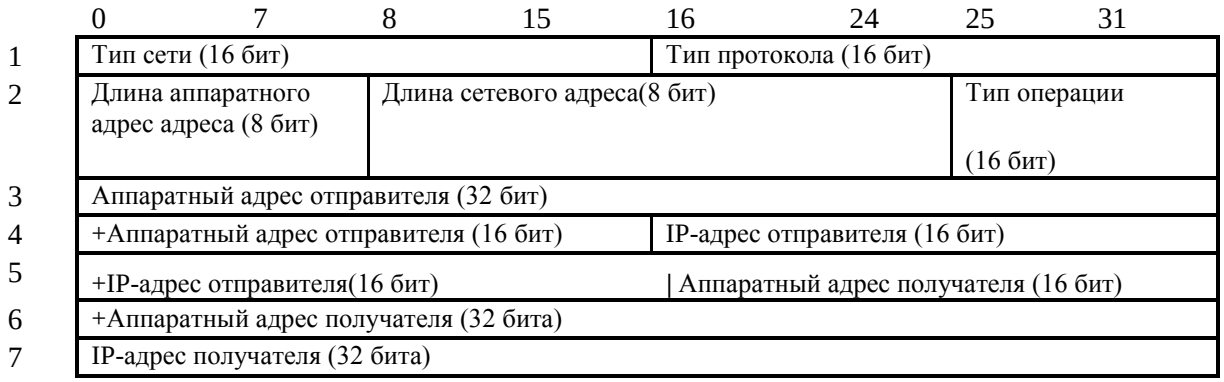
Сообщения протокола ARP (при передаче по сети инкапсулируются в поле данных кадра. Они не содержат IP-заголовка. В отличие от большинства протоколов сообщения, ARP не имеют фиксированного формата заголовка. Протокол ARP был разработан таким образом, чтобы его можно было использовать для разрешения адресов в различных сетях. Фактически протокол можно использовать с произвольными физическими адресами и сетевыми протоколами.

```
C:\arp –a

Interface: 172. 16. 112. 123

  Internet Address      Physical  Address      Type

  172. 16. 112. 123     00-00-0c-1a-ab-c5     dynamic

  72. 16. 112. 124      00-dd-01-07-57-15     dynamic

Interface: 172. 16. 113. 190

  Internet Address      Physical  Address      Type

  172. 16. 113. 138     00-00-0c-1a-ab-c5     dynamic
```

Рис. 4.1 Пример ARP-таблицы, сформированной на компьютере

На рис.4.1 показан формат сообщения ARP. В отличие от большинства протоколов, поля переменной длины в сообщениях ARP не выровнены по 32-битовой границе, что вносит определенные трудности в изучение протокола. Например, аппаратный адрес отправителя занимает 6 байт, поэтому на рис. 4.1 он занимает две строки (одна строка — одно двойное слово).

В поле **«Тип сети»** для сетей Ethernet указывается 1. Для других типов сетей значение этого поля определено соответствующими документами RFC. Поле **«Тип протокола»** позволяет использовать сообщения ARP не только для протокола IP, но и для других сетевых, протоколов.



4.3 Формат сообщения ARP

Для протокола IP значение этого поля равно 0800 (шестнадцатеричное). В поле **«Тип операции»** для ARP-запросов указывается 1, а для ARP-ответов указывается 2. Поля **«Длина аппаратного адреса»** и**«Длина сетевого адреса»** позволяют использовать протокол ARP в любых сетях, так как длину адресов можно задать. Протокол ARP работает по следующей схеме. Устройство, отправляющее ARP-запрос, заполняет в сообщении все поля, кроме искомого аппаратного адреса. Затем оно рассылает запросы по всей подсети. Поле заполняется устройством, опознавшим свой IP-адрес.

Преобразование адресов выполняется путем поиска в таблице. Эта таблица, называемая ARP-таблицей, хранится в памяти и содержит строки для каждого узла сети. В двух столбцах содержатся IP- и Ethernet-адреса.

Если требуется преобразовать IP-адрес в Ethernet-адрес, то ищется запись с соответствующим IP-адресом. Ниже приведен пример упрощенной ARP-таблицы.

Табл.4.1.

Пример ARP-таблицы

IP-адрес	Ethernet-адрес (MAC)
223.1.2.1	08:00:39:00:2F:C3
223.1.2.3	08:00:5A:21:A7:22

Принято все байты 4-байтного IP-адреса записывать десятичными числами, разделенными точками. При записи 6-байтного Ethernet-адреса каждый байт указывается в 16-ричной системе и отделяется двоеточием.

ARP-таблица необходима потому, что IP-адреса и Ethernet-адреса выбираются независимо, и нет какого-либо алгоритма для преобразования одного в другой. IP-адрес выбирает менеджер сети с учетом положения машины в сети Internet. Если машину перемещают в другую часть сети Internet, то ее IP-адрес должен быть изменен. Ethernet-адрес выбирает производитель сетевого интерфейсного оборудования из

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

4.4. Порядок преобразования адресов в ARP- таблице

Рассмотрим работу протокола ARP на примере передачи данных в ЛВС, содержащих станции A, B, C (см. рис. 4.2) соединенных каналом связи и использующим технологию Ethernet.

В примере ARP- таблица станции A имеет вид , представленный в табл. 4.1

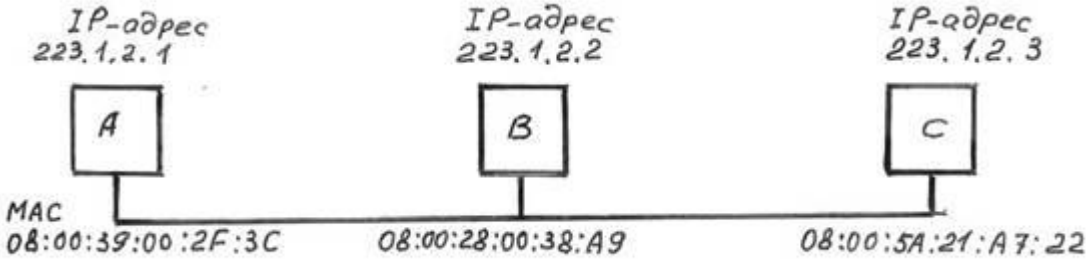


Рис.4.2 Схема фрагмента сети (пример)

Рассмотрим процедуру выполнения протокола ARP станцией A (см. рис.4.3) для ситуации, когда ей известен IP-адрес станции B, но не известен MAC-адрес станции B.

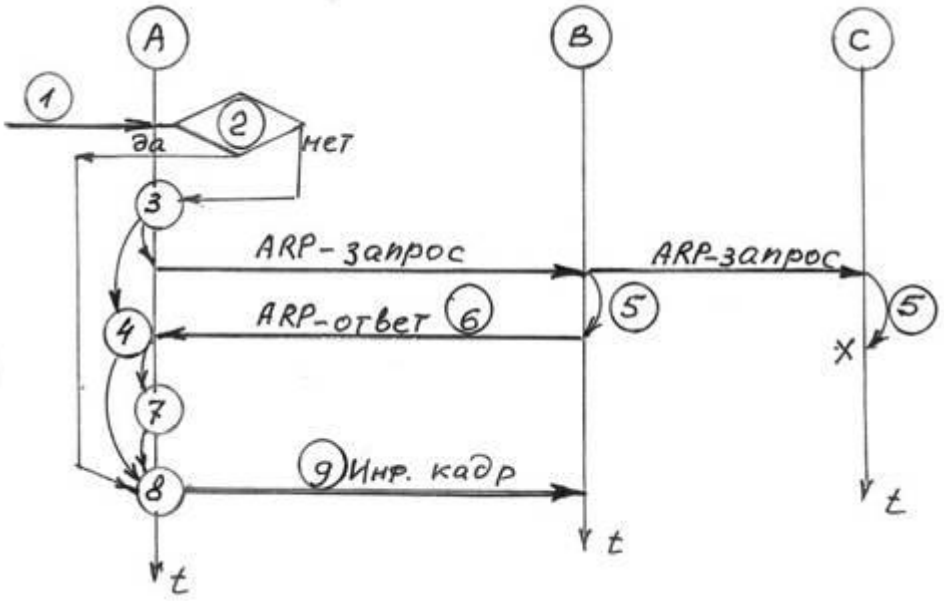


Рис.4.3 Процедура выполнения протокола ARP станцией A

1. В ходе обычной работы сетевая программа, например такая как TELNET, отправляет прикладное сообщение, пользуясь транспортными услугами TCP. Модуль TCP посылает соответствующее транспортное сообщение через модуль IP. В результате составляется IP-пакет, который должен быть передан драйверу Ethernet станции A. IP-адрес (221.1.2.2) места назначения (станция B) известен прикладной программе, модулю TCP и модулю IP. Необходимо на его основе найти (MAC)Ethernet-адрес места назначения (станции B).

2. Для определения искомого Ethernet-адреса используется ARP-таблица. Она заполняется автоматически модулем ARP, по мере необходимости.

Проверятся: «есть ли в ARP-таблице запись места назначения - (MAC)Ethernet-адрес соответствующий требуемому IP-адресу». Если есть, то переходим к п.9. Если нет, то переходим к п.3.

3. По сети передается широковещательный ARP-запрос в форме кадра Ethernet (см. рис. 3.1). В заголовке кадра:

- в поле **Адрес получателя** заносится FF:FF:FF:FF:FF:FF- признак широковещательного запроса;

- в поле **Адрес отправителя** заносится 08:00:39:00:2F:C3 - MAC-адрес станции A.

В поле **Данные** кадра Ethernet записываются данные в формате сообщения ARP (см. рис. 4.2) и для рассматриваемого примера имеющие следующие значения (см. рис. 4.4)

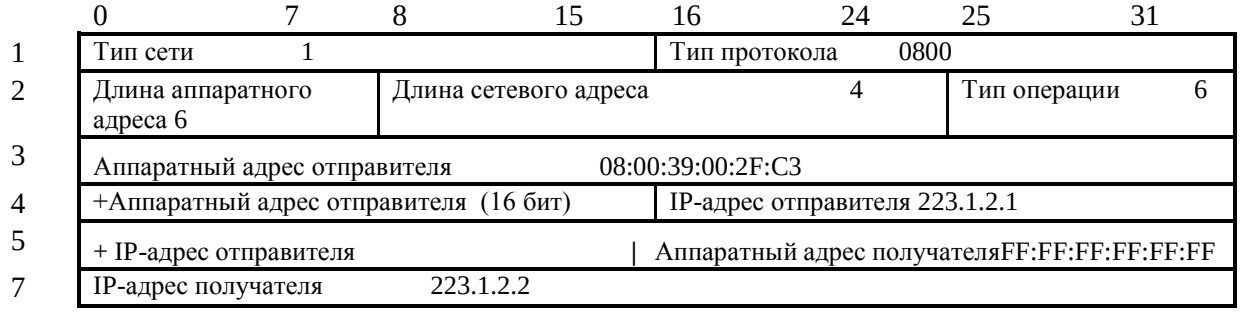


Рис. 4.4 Формат ARP - запроса для рассматриваемого примера

4. Исходящий IP-пакет ставится в очередь.

5 Каждый сетевой адаптер принимает широковещательные передачи. Все драйверы Ethernet проверяют поле типа в принятом Ethernet-кадре и передают ARP-пакеты модулю ARP. ARP-запрос можно интерпретировать так: "Если ваш IP-адрес совпадает с указанным, то сообщите мне ваш Ethernet-адрес".

6. Каждый модуль ARP проверяет поле искомого IP-адреса получателя в полученном ARP-пакете и, если адрес совпадает с его собственным IP-адресом, то посылает ответ прямо по Ethernet-адресу отправителя запроса.

В заголовке кадра:

- в поле **Адрес получателя** заносится 08:00:39:00:2F:C3 - MAC-адрес станции A;

- в поле **Адрес отправителя** заносится 08:00:28:00:38:A9 - MAC-адрес станции B.

В поле **Данные** кадра Ethernet записываются данные в формате сообщения ARP (см. рис. 4.2) и для рассматриваемого примера имеющие следующие значения (см. рис. 4.5)

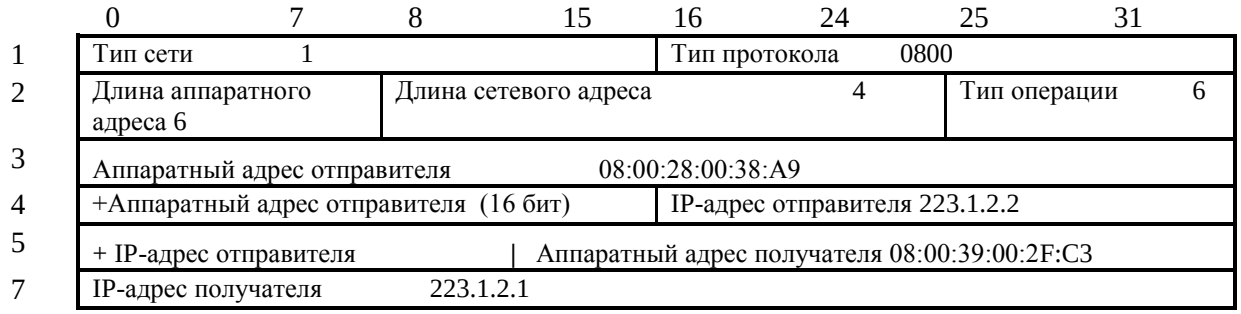


Рис. 4.5 Формат ARP - ответа для рассматриваемого примера

ARP-ответ можно интерпретировать так: "Да, это мой IP-адрес, ему соответствует такой-то Ethernet-адрес". Пакет с ARP-ответом выглядит примерно так:

7. Этот ответ получает станция A, сделавшая ARP-запрос. Драйвер этой станции проверяет поле типа в Ethernet-кадре и передает ARP-пакет модулю ARP. Модуль ARP анализирует ARP-пакет и добавляет запись в свою ARP-таблицу.

Обновленная таблица выглядит следующим образом:

Табл.4.4

ARP-таблица после обработки ответа

IP-адрес	Ethernet-адрес (MAC)
223.1.2.1	08:00:39:00:2F:C3
223.1.2.2	08:00:28:00:38:A9
223.1.2.3	08:00:5A:21:A7:22

Новая запись в ARP-таблице появляется автоматически, спустя несколько миллисекунд после того, как она потребовалась.

8. Для поставленного на шаге 3 в очередь исходящего IP-пакета выполняется с использованием обновленной ARP-таблицы преобразование IP- адреса в Ethernet-адрес.

9. Сформированный Ethernet-кадр передается по сети.

Если с помощью ARP-таблицы не удается сразу осуществить преобразование адресов, то IP-пакет ставится в очередь, а необходимая для преобразования информация получается с помощью запросов и ответов протокола ARP, после чего IP-пакет передается по назначению.

Если в сети нет машины с искомым IP-адресом, то ARP-ответа не будет и не будет записи в ARP-таблице. Протокол IP будет уничтожать IP-пакеты, направляемые по этому адресу. Протоколы верхнего уровня не могут отличить случай повреждения сети Ethernet от случая отсутствия машины с искомым IP-адресом.

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

Следует подчеркнуть, что каждая станция имеет собственную ARP-таблицу для каждого своего сетевого интерфейса.

4.5 Протокол RARP

Протокол RARP (Reverse Address Resolution Protocol — Обратный протокол преобразования адресов) — протокол сетевого уровня модели OSI, выполняет обратное отображение адресов, то есть для любого отправителя преобразует его физический MAC-адрес в IP-адрес. RARP является дополнением к ARP, и описан в RFC 903.

Протокол применяется во время загрузки узла (например компьютера - «тонкого клиента»), когда он посылает групповое сообщение-запрос со своим физическим MAC-адресом.

Под термином «тонкий клиент» подразумевается достаточно широкий с точки зрения системной архитектуры ряд устройств и программ, которые объединяются общим свойством: возможность работы в терминальном режиме. Таким образом, для работы тонкого клиента необходим терминальный сервер. Этим тонкий клиент отличается от толстого клиента, который, напротив, производит обработку информации независимо от сервера, используя последний в основном лишь для хранения данных.

Кроме общего случая, следует выделить аппаратный тонкий клиент (например, Windows- и Linux-терминалы) — специализированное устройство, принципиально отличное от ПК. Аппаратный тонкий клиент не имеет жёсткого диска, использует специализированную локальную ОС (одна из задач которой организовать сессию с терминальным сервером для работы пользователя), не имеет в своём составе подвижных деталей, выполняется в специализированных корпусах с полностью пассивным охлаждением.

Для расширения функциональности тонкого клиента прибегают к его «утолщению», например, добавляют возможности автономной работы, сохраняя главное отличие — работу в сессии с терминальным сервером. Когда в клиенте появляются подвижные детали (жёсткие диски), появляются возможности автономной работы, он перестаёт быть тонким клиентом в чистом виде, а становится универсальным клиентом.

Тонкий клиент в большинстве случаев обладает минимальной аппаратной конфигурацией, вместо жёсткого диска для загрузки локальной специализированной ОС используется DOM (DiskOnModule) [модуль с разъёмом IDE, флэш-памятью и микросхемой, реализующей логику обычного жёсткого диска — в BIOS определяется как обычный жёсткий диск, только размер его обычно в 2-3 раза меньше]. В некоторых конфигурациях системы тонкий клиент загружает операционную систему по сети с сервера, используя протоколы PXE, BOOTP, DHCP, TFTP и Remote Installation Services (RIS).

Сервер принимает отправленное тонким клиентом сообщение и просматривает свои таблицы (либо перенаправляет запрос куда-либо ещё) в поисках IP-адреса, соответствующего физическому MAC-адресу тонкого клиента.

После обнаружения найденный адрес отсылается обратно на запросивший его узел. Другие станции также могут «слышать» этот диалог и локально сохранить эту информацию в своих ARP-таблицах.

RARP позволяет разделять IP-адреса между не часто используемыми хост-узлами. После использования каким-либо узлом IP-адреса он может быть освобождён и выдан другому узлу.

Следует подчеркнуть, что RARP отличается от «обратного» ARP (Inverse Address Resolution Protocol, или InARP), описанного в RFC 2390, который предназначен для получения IP-адреса, соответствующего MAC-адресу другого узла. InARP является дополнением к протоколу разрешения адресов и используется для обратного поиска. RARP является скорее аналогом DHCP/BOOTP.

RARP: обратный протокол определения адреса

[TCP/IP КРУПНЫМ ПЛАНОМ http://www.soslan.ru/tcp/tcp00.html]

Когда загружается система с локальным диском, она обычно получает свой IP адрес из конфигурационного файла, который считывается с диска. Однако для систем, не имеющих диска, таких как X терминалы или бездисковые рабочие станции, требуются другой способ определения собственного IP адреса.

Каждая система в сети имеет уникальный аппаратный адрес, который назначается производителем сетевого интерфейса (сетевой платы). Принцип работы RARP заключается в том, что бездисковая система может считать свой уникальный аппаратный адрес с интерфейсной платы и послать RARP запрос (широковещательный фрейм в сеть), где потребует кого-нибудь откликнуться и сообщить IP адрес (с помощью RARP отклика).

Несмотря на то что концепция довольно проста, ее реализация как правило значительно сложнее чем ARP, который был описан в предыдущем разделе.

Формат пакета RARP практически идентичен пакету ARP (см. рисунок 4.2). Единственное отличие заключается в том, что поле тип фрейма (frame type) для запроса или отклика RARP установлено в 0x8035, а поле op имеет значение 3 для RARP-запроса и значение 4 для RARP-ответа.

RARP-запрос является широковещательным, а RARP-ответ обычно персональный.

Примеры RARP

В нашей сети мы можем заставить хост sun загружаться из сети, вместо того чтобы загружаться с локального диска. Если мы запустили RARP сервер и tcpdump на хосте bsdi, то получим вывод, показанный на рисунке 4.6. Мы используем флаг -e, чтобы команда tcpdump показывала аппаратные адреса:

```
1  0.0              8:0:20:3:f6:42 ff:ff:ff:ff:ff:ff rarp 60:
rarp who-is 8:0:20:3:f6:42 tell 8:0:20:3:f6:42
2  0.13 (0.13)      0:0:c0:6f:2d:40 8:0:20:3:f6:42 rarp 42:
rarp reply 8:0:20:3:f6:42 at sun
```

```
3  0.14 (0.01)      8:0:20:3:f6:42 0:0:c0:6f:2d:40 ip 65:
sun.26999>bsdi.tftp: 23 RRQ "8CFC0D21.SUN4C"
```

Рис. 4.6 Запрос и отклик RARP.

Запрос RARP широковещательный (строка 1), а отклик RARP (строка 2) персональный. Вывод в строке 2, at sun, означает, что RARP отклик содержит IP адрес хоста sun (140.252.13.33).

В строке 3 мы видим, что как только sun получил свой IP адрес, он выдал TFTP запрос на чтение (RRQ) файла 8CFC0D21.SUN4C. (TFTP это простой протокол передачи файлов - Trivial File Transfer Protocol. Мы рассмотрим его более подробно в главе 15.) Восемь шестнадцатиричных цифр в имени файла это шестнадцатиричное представление IP адреса 140.252.13.33 хоста sun. Это IP адрес, который мы получили в отклике RARP. Оставшаяся часть имени файла, SUN4C, указывает на тип системы, которая загружается.

В строке 3 tcpdump сообщает, что это IP датаграмма длиной 65, а не UDP датаграмма (которая в действительности является ей), так как мы запустили tcpdump с флаго -e, чтобы получить в выводе аппаратные адреса. Еще один момент на, который необходимо обратить внимание на рисунке 4.6, заключается в том, что длина Ethernet фрейма в строке 2 меньше чем установленный минимум (который, как мы упоминали в разделе "Примеры ARP" главы 4, должен быть равен 60 байтам). Причина этого заключается в том, что мы запустили tcpdump на системе, которая посылает этот Ethernet фрейм (bsdi). Приложение, rarpd, пишет 42 байта в устройство пакетного фильтра BSD (BSD Packet Filter) (14 байт - Ethernet заголовок и 28 байт - отклик RARP), именно это и видит tcpdump. Однако драйвер устройства Ethernet дополняет этот короткий фрейм до минимального размера, необходимого для передачи (60 байт). Если бы мы запустили tcpdump на другом компьютере, длина составила бы 60 байт.

Также мы видим, что когда бездисковая система получает свой IP адрес в отклике RARP, она осуществляет TFTP запрос, чтобы прочитать загрузочный имидж. Сейчас мы не будем подробно рассматривать, как загружаются бездисковые системы. (В главе 16 описывается последовательность загрузки бездисковых X терминалов с использованием RARP, BOOTP и TFTP.)

На рисунке 4.7 показаны пакеты, которые появляются в том случае, если в сети нет RARP сервера. Адрес назначения каждого пакета - широковещательный адрес Ethernet. Адрес Ethernet, следующий за who-is, это аппаратный адрес хоста которому требуется информация, а адрес Ethernet, следующий за tell, это аппаратный адрес отправителя.

Обратите внимание на частоту повторных передач. Первая повторная передача происходит через 6,55 секунд, затем интервал увеличивается до 42,80 секунд, затем через 5,34 секунды, затем через 6,55 секунд и снова через 42,79 секунды. Если мы рассчитаем разницу между каждым тайм-аутом, мы заметим эффект удвоения: между 5,34 и 6,55 - 1,21 секунды, между 6,55 и 8,97 - 2,42 секунды, между 8,97 и 13,80 - 4,83 секунды и так далее.

```
1    0.0            8:0:20:3:f6:42 ff:ff:ff:ff:ff:ff rarp 60:
                    rarp who-is 8:0:20:3:f6:42 tell 8:0:20:3:f6:42
2    6.55 ( 6.55)   8:0:20:3:f6:42 ff:ff:ff:ff:ff:ff rarp 60:
                    rarp who-is 8:0:20:3:f6:42 tell 8:0:20:3:f6:42
3   15.52 ( 8.97)   8:0:20:3:f6:42 ff:ff:ff:ff:ff:ff rarp 60:
                    rarp who-is 8:0:20:3:f6:42 tell 8:0:20:3:f6:42
```

```
4   29.32 (13.80)    8:0:20:3:f6:42 ff:ff:ff:ff:ff:ff rarp 60:
                     rarp who-is 8:0:20:3:f6:42 tell 8:0:20:3:f6:42
5   52.78 (23.46)    8:0:20:3:f6:42 ff:ff:ff:ff:ff:ff rarp 60:
                     rarp who-is 8:0:20:3:f6:42 tell 8:0:20:3:f6:42
6   95.58 (42.80)    8:0:20:3:f6:42 ff:ff:ff:ff:ff:ff rarp 60:
                     rarp who-is 8:0:20:3:f6:42 tell 8:0:20:3:f6:42
7  100.92 ( 5.34)    8:0:20:3:f6:42 ff:ff:ff:ff:ff:ff rarp 60:
                     rarp who-is 8:0:20:3:f6:42 tell 8:0:20:3:f6:42
8  107.47 ( 6.55)    8:0:20:3:f6:42 ff:ff:ff:ff:ff:ff rarp 60:
                     rarp who-is 8:0:20:3:f6:42 tell 8:0:20:3:f6:42
9  116.44 ( 8.97)    8:0:20:3:f6:42 ff:ff:ff:ff:ff:ff rarp 60:
                     rarp who-is 8:0:20:3:f6:42 tell 8:0:20:3:f6:42
10 130.24 (13.80)    8:0:20:3:f6:42 ff:ff:ff:ff:ff:ff rarp 60:
                     rarp who-is 8:0:20:3:f6:42 tell 8:0:20:3:f6:42
11 153.70 (23.46)    8:0:20:3:f6:42 ff:ff:ff:ff:ff:ff rarp 60:
                     rarp who-is 8:0:20:3:f6:42 tell 8:0:20:3:f6:42
12 196.49 (42.79)    8:0:20:3:f6:42 ff:ff:ff:ff:ff:ff rarp 60:
                     rarp who-is 8:0:20:3:f6:42 tell 8:0:20:3:f6:42
```

Рис. 4.7 RARP запрос при отсутствии в сети RARP сервера.

Когда тайм-аут достигает определенного предела (больше чем 42,80 секунды), он сбрасывается вновь в 5,34 секунды.

Подобное увеличение значения тайм-аута - это наилучший подход, так как каждый раз используется одно и то же значение. На рисунке 4.8 мы увидим неверный подход, с помощью которого осуществляются тайм-ауты и повторные передачи, а в главе 21 мы рассмотрим метод используемый в TCP.

Реализация RARP сервера

Тогда как концепция RARP довольно проста, реализация RARP сервера сильно зависит от системы и довольно сложна. Напротив, реализация ARP сервера проста и является, как правило, частью реализации ядра TCP/IP. Так как ядро знает собственные IP адреса и аппаратные адреса, при получении ARP запроса для одного из IP адресов оно просто формирует отклик с соответствующим аппаратным адресом.

RARP серверы как пользовательские процессы

Основная задача RARP сервера заключается в том, чтобы предоставить соответствие между аппаратными адресами и IP адресами для множества хостов (все бездисковые системы в сети). Необходимая информация содержится в дисковом файле (обычно /etc/ethers в UNIX системах). Так как ядро обычно не читает дисковые файлы, функция RARP сервера реализуется с использованием пользовательского процесса, который не является частью ядра TCP/IP.

Далее, можно отметить, что RARP запросы передаются в качестве Ethernet фреймов со специфическим полем типа фрейма Ethernet. Это означает, что RARP сервер должен обладать способностью отправлять и принимать Ethernet фреймы подобного типа. В приложении A мы опишем как для приема подобных фреймов используются BSD Packet Filter, Sun Network Interface Tap и SVR4 Data Link Provider Interface. Так

как посылка и прием подобных фреймов зависит от системы, реализация RARP сервера также зависит от системы.

Несколько RARP серверов в сети

Еще одна особенность заключается в том, что RARP запросы посылаются в виде широковещательных запросов аппаратного уровня, как показано на рисунке 4.7. Это означает, что они не перенаправляются маршрутизаторами. Чтобы позволить бездисковым системам загружаться, даже если RARP сервер выключен, в сети обычно существуют несколько RARP серверов (на одном и том же кабеле).

По мере того как количество серверов растет (чтобы повысить надежность), увеличивается сетевой трафик, так как каждый сервер посылает RARP отклик на каждый RARP запрос. Бездисковые системы, которые посылают RARP запросы, обычно используют первый полученный ими RARP отклик. (Мы никогда не имели подобных проблем с ARP, потому что только один хост посылает ARP отклик.) Более того, существует вероятность, что несколько RARP серверов отправят отклики одновременно, увеличивая тем самым количество коллизий в Ethernet.

Краткие выводы

RARP используется большинством бездисковых систем при загрузке, для получения свох IP адресов. Формат пакета RARP практически идентичен пакету ARP. Запрос RARP широковещательный, в нем содержится аппаратный адрес отправителя, при этом он спрашивает кого-либо послать ему его IP адрес. Отклик обычно персональный.

Проблемы с RARP заключаются в том, что он использует широковещательные запросы на канальном уровне, поэтому большинство маршрутизаторов не могут перенаправлять RARP запросы; а также в том, что передается минимум необходимой информации: только IP адрес системы. В главе 16 мы увидим, что BOOTP сообщает значительно больше информации необходимой при загрузке бездисковых систем: IP адрес, имя хоста, с которого происходит загрузка, и так далее.
Несмотря на то что концепция RARP довольно проста, реализация RARP сервера зависит от системы. Также надо отметить, что не все TCP/IP реализации предоставляют RARP сервер.

Примеры RARP
http://paramax.susu.ru/study/tcpips/-Protokol_obratnogo_otobrazheniya_adresov_RARP-Primery_RARP.htm
В нашей локальной сети мы можем загружать систему на хосте sun не только с его диска, но и способом удаленной загрузки по сети. Если мы одновременно запустим RARP-сервер и tcpdump на хосте bsdi, то получим распечатку, показанную на рис. 4.6. Для того чтобы выводились аппаратные адреса, мы задаем ключ -e. В строке виден широковещательный RARP-запрос, а в строке 2 — направленный конкретному адресату RARP-ответ. Запись at sun в конце строки 2 означает, что RARP-ответ содержит IP-адрес хоста sun (140.252.13.33).

```
1       0.0     8:0:20:3:f6:42   ff:ff:ff:ff:ff:ff  rarp  60:
rarp who-is 8:0:20:3:f6:42 tell 8:0:20:3:f6:42
2       0.13    (0.13) 0:0:c0:6f:2d:40 8:0:20:3:f6:42 rarp 42:
rarp reply 8:0:20:3:f6:42 at sun
3       0.14 (0.01) 8:0:20:3:f6:42 0:0:cO:6f:2d:40 ip 65:
```

```
sun.26999 > bsdi.tftp:   23  RRQ  "8CFCOD21.SUN4C"
```

Рис. 4.8 5.1. RARP-запрос и RARP-ответ

Получив свой IP-адрес, sun посылает TFTP-запрос RRQ. (Read Request) на чтение файла 8CFCOD21. SUN4C. Восемь символов в имени файла соответствуют шестнадцатиричному представлению IP-адреса 140.252.13.33 хоста sun. Это тот IP-адрес, который был возвращен в RARP-ответе. Расширение в имени файла SUN4C соответствует типу загружаемой системы.

В строке 3 показано, что TFTP-запрос содержится как бы непосредственно в IP-дейтаграмме длиной 65 байтов, хотя на самом деле он упакован в UDP-дейтаграмме. Эта неточность является следствием того, что мы запустили tcpdump с ключом -e. Кроме того, обратите внимание, что на рис. 4.8 длина кадра Ethernet, указанная в строке 2, меньше того минимума (60 байтов). Дело в том, что, когда программа tcpdump выполняется на том же хосте (в нашем случае bsdi), который посылает этот кадр Ethernet, она не может учесть "заполняющие" байты кадра. Приложение rarpd записывает 42 байта в буфер (14 байтов Ethernet-заголовка и 28 байтов RARP-ответа), и именно эти 42 байта (точнее, их копию) получает tcpdump. Лишь потом драйвер Ethernet дополняет этот кадр до минимального размера (60 байтов вместе с заголовком). Если бы мы выполняли tcpdump на другом хосте, размер этого кадра был бы показан равным 60 байтам.

После того как бездисковая система узнает свой IP-адрес из RARP-ответа, она наконец может послать TFTP-запрос, чтобы прочитать образ диска начальной загрузки, что мы и видим в рассмотренном примере. Здесь мы не будем вдаваться в дополнительные подробности того, как происходит начальная загрузка бездисковых систем.
Что происходит, когда в сети отсутствует RARP-сервер. Адрес приемника в каждом кадре — широковещательный. После who-is показан аппаратный адрес, для которого запрашивается соответствующий ему IP-адрес, а следом за tell указан аппаратный адрес передатчика (эти аппаратные адреса совпадают).

Обратите внимание на частоту повторения запросов по таймауту. Первый повтор происходит через 6.5 с, затем интервал возрастает до 42.80 с, затем падает до 5.34 с, далее он снова составляет 6.5 с, и так до бесконечности. Если мы подсчитаем разницу между интервалами, то увидим эффект удвоения: 1.21 с от 5.34 до 6.55, 2.42 с от 6.55 до 8.97, 4.83 с от 8.97 до 13.80 и т. д. Когда интервал достигает некоторого предела (в данном случае 42.80 с), он возвращается к начальному значению 5.34 с.
Такое увеличение таймаута лучше, чем фиксированная периодичность.

Протоколы RARP, BOOTP и DHCP

Протокол ARP решает проблему определения по заданному IP-адресу Ethernet-адреса хоста. Иногда бывает необходимо решить обратную задачу, то есть по заданному Ethernet-адресу определить IP-адрес. В частности, эта проблема возникает при загрузке бездисковой рабочей станции. Обычно такая машина получает двоичный образ своей операционной системы от удаленного файлового сервера.
Но как ей узнать его IP-адрес?
Первым для решения проблемы был разработан протокол **RARP** (Reverse Address Resolution Protocol — протокол обратного определения адреса), описанный в RFC 903. Этот протокол позволяет только что загрузившейся рабочей станции разослать всем свой Ethernet-адрес и сказать: ≪Мой 48-разрядный Ethernet-адрес — 14.04.05.18.01.25. Знает ли кто-нибудь мой IP-адрес?≫ RARP-сервер видит этот запрос, ищет Ethernet-адрес в своих файлах конфигурации и посылает обратно соответствующий IP-адрес.

Использование протокола RARP лучше внедрения IP-адреса в образ загружаемой памяти, так как это позволяет использовать данный образ памяти для разных машин. Если бы IP-адреса хранились бы где-то в глубине образа памяти, каждой машине понадобился бы свой отдельный образ.

Недостаток протокола RARP заключается в том, что в нем для обращения к RARP-серверу используется адрес, состоящий из одних единиц (ограниченное широковещание). Однако эти широковещательные запросы не переправляются маршрутизаторами в другие сети, поэтому в каждой сети требуется свой RARP-сервер. Для решения данной проблемы был разработан альтернативный загрузочный протокол **BOOTP.** В отличие от RARP, он использует UDP-сообщения, пересылаемые маршрутизаторами в другие сети. Он также снабжает бездисковые рабочие станции дополнительной информацией, включающей IP-адрес файлового сервера, содержащего образ памяти, IP-адрес маршрутизатора по умолчанию, а также маску подсети. Протокол BOOTP описан в документах RFC 951, 1048 и 1084.

Серьезной проблемой, связанной с применением BOOTP, является то, что таблицы соответствия адресов приходится настраивать вручную. Когда к ЛВС подключается новый хост, протокол BOOTP невозможно использовать до тех пор, пока администратор сети не присвоит ему IP-адрес и не пропишет вручную в конфигурационных таблицах пару (Ethernet-адрес, IP-адрес). Для устранения влияния этого фактора протокол BOOTP был изменен и получил новое имя:

DHCP (Dynamic Host Configuration Protocol — протокол динамической настройки хостов). DHCP позволяет настраивать таблицы соответствия адресов как вручную, так и автоматически. Этот протокол описан в RFC 2131,и 2132. В большинстве систем он уже практически заменил RARP и BOOTP.

Подобно RARP и BOOTP, DHCP основан на идее специализированного сервера, присваивающего IP-адреса хостам, которые их запрашивают. Такой сервер не обязательно должен быть подключен к той же ЛВС, что и запрашивающий хост. Поскольку сервер DHCP может быть недоступен с помощью широковещательной рассылки, в каждой ЛВС должен присутствовать агент **ретрансляции.**

5. АДРЕСАЦИЯ В INTERNET

5.1 Базовая адресация в Internet.

IP-адрес узла ВС идентифицирует точку доступа модуля IP к сетевому интерфейсу, а не всю машину.

Менеджер сети присваивает IP-адреса машинам в соответствии с тем, к каким IP-сетям они подключены. Старшие биты 4-х байтного IP-адреса определяют номер IP-сети. Оставшаяся часть IP-адреса - номер узла (хост-номер).

Существуют 5 классов IP-адресов, отличающиеся количеством бит в сетевом номере и хост-номере. Класс адреса определяется значением его первого октета.

В течение вот уже нескольких десятилетий IP-адреса делятся на пять классов, показанных на рис. 5.1 Такое распределение обычно называется ***базовой*** или ***полноклассовой*** **адресацией.**

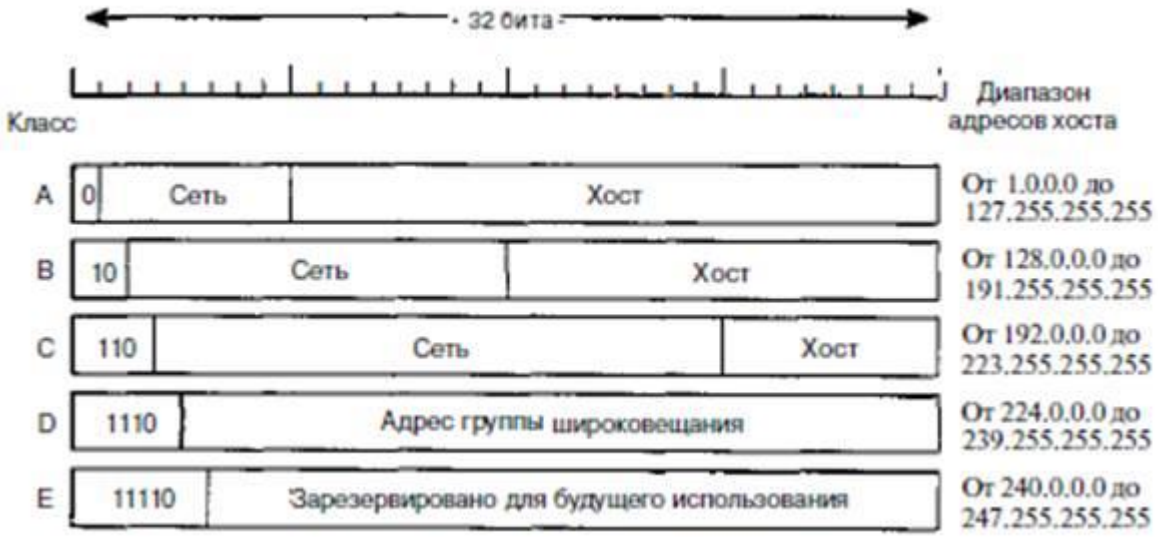


Рис. 5.1 Базовые форматы IP-адресов

Форматы классов A, B, C и D позволяют задавать адреса до 128 сетей с 16 млн. хостов в каждой, 16 384 сетей с 64 тысячами хостов или 2 миллионов сетей (например, ЛВС) с 256 хостами (хотя некоторые из них могут быть специализированными). Предусмотрен класс для многоадресной рассылки, при которой дейтаграммы рассылаются одновременно на несколько хостов. Адреса, начинающиеся с 1111, зарезервированы для будущего применения. В настоящее время к Интернету подсоединено более 500 000 сетей, и это число растет с каждым годом. Во избежание конфликтов, номера сетям назначаются некоммерческой корпорацией по присвоению имен **и** номеров, **ICANN** (Internet Corporation for Assigned Names and Numbers). В свою очередь, ICANN передала полномочия по присвоению некоторых частей адресного пространства региональным органам, занимающимся выделением IP-адресов провайдерам и другим компаниям.

Сетевые адреса, являющиеся 32-разрядными числами, обычно записываются в виде четырех десятичных чисел, которые соответствуют отдельным байтам, разделенных точками. Например, шестнадцатеричный адрес C0290614 записывается как 192.41.6.20. Наименьший IP-адрес равен 0.0.0.0, а наибольший — 255.255.255.255.

В табл.5.1 приведено соответствие классов адресов значениям первого октета и указано количество возможных IP-адресов каждого класса.

Табл.5.1

Характеристики классов адресов

Класс	Диапазон значений первого октета	Возможное кол-во сетей	Возможное кол-во узлов
A	1 - 126	126	16777214
B	128-191	16382	65534
C	192-223	2097150	254
D	224-239		2**28
E	240-247		2**27

Адреса класса A предназначены для использования в больших сетях общего пользования. Они допускают большое количество номеров узлов.

Адреса класса B используются в сетях среднего размера, например, сетях университетов и крупных компаний. Адреса класса C используются в сетях с небольшим числом компьютеров. Адреса класса D используются при обращениях к группам машин, а адреса класса были E зарезервированы на будущее.

Базовая адресация поддерживает: единичную передачу (unicast); групповую передачу (multicast); широковещательную передачу (broadcast).

Некоторые IP-адреса являются *выделенными* и трактуются по-особому, как показано на рис. 5.2

номер сети	номер узла	
все нули		Данный узел (передающий)
номер сети	все нули	Данная IP-сеть
все нули	номер узла	Узел в данной (локальной) IP-сети
все единицы		Все узлы в данной (локальной) IP-сети
номер сети	все единицы	Все узлы в указанной IP-сети
127	Например, все единицы	"Петля"

Рис.5.2 Выделенные IP-адреса

Как показано на рис. 5.2, в выделенных IP-адресах все нули соответствуют либо данному узлу, либо данной IP-сети, а IP-адреса, состоящие из всех единиц, используются при широковещательных передачах. Для ссылок на всю IP-сеть в целом используется IP-адрес с нулевым номером узла. Особый смысл имеет IP-адрес, первый октет которого равен 127. Он используется для тестирования программ и взаимодействия процессов в пределах одной машины. Когда программа посылает данные по IP-адресу 127.0.0.1, то образуется как бы "петля". Данные не передаются по сети, а возвращаются модулям верхнего уровня, как только что принятые. Поэтому в IP-сети запрещается присваивать машинам IP-адреса, начинающиеся со 127.

5.2 Имена сетей и узлов.

Людям удобнее называть машины по именам, а не числами. Например, у машины по имени alpha может быть IP-адрес 223.1.2.1. В маленьких сетях информация о соответствии имен IP-адресам хранится в файлах "hosts" на каждом узле. Конечно, название файла зависит от конкретной реализации.
В больших сетях эта информация хранится на сервере и доступна по сети.
Несколько строк из файла "hosts" могут выглядеть примерно так:

IP-адрес	Имя узла
223.1.2.1	alpha
223.1.2.2	beta
223.1.2.3	gamma
223.1.2.4	delta
223.1.3.2	epsilon
223.1.4.2	iota

В первом столбце - IP-адрес, во втором - название машины.

В большинстве случаев файлы "hosts" могут быть одинаковы на всех узлах. Заметим, что об узле delta в этом файле есть всего одна запись, хотя на (рис.5.1). он имеет три IP-адреса. Узел delta доступен по любому из этих IP-адресов. Какой из них используется, не имеет значения. Когда узел delta получает IP-пакет и проверяет IP-адрес места назначения, то он опознает любой из трех своих IP-адресов.
IP-сети также могут иметь имена. Например, для трех IP-сетей файл "networks" может выглядеть так:

сетевой номер	имя сети.
223.1.2	development
223.1.3	accounting
223.1.4	factory

В первой колонке -, во второй - имя сети.

В данном примере alpha является узлом номер 1 в сети development, beta является узлом номер 2 в сети development и т.д.

Показанный выше файл hosts удовлетворяет потребности пользователей, но для управления сетью internet удобнее иметь названия всех сетевых интерфейсов. Менеджер сети, возможно, заменит строку, относящуюся к delta:

```
223.1.2.4   devnetrouter   delta
223.1.3.1   accnetrouter
223.1.4.1   facnetrouter
```

Эти три строки файла hosts задают каждому IP-адресу узла delta символьные имена. Фактически, первый IP-адрес имеет два имени: "devnetrouter" и "delta", которые являются синонимами. На практике имя "delta" используется как общеупотребительное имя машины, а остальные три имени - для администрирования сети.

Файлы hosts и networks используются командами администрирования и прикладными программами. Они не нужны собственно для работы сети Internet, но облегчают ее использование.

5.3. Подсети

Как известно, IP-адрес состоит из двух иерархических уровней. Необходимость во введении третьего уровня иерархии — уровня подсетей — была продиктована возникновением дефицита номеров сетей и резким ростом таблиц маршрутизации маршрутизаторов в Internet. После введения уровня подсети номер устройства разделяется на две части — номер подсети и номер устройства в этой подсети (рис.5.3).

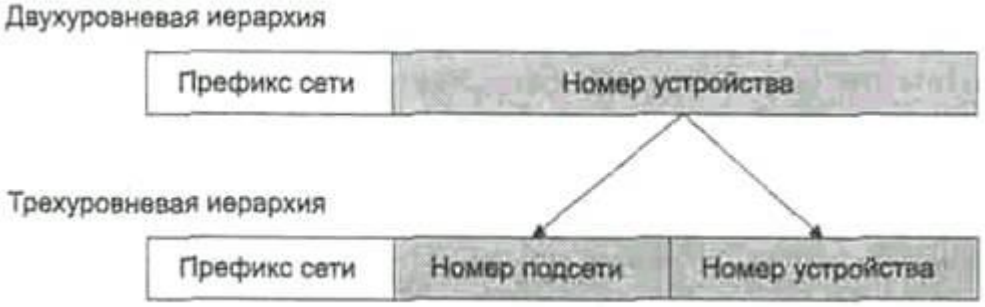


Рис.5.3 Формирование трехуровневой иерархии адресов

Увеличение количества уровней снимает проблему роста таблиц маршрутизации благодаря тому, что информация о топологии корпоративных сетей становится ненужной магистральным маршрутизаторамInternet. Маршруты из сети Internet до любой конкретной подсети, расположенной в сети с данным IP-адресом, одинаковы и не зависят от того, в какой подсети расположен получатель. Это стало возможным благодаря тому, что все подсети сети с данным номером используют один и тот же сетевой префикс, хотя их номера (номера подсетей) разные. Маршрутизаторам в частной сети требуется различать отдельные подсети, но для маршрутизаторов Internet все подсети относятся к единственной записи в таблице маршрутизации. Это позволяет администратору частной сети вносить любые изменения в логическую структуру своей сети, не влияя на размер таблиц маршрутизации маршрутизаторов Internet.

Кроме того, легко решается проблема выделения номеров при росте организации. Организация получает номер сети, а затем администратор произвольно присваивает номера подсетей для каждой внутренней сети. Это позволяет организации расширять свою сеть без необходимости получения еще одного сетевого номера.

На рис.5.4 показана корпоративная сеть (класса B), состоящая из нескольких логических подсетей. Граничный маршрутизатор получает весь трафик из сети Internet, адресованный к сети 130.5.0.0 и передает его внутренним подсетям, основываясь на информации, содержащейся в третьем октете.

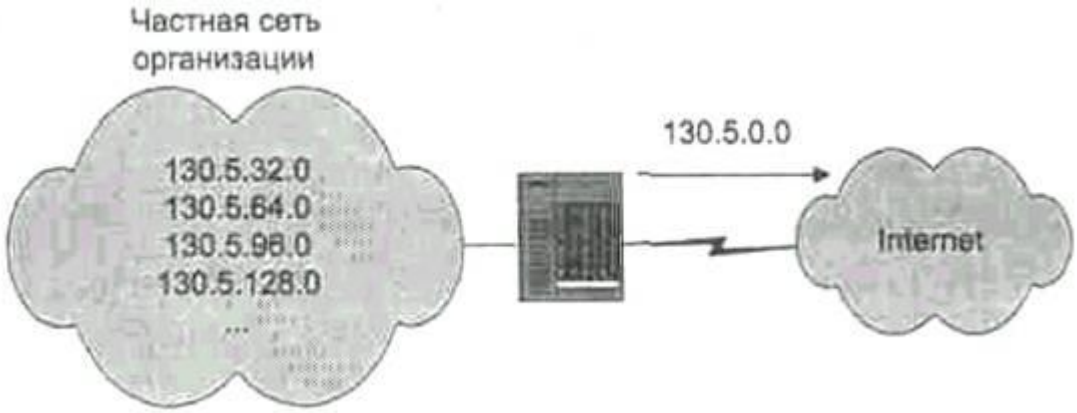


Рис. 5.4 Подсети

Перечислим некоторые преимущества, которые обеспечивает формирование подсетей внутри частной сети:
Размер глобальных таблиц маршрутизации в сети Internet не растет;
Администратор может по своему усмотрению создавать новые подсети без необходимости получения новых номеров сетей;
Изменение топологии частной сети не влияет на таблицы маршрутизации в сети Internet, поскольку маршрутизаторы в Internet не имеют маршрутов в индивидуальные подсети организации — они хранят только маршрут с общим номером сети.

Адресное пространство сети Internet может быть разделено на непересекающиеся подпространства - "подсети", с каждой из которых можно работать как с обычной сетью TCP/IP. Таким образом, единая IP-сеть организации может строиться как объединение подсетей. Как правило, подсеть соответствует одной физической сети, например, одной сети Ethernet.
Конечно, использование подсетей необязательно. Можно просто назначить для каждой физической сети свой сетевой номер, например, номер класса C.
Однако такое решение имеет два недостатока. Первый, и менее существенный, заключается в пустой трате сетевых номеров. Более серьезный недостаток состоит в том, что если ваша организация имеет несколько сетевых номеров, то машины вне ее должны поддерживать записи о маршрутах доступа к каждой из этих IP-сетей. Таким образом, структура IP-сети организации становится видимой для всего мира. При каких-либо изменениях в IP-сети информация о них должна быть учтена в каждой из машин, поддерживающих маршруты доступа к данной IP-сети.
Подсети позволяют избежать этих недостатков. Большая организация (например, провайдер) может получить один сетевой номер, например, номер класса B. Стандарты TCP/IP определяют структуру IP-адресов. Для IP-адресов класса B первые два октета являются номером сети. Оставшаяся часть IP-адреса может использоваться как угодно.
Администратор сети может решить, что третий октет будет определять номер подсети, а четвертый октет - номер узла в ней. Администратор сети должны описать конфигурацию подсетей в файлах, определяющих маршрутизацию IP-пакетов. Это описание является локальным для вашей организации и не видно вне ее. Все машины вне рассматриваемой организации видят одну большую IP-сеть. Следовательно, они должны поддерживать только маршруты доступа к шлюзам, соединяющим рассматриваемую IP-сеть с остальным миром. Изменения, происходящие в IP-сети организации, не видны вне ее. Администратор сети легко можете добавить новую подсеть, новый шлюз и т.п.

После того, как решено использовать подсети или множество IP-сетей, администратор сети должен решить, как назначать им номера. Обычно это довольно просто.

Каждой физической сети, например, Ethernet или Token Ring, назначается отдельный номер подсети или номер сети. В некоторых случаях имеет смысл назначать одной физической сети несколько подсетевых номеров.

Например, предположим, что имеется сеть Ethernet, охватывающая три здания. Ясно, что при увеличении числа машин, подключенных к этой сети, придется ее разделить на несколько отдельных сетей Ethernet. Для того, чтобы избежать необходимости менять IP-адреса, когда это произойдет, можно заранее выделить для этой сети три подсетевых номера - по одному на здание. (Это полезно и в том случае, когда не планируется физическое деление сети.

Просто такая адресация позволяет сразу определить, где находится та или иная машина.) Однако прежде, чем выделять три различных подсетевых номера одной физической сети, тщательно проверьте, что все ваши программы способны работать в такой среде.

5.4 Маска подсети

Если маршрутизаторы в сети Internet используют только сетевой префикс адреса получателя для передачи трафика в организацию, то маршрутизаторы внутри частной сети организации используют расширенный сетевой префикс для передачи трафика индивидуальным подсетям. *Расширенным сетевым префиксом* называют префикс сети и номер подсети. Так что схему на рис.5.3 можно представить также следующим образом (рис. 5.5):

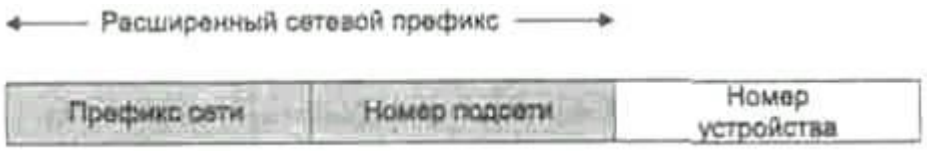


Рис. 5.5 Расширенный сетевой префикс

Понятие расширенного сетевого префикса, по сути, эквивалентно понятию *маска подсети* (subnet mask). Маска подсети — это двоичное число, содержащее единицы в тех разрядах, которые относятся к расширенному сетевому префиксу. Маска подсети позволяет разделить IP-адрес на две части: номер подсети и номер устройства в этой подсети.

Старшие биты IP-адреса используются рабочими станциями и маршрутизаторами для определения класса адреса. После того как класс определен, устройство может легко вычислить границу между битами, использующимися для идентификации номера сети, и битами номера устройства в этой сети. Однако для определения границ битов, идентифицирующих номер подсети, такая схема не подходит. Для этого как раз и используется 32-битная маска подсети, которая помогает однозначно определить требуемую границу. Для стандартных классов сетей маски имеют следующие значения:

255.0.0.0 — маска для сети класса A;

255.255.0.0 — маска для сети класса B;

255.255.255.0 — маска для сети класса C.

Например, если сетевой администратор хочет использовать весь третий октет для номера подсети в сети класса B 130.5.0.0, то ему необходимо указать маску подсети 255.255.255.0. Биты в маске подсети должны быть установлены в единицу, если система, проверяющая адрес, должна рассматривать соответствующий бит в IP-адресе как часть расширенного сетевого префикса. Другими словами, после определения класса IP-адреса, любой бит в номере устройства, который имеет соответствующий установленный бит в маске подсети, используется для идентификации номера подсети. Оставшаяся часть номера устройства, которой соответствуют нулевые биты в маске подсети, используется для задания номера устройства. На рис.5.6 показан пример IP-адреса класса B с соответствующей маской подсети.

В стандартах, описывающих современные протоколы маршрутизации, часто используется длина расширенного сетевого префикса, а не маска подсети. Эта длина показывает число установленных в единицу бит в маски подсети.

Так сетевой адрес 130.5.5.25 с маской подсети 255.255.255.0 может быть записан как 130.5.5.25 /24 (в маске подсети 255.255.255.0 число бит, установленных в единицу, равно 24).

Адрес		130.5.5.25			
Адрес в двоичном виде		10000010.	00000101.	00000101.	00011001
Маска подсети		255.255.255.0			
Маска подсети в двоичном виде		11111111.	11111111.	11111111.	00000000
Сетевой префикс		10000010.	00000101.		
Расширенный сетевой префикс		10000010.	00000101.	00000101.	00011001
Или в более наглядном виде:					
		Сетевой префикс		Номер подсети	Номер устройства
IP-адрес	130.5.5.25	10000010	00000101.	00000101.	00011001
Маска подсети	255.255.255.0	11111111.	11111111.	11111111.	00000000
		Расширенный сетевой префикс			

Рис.5.6. IP-адрес класса B с соответствующей маской подсети

Такая запись является более компактной и легче воспринимается, чем маска подсети в ее традиционном точечно-десятичном формате. В табл.5.2 приведен пример использования расширенного сетевого префикса. В табл.5.3 устройство того же адреса представлено в несколько другом виде.

Таблица 5.2.

Пример записи с использованием расширенного сетевого префикса

	Сетевой префикс		Номер подсети	Номер устройства
130.5.5.25	10000010.	00000101.	00000101.	00011001
255.255.255.0	11111111.	11111111.	11111111.	00000000
Эквивалентная запись				
	24-битовый расширенный сетевой префикс			Номер устройства
130.5.5.25/24	10000010.	00000101.	00000101.	00011001

Таблица5.3.

Структура адреса с расширенным сетевым префиксом

Адрес	130.5.5.25			
Адрес в двоичном виде	10000010.	00000101.	00000101.	00011001
Маска подсети в десятичном виде	255.255.255.0			
Маска подсети в двоичном виде	11111111.	11111111.	11111111.	00000000
Номер подсети			00000101.	
Номер устройства				00011001
Адрес с расширенным сетевым префиксом	130.5.5.25/24			
Адрес с расширенным сетевым префиксом в двоичном виде	10000010.	00000101.	00000101.	00011001

Однако следует учитывать, что большинство современных протоколов маршрутизации переносят маску подсети в своих сообщениях. В то же время, не существует стандартного протокола маршрутизации, который имел бы дополнительное однобайтовое поле в заголовке своих сообщений, содержащее запись о числе бит в расширенном сетевом префиксе. Каждый протокол маршрутизации передает полную 4-октетную маску подсети.

Для администратора сети чрезвычайно важно знать четкие ответы на следующие вопросы:

Сколько подсетей требуется организации сегодня?

Сколько подсетей может потребоваться организации в будущем?

Сколько устройств в наибольшей подсети организации сегодня?

Сколько устройств будет в самой большой подсети организации в будущем?

Первым шагом в процессе планирования является определение максимального количества требуемых подсетей. Данное число округляется вверх до ближайшей степени двойки. Затем важно учесть возможность увеличения числа подсетей. Наконец, проверяется достаточность адресов устройств в самой большой подсети организации на настоящий момент и в обозримом будущем.

Предположим, что организация получила сеть класса С 193.1.1.0 и ей необходимо сформировать шесть подсетей. Наибольшая подсеть должна поддерживать 25 устройств. На первом шаге определяется число бит, необходимых для выделения шести подсетей. Очевидно, необходимо выделить три бита (23=86). Так как организации выделены адреса класса С (префикс /24), то получаемый расширенный сетевой префикс равен /27 (24+3=27). Это соответствует маске подсети 255.255.255.224 (табл. 5.4).

Таблица5.4

Пример определения маски подсети в организации

	Сетевой префикс			Байт для задания номеров устройств в данной сети	
	Байты для задания номера сети			Биты для номеров подсетей	
193.1.1.0	11000001.	00000001.	193.1.1.0	11000001.	00000001.

255.255.255.224	11111111	11111111	255.255.255.224	11111111.	11111111
Эквивалентная запись					
193.1.1.0/27	11000001.00000001.00000001.000				00000

В табл. 5.5 устройство адреса в этом примере рассматривается более подробно.

Таблица5.5

Маска подсети в организации

Адрес				193.1.1.0	
Адрес в двоичном виде	11000001.	Адрес в двоичном виде	11000001.	Адрес в двоичном виде	11000001.
Маска подсети		Маска подсети		Маска подсети	
Маска подсети в двоичном виде	11111111.	Маска подсети в двоичном виде	11111111.	Маска подсети в двоичном виде	11111111.
Эквивалентная запись					
Адрес с расширенным сетевым префиксом				193.1.1.0/27	
Адрес с расширенным сетевым префиксом в двоичном виде	11000001.	Адрес с расширенным сетевым префиксом в двоичном виде	11000001.	Адрес с расширенным сетевым префиксом в двоичном виде	11000001.

Номер подсети необязательно должен располагаться сразу после сетевого префикса. Администратор может устанавливать биты в маске подсети независимо от остальной части адреса. В примере с адресом 193.1.1.0 /27 третий байт маски подсети вместо 11100000_2 может быть, например, установлен в 00011100_2 . Однако на практике в большинстве случаев так не поступают.

Используемый 27-битовый расширенный сетевой прификс оставляет 5 бит для задания номеров устройств в каждой из подсетей. Это означает, что в каждой подсети может быть использовано до 32 (2^5=32) устройств. Однако, так как адреса, у которых все биты равны нулю или единице, являются зарезервированными, общее число адресов устройств в каждой подсети равно 30 (32-2).

Для выделения подсети сетевой администратор помещает двоичное представление номера этой подсети (для восьми подсетей это может быть число от 0 до 7) в битовое поле номера подсети. Например, для определения подсети 4 администратор просто помещает двоичное представление числа 4 (100_2) в трехбитовое поле номера подсети. Таблица 5.6 содержит все восемь возможных вариантов подсетей в рассматриваемом примере.

Самым простым способом проверить, что все подсети выделены правильно, является следующий. Убедитесь в том, что все десятичные номера подсетей кратны номеру подсети #1. В данном примере все номера подсетей кратны 32.

Таблица5.6

Возможные варианты подсетей

Сеть/адрес	Точечно-десятичный формат	Двоичный формат
Базовая сеть	193.1.1.0/24	11000001.00000001.00000001.00000000
Подсеть #0	193.1.1.0/27	11000001.00000001.00000001.**000**00000
Подсеть #1	193.1.1.32/27	11000001.00000001.00000001.**001**00000
Подсеть #2	193.1.1.64/27	11000001.00000001.00000001.**010**00000
Подсеть #3	193.1.1.96/27	11000001.00000001.00000001.**011**00000

Подсеть #4	193.1.1.128/27	11000001.00000001.00000001.**100**00000
Подсеть #5	193.1.1.160/27	11000001.00000001.00000001.**101**00000
Подсеть #6	193.1.1.192/27	11000001.00000001.00000001.**110**00000
Подсеть #7	193.1.1.224/27	11000001.00000001.00000001.**111**00000

Первоначально документ RFC 950 запрещал использование номеров подсетей, у которых все биты установлены в единицы или нули. Причиной такого ограничения являлось то, что некоторые протоколы маршрутизации не переносят в своих служебных сообщениях ни маски подсети, ни длины расширенного сетевого префикса. Например, при использовании протокола маршрутизации RIP версии 1 маршруты в разные подсети с адресами 193.1.1.0 /27 (00000) и 193.1.1.0 /24 (00000000) будут рассматриваться как идентичные. Аналогичная проблема возникает и в случае установки всех бит в единицу. Например, адрес 193.1.1.255 будет широковещательным адресом и для сети 193.1.1.0 /24 (номер устройства 11111111) и для сети 193.1.1.224 /27 (номер устройства 11111). В табл. 8.8 показаны обе рассмотренные ситуации.

С разработкой протоколов маршрутизации, переносящих в своих служебных сообщениях маску подсети (OSPF, IS-IS), стало возможным использование подсетей, все биты номеров которых установлены в единицу или ноль — вопреки документу RFC 950. В результате производители позволяют настраивать подсети с такими номерами на портах своих маршрутизаторов. При этом, однако, нужно учитывать два обстоятельства: используемые в корпоративной сети протоколы маршрутизации, относящиеся к классу IGP, должны поддерживать маску подсети или расширенный сетевой префикс. Кроме того, необходимо, чтобы маршрутизаторы в сети поддерживали номера подсетей со всеми единичными или нулевыми битами. При этом важно учитывать номер версии программного обеспечения маршрутизатора. Например, маршрутизатор NetBuilder II фирмы 3Com включает полную поддержку таких подсетей, начиная с версии 8.3.0.2.

Таблица5.7

Идентичные маршруты и широковещательные адреса

Маршруты в сети	193.1.1.0/24	11000001.00000001.00000001.(24-бито-вый расширенный сетевой префикс)	00000000
	193.1.1.0/27	11000001.00000001.00000001.000 (27-би-товый расширенный сетевой префикс)	00000
Широковещательные адреса	193.1.1.0/24	11000001.00000001.00000001. (24-бито-вый расширенный сетевой префикс)	11111111
	193.1.1.224/27	11000001.00000001.00000001.111 (27-битовый расширенный сетевой префикс)	11111

В рассмотренном примере остается 5 бит для задания адресов устройств в каждой подсети. В результате каждая подсеть может содержать блок из 30 адресов устройств (2^5-2). Устройства нумеруются от 1 до 30. Для определения адреса устройства # N в сети администратор помещает двоичное представление числа N в поле номера устройства. Например, для выделения адреса устройству #28 в подсети #2 администратор вставляет двоичное представление 28 (11100_2) в пятибитовое поле подсети #2. В табл. 5.8 показаны некоторые возможные номера устройств в подсети #2.

Таблица. 5.8

Адреса устройств в подсети #2

Сеть (устройство)/адрес	Точечно-десятичный формат	Двоичный формат
Подсеть #2	193.1.1.64/27	11000001.00000001.00000001.01000000
Устройство #1	193.1.1.65/27	11000001.00000001.00000001.01000001
Устройство #2	193.1.1.66/27	11000001.00000001.00000001.01000010
Устройство #3	193.1.1.67/27	11000001.00000001.00000001.01000011
Устройство #28	193.1Л.92/27	11000001.00000001.00000001.01011100
Устройство #29	193.1.1.93/27	11000001.00000001.00000001.01011101
Устройство #30	193.1.1.93/27	11000001.00000001.00000001.01011110
Широковещательный адрес для подсети #2		
	193.1.1.95	11000001.00000001.00000001.01011111

Для того чтобы проверить правильность широковещательного адреса для определенной подсети, можно использовать следующее простое правило. Во всех случаях широковещательный адрес для подсети #N на единицу меньше, чем базовый адрес для подсети #(N+1). Например, широковещательный адрес для подсети #2 (193.1.1.95) на единицу меньше базового адреса подсети #3 (193.1.1.96).

При введении подсетей значительно усложнился процесс определения принадлежности отправителя и получателя к одной сети.

Теперь перед отправкой дейтаграмм устройству необходимо определить:

располагается ли получатель в той же подсети, что и отправитель;

какой маршрутизатор необходимо использовать (в том случае, если существует несколько (более одного) маршрутизаторов, имеющих маршрут в нужную сеть).

До введения подсетей в поле сетевого номера IP-адрес получателя сравнивался отправителем с собственным сетевым номером. Если сетевые номера совпадали, то считалось, что устройства располагаются в одной локальной сети.

Однако после введения подсетей получатель может располагаться в другой подсети той же самой сети, что и получатель. В этом случае для проверки используется маска подсети. Над IP-адресом получателя и маской подсети выполняется операция логическое «И». Результат сравнивается с результатом выполнения этой же операции над собственным IP-адресом и той же маской подсети. Если результаты совпадают, то отправитель и получатель находятся в одной подсети и дейтаграмма может быть послана напрямую. Если результаты различны, то получатель находится в другой подсети. В этом случае дейтаграмма посылается маршрутизатору.

Документ RFC 1219 определяет основное правило, которому желательно следовать при присваивании номеров подсетям и устройствам. Номера подсетей назначают таким образом, чтобы старшие биты в номере подсети устанавливались первыми. Например, если поле номера подсети состоит из четырех бит, то первые несколько номеров подсетей должны быть следующими: 8 (1000_2), 4 (0100_2), 12 (1100_2), 2 (0010_2),

6 (0110_2) и т. д. Иными словами, единичные биты номеров подсетей рекомендуется устанавливать, начиная с крайней левой позиции. В то же время единичные биты номеров устройств рекомендуется устанавливать, начиная с крайней правой позиции (табл. 8.10). В нашем случае сетевой префикс состоит из двух октетов (в маске 11111111.11111111.), за ними (в адресе) следует 4 бита номера подсети и 12 бит остается под номер устройства.

Если следовать данному правилу, то на границе между номером подсети и номером устройства будут существовать нулевые неиспользуемые биты. Это позволяет изменять маску подсети без изменения IP-адреса, присвоенного устройству. Необходимость изменения маски подсети может возникнуть при увеличении числа устройств в каждой подсети. В этом случае можно «заимствовать» часть бит из числа зарезервированных под номера подсетей. Достоинством описанного правила является то, что администратору при изменении маски подсети на устройстве не надо менять IP-адрес устройства. Изменение адресов может потребовать больших усилий от администратора: перенастройки почтовых служб, модификации статических таблиц маршрутизации и т. д.

Таблица . 5.9

Рекомендуемая схема присвоения адресов

Номера подсетей	Биты адреса	Номера устройств
128	1000 0000. 0000 0001	1
64	0100 0000. 0000 0010	2
192	1100 0000. 0000 0011	3
32	0010 0000. 0000 0100	4
160	1010 0000. 0000 0101	5
96	01100000. 0000 0110	6
224	1110 0000. 0000 0111	7

В сети с подсетями можно использовать два вида широковещания: направленное и ограниченное. Направленное широковещание используется для передачи дейтаграммы всем устройствам определенной подсети. Для посылки дейтаграммы всем устройствам во всех подсетях необходимо использовать ограниченное широковещание с адресом 255.255.255.255. Необходимо, однако, учесть, что маршрутизаторы не пропускают дейтаграммы с таким адресом (поэтому такое широковещание и называется ограниченным). В средах с подсетями существует ограничение на направленное широковещание. Биты, используемые для формирования номеров подсетей и обычно (в традиционных сетях) являющиеся частью поля номера устройства, не могут быть установлены в нули или единицы. Например, пусть у нас есть адрес класса С, в котором третий байт выделен под номера подсетей: 128.1.Номер подсети. Номер устройства. В этом случае адрес направленного широковещания не может быть равен 128.1.255.255, 128.1.0.255, 128.1.255.0 или 128.1.0.0.

На рис. 5.7 показан пример сети с подсетями, связанными маршрутизаторами. Каждый из маршрутизаторов хранит маршруты во все подсети. Маска подсети равна 255.255.255.0. В табл. . 5.10 приводится список получателей широковещательных дейтаграмм, отправляемых рабочей станцией А.

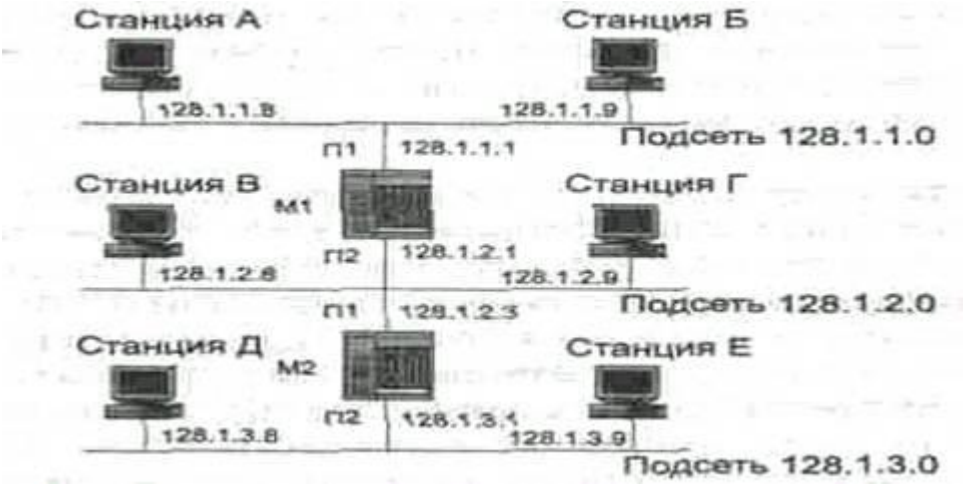


Рис. 5.7 Пример широковещания в сети

Таблица. 5.10

Получатели широковещательных дейтаграмм от станции А

IP-адрес	Получатели
255.255.255.255	Станция Б и порт 1 (П1) маршрутизатора Ml
128.1.1.255	Станция Б и порт 1 Ml
128.1.2.255	Станции В и Г; порт 2 Ml и порт 1 М2
128.1.3.255	Станции Д и Е; порт 2 М2

Маска подсети переменной длины

В 1987 году вышел документ RFC 1009, определяющий использование разных масок подсетей в одной сети, состоящей из большого количества подсетей. Так как в этом случае расширенные сетевые префиксы в различных подсетях имеют разную длину, говорят о масках подсетей переменной длины. Маску подсети переменной длины поддерживают современные протоколы маршрутизации, такие какOSPF и IS-IS (см. ниже). Сообщения этих протоколов переносят как адрес подсети, так и соответствующую ему маску. Если протокол маршрутизации не позволяет использовать маску подсети, маршрутизатор будет либо предполагать, что должна использоваться маска подсети, присвоенная его локальному порту, либо выполнять поиск в статически настроенной таблице, содержащей всю информацию о масках подсетей. Первое решение не гарантирует правильности выбора маски подсети, а статическая таблица не имеет возможности масштабирования. Кроме того, ею сложно управлять и исправлять в ней ошибки также непросто.

Таким образом, если требуется использование маски подсети переменной длины в сложной сетевой топологии, то наилучшим выбором является применение протоколов маршрутизации OSPF, IS-IS, а неRIP-1 IP. Однако при этом нужно учитывать, что вторая версия протокола RIP (RIP-2 IP), описанная в документе RFC 1388, расширяет возможности первой версии протокола, в том числе и добавлением возможности переноса маски подсети.

Так как протокол RIP-1 не переносит информацию о масках подсетей в своих сообщениях об обновлении маршрутизации, то сохраняются маски подсетей, используемые с каждым номером сети. При отсутствии данной информации протокол маршрутизации RIP-1 IP выбирает маску подсети, которая соответствует каждому маршруту в его таблице маршрутизации.

Рассмотрим пример сети, на входе которой стоит маршрутизатор. Порту 1 этого маршрутизатора присвоен адрес 130.24.13.1 с маской 255.255.255.0 (расширенный сетевой префикс /24), а порту 2 — адрес 200.14.13.2 с такой же маской подсети. Анализируя первые биты адреса порта 1 и маску подсети, маршрутизатор определит, что это адрес класса B, поэтому третий байт адреса используется для задания номера подсети. Порту 2 присвоен адрес класса C без выделения подсетей.

Если маршрутизатор получает информацию о маршруте к сети 130.24.36.0 от своего соседа через порт 1, он будет использовать маску подсети 255.255.255.0 (расширенный сетевой префикс /24), так как порту 1 присвоен адрес с тем же номером сети 130.24.0.0. Маска подсети просто наследуется. Но если маршрутизатор получит от соседа информацию о маршруте к сети 131.25.0.0, он будет использовать стандартную маску подсети 255.255.0.0, так как адрес 131.25.0.0 является адресом класса B, а этому классу соответствует маска подсети 255.255.0.0. Будет использоваться именно эта маска, так как маршрутизатор не имеет другой информации о маске подсети.

Маршрутизатор, поддерживающий протокол RIP-1 IP, включает биты, определяющие номера подсетей в сообщения об обновлении маршрутов, только в том случае, если порт, через который предполагается посылать сообщения, настроен на подсеть с тем же номером сети. Если порт настроен с другим сетевым номером, маршрутизатор будет рассылать только сетевую часть адреса.

Теперь предположим, что входной маршрутизатор получил информацию от соседа о маршруте к сети 130.24.36.0. Так как порт 1 настроен на адрес того же класса, то маршрутизатор предположит, что сеть 130.24.36.0 имеет маску 255.255.255.0. Поэтому, когда наступает время оповестить о данном маршруте, он будет информировать о маршруте к сети с адресом 130.24.36.0 через свой порт 1 и о маршруте к сети 130.24.0.0 через порт 2. Во втором случае оказывается утраченной информация, содержащаяся в третьем байте адреса (36).

Протокол RIP-1 IP может использовать только одну маску подсети для данного номера сети. Возможность присваивания одному адресу нескольких масок подсетей предоставляет несколько преимуществ. Множество масок подсетей позволяет более эффективно использовать выделенное организации адресное пространство. Кроме того, удается объединять маршруты, что значительно уменьшает количество маршрутной информации внутри домена маршрутизации.

О нескольких масках подсетей, присвоенных одному адресу, часто говорят как о *маске подсети переменной длины* (Variable Length Subnet Mask, VLSM). Основной проблемой этого метода является совместимость с предыдущими версиями протоколов, которые использовали только одну маску подсети.

Пусть администратор сети организации хочет настроить сеть класса B 130.5.0.0 на расширенный сетевой префикс /22 (табл. . 5.11). Для задания номеров подсетей могут использоваться 6 бит.

Таблица 5.11

Сеть класса B 130.5.0.0 при расширенном сетевом префиксе /22

Адрес сети с расширенным сетевым префиксом	130.5.0.0/22			
Сетевой префикс (класс B)	10000010.	00000101.		
Биты для номеров подсетей			000000	
Биты для номеров устройств				00.00000000

В этой сети с расширенным сетевым префиксом /22 будут доступны 64 подсети (2^6=64), каждая из которых поддерживает максимум до 1022 (2^{10}-2=1022) адресов устройств. Такой вариант устроит администратора, если организации нужно небольшое число подсетей с большим количеством устройств в них. Однако, допустим, организации нужны подсети с числом устройств, не превышающим 30. При использовании

фиксированной маски подсети администратору придется создавать подсети, рассчитанные на значительно большее чем 30 количество устройств (а именно, 1022). В результате невостребованными оказываются около 1000 адресов устройств в подсетях. Как видно из этого примера, ограничения, вызываемые необходимостью применять единую маску подсети, значительно уменьшают эффективность использования всего адресного пространства, выделенного организации.

Использование маски подсети переменной длины дает возможность легко преодолеть эти трудности. Действительно, предположим, что администратор хочет использовать расширенный сетевой префикс /26. Сеть класса В с таким расширенным сетевым префиксом позволяет поддерживать до 1024 подсетей (2^{10}), каждая из которых может содержать до 62 (2^6-2) индивидуальных адресов устройств (табл. 5.12). Такой расширенный сетевой префикс идеально подходит к небольшим подсетям с числом устройств порядка 60.

Таблица5.11

Распределение адресного пространства при префиксе /26

Адрес сети с расширенным сетевым префиксом	130.5.0.0/26			
Сетевой префикс (класс В)	10000010.	00000101.		
Биты для номеров подсетей			00000000.00	
Биты для номеров устройств				000000

Как видно, применение различных расширенных сетевых префиксов (/22 и /26) позволило получить две разные подсети, отличающиеся по числу поддерживаемых устройств. Маска подсети переменной длины позволяет администратору выделять подсети с необходимыми характеристиками. При этом созданные подсети можно со временем легко изменять. Общая схема такова: сначала сеть делится на подсети, затем некоторые из этих подсетей делятся на более мелкие подсети и т. д. То есть происходит рекурсия (дробление) подсетей.

Рассмотрим другой пример. На рис.5.8 показано, как сеть класса А с адресом 10.0.0.0 сначала разделяется на подсети с расширенным сетевым префиксом /16 (маска подсети 255.255.0.0). Получается 254 подсети. В каждой подсети поддерживается до 65 534 (2^{16}-2) индивидуальных адресов устройств. Полученная подсеть с адресом 10.253.0.0 с расширенным сетевым префиксом /24 поддерживает 254 подсети, каждая из которых включает до 254 (2^8-2) устройств. При дальнейшей рекурсии с расширенным сетевым префиксом /27 подсеть с адресом 10.253.1.0 будет включать 6 подсетей с номерами, кратными 32, содержащих до 30 (2^5-2) устройств.

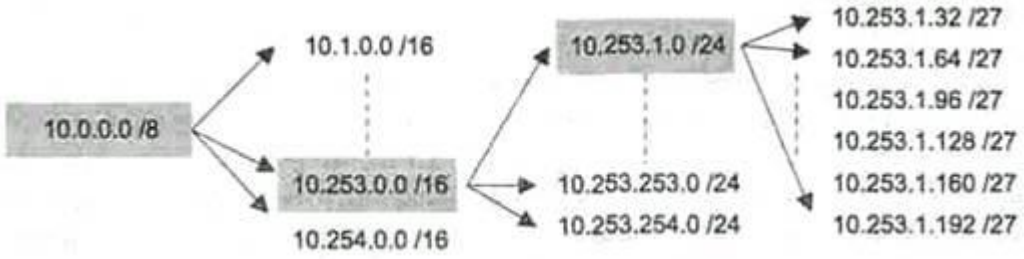


Рис. 5.8 Иерархия адресов подсетей

Таким образом, иерархическое (рекурсивное) разбиение адресного пространства позволяет гибко настроить сеть организации. Кроме того, внедрение маски подсети переменной длины позволяет значительно уменьшить объемы таблиц маршрутизации.

Дело в том, что каждый маршрутизатор теперь может включить информацию о всех своих подсетях в одну запись сообщения об обновлении. Так как структура подсетей не имеет значения для внешних сетей, маршрутизатор Ml оповещает маршрутизаторы в сети Internet только о маршруте с адресом 10.0.0.0 (рис.5.9).

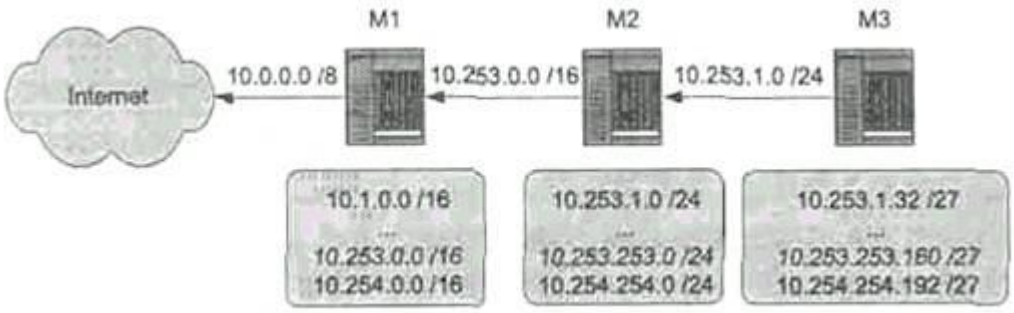


Рис. 5.9 Объединение подсетей в одну запись

Естественно, даже при использовании маски подсети переменной длины администратору следует убедиться, что класс сети организации достаточен для выделения требуемого количества подсетей.

Рассмотрим сеть организации, которая охватывает несколько удаленных филиалов. Если организация имеет три удаленных сети, то ей потребуется выделить 3 бита для формирования подсетей (2^3=83). Этих 3 бит хватит и в обозримом будущем. Предположим, что администратор хочет сформировать отдельные подсети внутри каждого филиала — второй уровень в иерархии подсетей. Внутри этих подсетей необходимо выделить отдельные рабочие группы и их подсети. Следуя приведенной выше модели, мы видим, что верхний уровень иерархии определяется числом удаленных филиалов, второй — числом зданий внутри филиалов, а третий — максимальным числом подсетей в каждом здании и максимальным числом устройств в каждой из подсетей.

Для поддержки маски подсети переменной длины требуется выполнение трех основных условий:

Протокол маршрутизации должен переносить информацию о расширенном сетевом префиксе;

Все маршрутизаторы должны поддерживать алгоритм передачи, основывающийся на технологии наибольшего совпадения (longest match);

Адреса должны присваиваться в соответствии с существующей топологией сети.

Правило наибольшего совпадения основывается на том факте, что маршрут в таблице маршрутизации с большим расширенным сетевым префиксом определяет меньший набор получателей, чем тот же маршрут с коротким расширенным сетевым префиксом. Поэтому маршрутизатор должен выбирать маршрут с наибольшим расширенным сетевым префиксом (как наиболее точно определяющий получателей) при передаче трафика. В этом и состоит правило наибольшего совпадения.

Например, если адрес получателя равен 11.1.2.5 и в таблице маршрутизации есть три маршрута к этой сети (табл. 5.12), маршрутизатор выберет маршрут #1, так как его расширенный сетевой префикс совпадает с адресом получателя в большем числе бит.

Таблица. 5.12

Выбор маршрута с наибольшим совпадением

Получатель	11.1.2.5	00001011.00000001.00000010.00000101
Маршрут #1	11.1.2.0/24	00001011.00000001.00000010.00000000
Маршрут #2	11.1.0.0/16	00001011.00000001.00000000.00000000
Маршрут #3	11.0.0.0/8	00001011.00000000.00000000.00000000

Здесь необходимо сделать одно важное замечание. Адрес получателя (11.1.2.5) совпадает с тремя маршрутами. Согласно правилу наибольшего совпадения будет выбран маршрут к подсети 11.1.2.0 /24. Но может оказаться так, что устройство с адресом 11.1.2.5 не будет входить в подсеть 11.1.2.0. Тогда маршрутизатор не сможет передать трафик этому устройству. Поэтому назначение адресов следует обязательно проводить исходя из существующей сетевой топологии и при этом непременно учитывать правило наибольшего совпадения.

Иерархическая маршрутизация (реализованная в протоколе OSPF) требует, чтобы адреса устройств отражали действительную сетевую топологию на всех уровнях. Только при этом условии несколько подсетей можно объединить в одном сообщении о маршруте. Этот постулат является основополагающим при рассмотрении технологии бесклассовой маршрутизации (CIDR).

5.5. IP-таблица маршрутов

Как модуль IP узнает, какой именно сетевой интерфейс нужно использовать для отправления IP-пакета? Модуль IP осуществляет поиск в таблице маршрутов. Ключом поиска служит номер IP-сети, выделенный из IP-адреса места назначения IP-пакета.

Таблица маршрутов содержит по одной строке для каждого маршрута.

Основными столбцами таблицы маршрутов являются номер сети, флаг прямой или косвенной маршрутизации, IP-адрес шлюза и номер сетевого интерфейса.

Эта таблица используется модулем IP при обработке каждого отправляемого IP-пакета.

В большинстве систем таблица маршрутов может быть изменена с помощью команды "route". Содержание таблицы маршрутов определяется менеджером сети, поскольку менеджер сети присваивает машинам IP-адреса.

6. ЗАГОЛОВОК ДЕЙТАГРАММЫ IPv4

Описание протокола IP (Internet Protocol) дано в документе RFC 791. IP является базовым протоколом всего стека TCP/IP. Он отвечает за передачу информации по сети. Информация передается блоками, которые называются *дейтаграммами.*

IP является протоколом сетевого уровня. При этом для каждой среды передачи данных, например, Ethernet и ATM, определен способ инкапсуляции IP-дейтаграмм. Маршрутизаторы пересылают инкапсулированные дейтаграммы по различным сетям, образуя объединение IP-сетей, по которому каждая рабочая станция может поддерживать связь по протоколу IP с любой другой рабочей станцией.

Услуги, предлагаемые протоколом IP, сводятся к негарантированной доставке дейтаграмм. Протокол IP не исключает потерь дейтаграмм, доставки дейтаграмм с ошибками, а также дублирования и нарушения порядка следования дейтаграмм, заданного при их отправлении.

Протокол IP выполняет фрагментацию и сборку дейтаграмм, если принятый размер кадров в данной сети (или участке распределенной сети) отличается от размера исходных дейтаграмм. В протоколе IPотсутствуют механизмы повышения достоверности передачи данных, управления протоколом и синхронизации, которые обычно предоставляются в протоколах более высокого уровня. Протокол IPполучает информацию для передачи от протоколов, расположенных по сравнению с ним на более высоком уровне. К этим протоколам, прежде всего, относятся протоколы TCP и UDP. После получения информации от них протокол IP передает дейтаграммы через распределенную сеть, используя сервисы локальных сетей.

Дейтаграмма состоит из заголовка и поля данных, которое следует сразу за заголовком. Длина поля данных определяется полем «Общая длина» в заголовке. На рис. 6.1 показан формат заголовка IP-дейтаграммы.

Номер версии (4 бита)	Длина заголовка (4 бита)	Тип сервиса (8 бит)	Общая длина(16 бит)
Идентификатор (16 бит)	Флаги (3 бита)	Смещение фрагмента (13 бит)	
Время жизни (8 бит)	Протокол (8 бит)	Контрольная сумма заголовка (16 бит)	
Адрес отправителя (32 бита)			
Адрес получателя (32 бита)			
Опции (поле переменной длины)	Выравнивание до 32-битной границы		

Рис. 6.1. Формат заголовка дейтаграммы протокола IP

Поле «Номер версии» указывает на версию используемого протокола IP. В настоящее время распространена версия 4, но планируется переход к версии 6. Связь между абонентами гарантируется только в том случае, если все они работают с одной версией протокола IP. Перед обработкой дейтаграммы это поле проверяется. Если используется, например, версия 4, то при обработке будут отбрасываться дейтаграммы с версией 6.

Поле «Длина заголовка» определяет длину заголовка в 32-битовых словах. Заголовок может иметь минимальный размер 5 слов. При увеличении объема служебной информации эта длина может быть увеличена за счет поля «Опции».

Поле «Тип сервиса» определяет способ обслуживания дейтаграммы. Первые три бита (0-2) этого поля задают приоритет дейтаграммы. Возможные значения приоритета — от 0 (обычная дейтаграмма) до 7 (управляющая дейтаграмма). Устройства в сети учитывают приоритет дейтаграммы и обрабатывают в первую очередь более важные. Информация в остальных битах поля используется протоколами маршрутизации OSPF и BGP. Протоколы маршрутизации отвечают за вычисление наилучшего маршрута к получателю, основываясь на понятии «стоимость пути». Ею может быть скорость, надежность и т. д.

Третий бит (бит 2 — отсчет начинается с нулевого бита) определяет вид задержки: 0 — нормальная задержка, 1 — малая задержка. Этот бит учитывается различными алгоритмами управления перегрузкой сети. Четвертый бит (3) определяет пропускную способность (нормальная или высокая). Пятый бит (4) определяет надежность доставки. Шестой и седьмой биты зарезервированы. Отметим, что программное обеспечение большинства рабочих станций и маршрутизаторов игнорирует тип сервиса.

Протокол IP обрабатывает каждую дейтаграмму в независимости от ее принадлежности к тому или иному пакету. При этом используются четыре основных механизма: установка типа сервиса, установка времени жизни, установка опций и вычисление контрольной суммы заголовка. Тип сервиса характеризует набор услуг, которые требуются от маршрутизаторов в распределенной сети. Эти параметры должны использоваться для управления выбором реальных рабочих характеристик при передаче дейтаграмм. В некоторых случаях передача дейтаграммы осуществляется с установкой приоритета, который дает данной дейтаграмме по сравнению с остальными некоторые преимущества при обработке. Тип сервиса определяется тремя показателями: малой задержкой при передаче, высокой достоверностью и большой пропускной способностью.

Поле «Время жизни». При определенных условиях IP-дейтаграммы могут попасть в замкнутый логический контур, образованный некоторой группой маршрутизаторов. Иногда такие логические контуры существуют в течение короткого промежутка времени, порой они оказываются достаточно долговечными. Чтобы избавить сеть от дейтаграмм, циркулирующих в таких логических контурах слишком долго, протоколом IP устанавливается предельный срок пребывания дейтаграммы в сети. Он задается в поле «Время жизни» — TTL (Time To Live). Его содержимое уменьшается на единицу при прохождении дейтаграммы через маршрутизатор; при обнулении поля TTL дейтаграмма отбрасывается.

Первоначально спецификации IP включали еще одно требование: поле TTL должно уменьшаться, по крайней мере, один раз в секунду. Поскольку поле TTL является восьмиразрядным, это означает, что дейтаграмма могла находиться в сети не более 4.25 мин. На практике требование ежесекундного уменьшения поля TTL игнорируется, тем не менее, в спецификациях многих протоколов следующих

уровней (TCP) по-прежнему предполагается, что максимальное время жизни дейтаграммы в сети составляет лишь две минуты.

Поле «Идентификатор» используется для распознавания дейтаграмм, образованных в результате фрагментации. Все фрагменты фрагментированного пакета данных должны иметь одинаковое значение этого поля.

Поле «Общая длина» указывает общую длину дейтаграммы (заголовок и поле данных). Максимальный размер дейтаграммы может составлять 65535 байт. В подавляющем большинстве сетей столь большой размер дейтаграмм не используется. По стандарту все устройства в сети должны быть готовы принимать дейтаграммы длиной 576 байт. Эти ограничения необходимы для передачи дейтаграмм в физических кадрах. Передача дейтаграммы в кадре называется инкапсуляцией. С точки зрения низших уровней дейтаграмма выглядит так же, как и любое другое сообщение в сети. Сетевое оборудование не работает с дейтаграммами, поэтому дейтаграмма является частью области данных кадра (рис. 6.2).

<table>
<tr><td></td><td>Заголовок IP-дейтаграммы</td><td>Область данных IP-дейтаграммы</td><td></td></tr>
<tr><td>Заголовок кадра канального уровня</td><td colspan="2">Область данных кадра</td><td>Контрольная сумма</td></tr>
</table>

Рис. 6.2. Инкапсуляция дейтаграммы в кадр

Функции фрагментации и сборки также возложены на протокол IP. *Фрагментация* — это разделение большой дейтаграммы на несколько небольших частей. В большинстве локальных и глобальных сетей есть ограничения на максимальный размер кадра. Эту величину называют максимальной единицей передачи (Maximum Transmission Unit, MTU). Например, в сетях Ethernet данная величина составляет 1500 байт, а в сетях FDDI — 4096 байт.

Когда маршрутизатор переправляет дейтаграмму из одной сети в другую, может оказаться, что ее размер окажется недопустимым в новой сети. Спецификация IP предусматривает следующее решение этой проблемы: маршрутизатор может разбить дейтаграмму на более мелкие фрагменты, приемлемые для выходной среды, а в пункте назначения эти фрагменты будут вновь объединены в дейтаграмму исходного вида. Формируемые маршрутизатором фрагменты идентифицируются смещением относительно начала исходной дейтаграммы. Дейтаграмма идентифицируется по отправителю, пункту назначения, типу протокола высокого уровня и 16-разрядному полю «Идентификатор». Все это в совокупности должно образовывать уникальную комбинацию.

Следует подчеркнуть связь между полями «Время жизни» и «Идентификатор». Действительно, во избежание смешивания фрагментов двух разных дейтаграмм источник IP-данных обязан исключить ситуацию, когда в один пункт назначения по одному и тому же протоколу в течение жизненного цикла дейтаграммы будут отправлены две дейтаграммы с совпадающими идентификаторами. В связи с тем, что идентификатор 16-разрядный, а наибольшее время жизни дейтаграммы исчисляется минутами (будем считать, что оно порядка 2 мин) получаем скорость передачи — 546 дейтаграмм в секунду. При максимальном размере дейтаграммы, равном 64 Кбайт, имеем общую скорость около 300 Мбит/с.

Проблема эффективного использования битов идентификатора оказалась практически разрешенной с появлением метода MTU Discovery, позволяющего определить значения MTU на всем пути к пункту назначения. Согласно этому методу конечная система может устанавливать в заголовке IP-дейтаграммы бит DF (Don't Fragment — не фрагментировать), запрещающий фрагментацию, ведь конечные системы могут заранее узнать о том, что отправляемые ими дейтаграммы имеют чрезмерную длину. Источник IP-трафика, устанавливающий бит DF теперь может не опасаться того, что две дейтаграммы перепутаются. Однако в сетевой среде, где технология MTU Discovery не применяется (в ней бит DF не несет функциональной нагрузки), необходимо предпринимать дополнительные меры для предотвращения подобной ситуации.

На рис. 6.3 показана процедура фрагментации и сборки дейтаграммы.

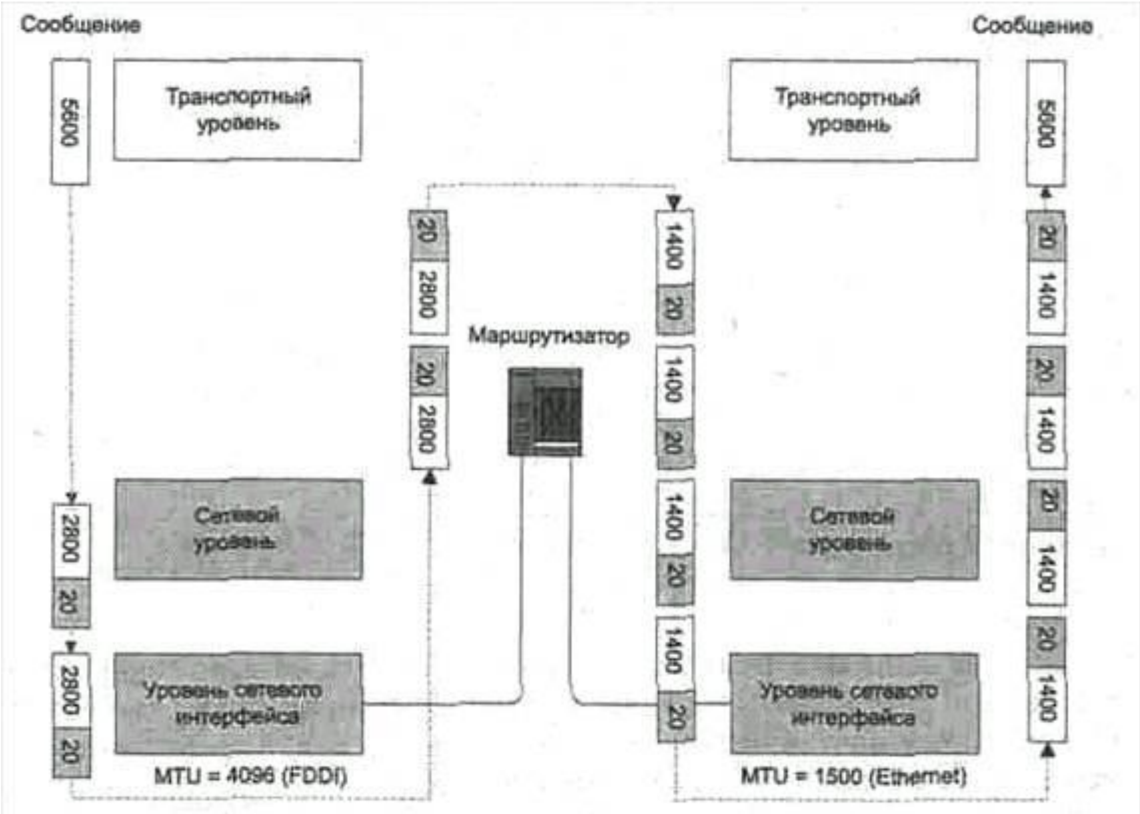


Рис. 6.3 Фрагментация дейтаграммы

Рассмотрим пример фрагментации. Предположим, отправителю необходимо передать сообщение длиной 5600 байт. Отправитель работает в сети, у которой значение MTU составляет 4096 байт. При поступлении пакета на сетевой уровень, протокол IP делит его на две равные дейтаграммы по 2800 байт, устанавливая в первой дейтаграмме признак фрагментации и присваивая пакету уникальный идентификатор. Бит фрагментации во второй дейтаграмме равен нулю, что указывает на последний фрагмент сообщения. Таким образом, дейтаграммы укладываются в кадр физического уровня данной сети (2800 байт данных + 20 байт заголовка меньше 4096 байт).

После маршрутизатора дейтаграммы необходимо передать в сеть с MTU, равным 1500 байт. Для этого маршрутизатор делит поступающие дейтаграммы пополам. Он формирует новые дейтаграммы, каждая из которых имеет размер 1400+20 байт, чтобы уложиться в MTU второй сети. Необходимо отметить, что маршрутизатор не собирает фрагменты в более крупные дейтаграммы, даже если на пути встречается сеть, допускающая такое укрупнение.

Фрагментация и сборка производятся автоматически, не требуя от отправителя специальных действий. Каждая фрагментированная часть исходной дейтаграммы имеет тот же формат. Использование фрагментации повышает вероятность потери исходной дейтаграммы, так как потеря даже одного фрагмента приводит к потере всей дейтаграммы. Сборка дейтаграммы осуществляется на месте назначения. Такой метод позволяет маршрутизировать фрагменты независимо.

Поля «Идентификатор», «Флаги» и «Смещение фрагмента» управляют фрагментацией и сборкой дейтаграммы.

Рассмотрим еще один пример фрагментации дейтаграммы с приведением конкретных значений полей заголовка дейтаграммы протокола дейтаграммы. На рис. 6.4 показан исходный заголовок дейтаграммы. Пусть MTU будет равно 280 байт. Допустим, что общая длина передаваемой дейтаграммы составляет 472 байта.

Номер версии = 4	Длина заголовка = 5	Тип сервиса	Общая длина = 472

Идентификатор =111	Флаги = 0	Смещение фрагмента = 0
Время жизни =123	Протокол = 6	Контрольная сумма заголовка
Адрес отправителя		
Адрес получателя		
Данные		

Рис. 6.4. Заголовок дейтаграммы до фрагментации

На рис. 6.5 показаны поля заголовков двух полученных в результате фрагментации дейтаграмм. Получающиеся дейтаграммы имеют максимальный размер поля данных, равный 256 байт.

Поле «Флаги» используется при фрагментации. Нулевой первый бит разрешает фрагментацию, единичный — запрещает. Единичный второй бит указывает на последний фрагмент дейтаграммы.

Поле «Смещение фрагмента» используется для указания смещения данных во фрагменте относительно начала исходной дейтаграммы. Чтобы получить смещение в байтах, надо значение этого поля умножить на 8. Первый фрагмент всегда имеет нулевое смещение. Поле используется при сборке фрагментов дейтаграммы после передачи по сетям с различными MTU**.**

Поле «Протокол» показывает, какому протоколу верхнего уровня принадлежит дейтаграмма. При поступлении дейтаграммы это поле указывает, какому приложению следует ее передать. В табл. 6.1 содержится перечень (неполный) протоколов.

Таблица 6.1

Заголовок фрагмента дейтаграммы #1

Номер версии = 4	Длина заголовка = 5	Тип сервиса	Общая длина = 276
Идентификатор =111	Флаги = 1	Смещение фрагмента = 0	
Время жизни =119	Протокол = 6	Контрольная сумма заголовка	
Адрес отправителя			
Адрес получателя			
Данные			

Заголовок фрагмента дейтаграммы #2

Номер версии = 4	Длина заголовка =5	Тип сервиса	Общая длина = 276
Идентификатор=111	Флаги = 1	Смещение фрагмента = 0	
Время жизни=119	Протокол = 6	Контрольная сумма заголовка	

Адрес отправителя
Адрес получателя
Данные

Рис. 6.5. Заголовки двух дейтаграмм после фрагментации

Таблица 6.1.

Значения поля «Протокол»

Знач. поля	Протокол	Пояснение
0	резерв	
1	ICMP	Internet Control Message Protocol, протокол управляющих сообщений
2	IGMP	Internet Group Management Protocol, протокол управления группами
4	IP	Инкапсуляция IP в IP
б	TCP	Transmission Control Protocol, протокол управления передачей
8	EGP	Exterior Gateway Protocol, внешний шлюзовый протокол
17	UDP	User Datagram Protocol, протокол пользовательских дейтаграмм
88	IGRP	Interior Gateway Routing Protocol, внутренний протокол маршрутизации
89	OSPF	Open Shortest Path First, «первый кратчайший путь»

Поле «Контрольная сумма» рассчитывается по всему заголовку. Так как некоторые поля заголовка меняют свое значение, например время жизни, при прохождении дейтаграммы через маршрутизаторы контрольная сумма проверяется и повторно рассчитывается при каждой модификации заголовка. Определение контрольной суммы заголовка обеспечивает безошибочность передачи дейтаграммы через сеть. Перед отправкой дейтаграммы вычисляется контрольная сумма, которая вносится в ее заголовок. При получении дейтаграммы вычисляется ее контрольная сумма, которая сравнивается со значением контрольной суммы в ее заголовке. При обнаружении ошибки в контрольной сумме дейтаграмма отбрасывается. Алгоритм вычисления контрольной суммы заголовка дейтаграммы протокола IPприменяется и во многих других протоколах, таких как UDP, TCP, ICMP и OSPF.

Поля «Адрес отправителя» и «Адрес получателя» имеют одинаковые длину и структуру. Поля содержат 32-битные IP-адреса отправителя и получателя дейтаграммы.

Поле «Опции» необязательно и обычно используется при настройке сети. В поле могут быть указаны точный маршрут прохождения дейтаграммы в распределенной сети, данные о безопасности, различные

временные отметки и т. д. Поле не имеет фиксированной длины, поэтому для выравнивания заголовка дейтаграммы по 32-битной границе предусмотрено следующее поле — **поле «Выравнивание».** Выравнивание осуществляется нулями.

Длина поля «Опции» меняется в зависимости от того, какие опции были выбраны. Опции в дейтаграмме размещаются друг за другом, без разделителей. Каждая опция состоит из кода опции (1 байт), за которым могут следовать длина опции (1 байт) и байты данных этой опции. На рис. 6.6 показан формат поля «Опции».

0	1	2	3	4	5	6	7
Копировать	Класс опции		Номер опции				

Рис. 6.6. Формат поля «Опции»

Байт кода опции делится на три поля: флаг «Копировать», «Класс опции» к «Номер опции». Флаг «Копировать» управляет тем, как маршрутизаторы учитывают опции при фрагментации дейтаграммы. Если бит установлен, опции должны копироваться во все фрагменты дейтаграммы. Если флаг не установлен опцию нужно скопировать только в первый фрагмент.

Поля «Класс опции» и «Номер опции» указывают класс опции и номер опции внутри этого класса (табл. 6.2 и 6.3).

Таблица 6.2.

Значения поля «Класс опции;

Значение поля	Пояснение
0	Управление дейтаграммами или сетью
1	Зарезервировано
2	Отладка сети
3	Зарезервировано

Из класса 3 применяется опция с номером 4. В нее записываются межсетевые временные метки. Они используются при протоколировании следования дейтаграммы по маршруту.

В настоящее время некоторые опции практически не используются. Например, опция «Безопасность» с номером 2 была разработана исключительно для нужд министерства обороны США, и в гражданских сетях не используется. Опция «Идентификатор потока» использовалась только в экспериментах с сетями Satnet и сейчас не встречается.

Таблица 6.3.

Номера опций класса 0

Номер опции	Длина	Пояснение

0	-	Конец списка опций. Используется, если опция не заканчивается в конце заголовка
1	-	Нет операций. Используется для выравнивания по 32-битной границе в списке опций
2	11	Безопасность
3	Переменная	Используется для маршрутизации дейтаграммы с учетом информации, предоставленной отправителем (маршрут однозначно не определен)
7	Переменная	Запись маршрута
8	4	Идентификатор маршрута. Используется для поддержки идентификации потока
9	Переменная	Используется для маршрутизации дейтаграммы с учетом информа-ции, предоставленной отправителем (маршрут определен однозначно)
Другой	-	Не используется

7. ЗАГОЛОВОК ДЕЙТАГРАММЫ IP v.6

Работа по расширению протокола IP была начата в 1992 году. Необходимость этого диктовалось тем, что практически все ресурсы старой версии протокола IP (IPv4) были исчерпаны. Быстрый рост сетиInternet привел к появлению дефицита IP-адресов. Возросший трафик начал вызывать перегрузки магистральных маршрутизаторов. Изменился и характер передаваемого трафика. Все большую долю в нем стали занимать мультимедийные данные.

Новая версия протокола IP – версия 6 (IPv6) – была принята организацией IETF в 1995 году. Она описана в документе RFC 1752. В настоящее время осуществляется постепенный переход к протоколу IPv6. Существует несколько фрагментов сети Internet, в которых маршрутизаторы поддерживают обе версии IP. Эти фрагменты объединены между собой и образуют так называемую «шестую» магистраль (6bone). Для того чтобы передать дейтаграммы протокола IPv6, магистраль 6 bone инкапсулирует их в дейтаграмму IP и передает через те части сети Internet, которые не поддерживают новую версию протокола. Этот процесс называется *туннелированием.* Следует помнить, что появление дополнительного заголовка при туннелировании ведет к росту сетевого трафика. Документ RFC 1933 определяет четыре конфигурации туннелей между рабочими станциями и маршрутизаторами:

- маршрутизатор–маршрутизатор;
- рабочая станция–маршрутизатор;
- рабочие станции–маршрутизаторы;
- маршрутизатор–рабочая станция.

На рис. 7.1 показан пример организации механизма туннелирования для конфигурации маршрутизатор–маршрутизатор.

Другим методом, позволяющим осуществить плавный переход на новую версию, является использование двойных стеков.

Двойные стеки позволяют узлу в сети IP поддерживать обе версии протокола.

Такие узлы называются IPv6/IPv4-узлами. Использование двойного стека позволяет раздельно переводить на протокол IPv6 каждое устройство в сети. При этом необходимо задействовать дополнительные ресурсы такого устройства, изменить его конфигурационную информацию и провести ряд других операций.

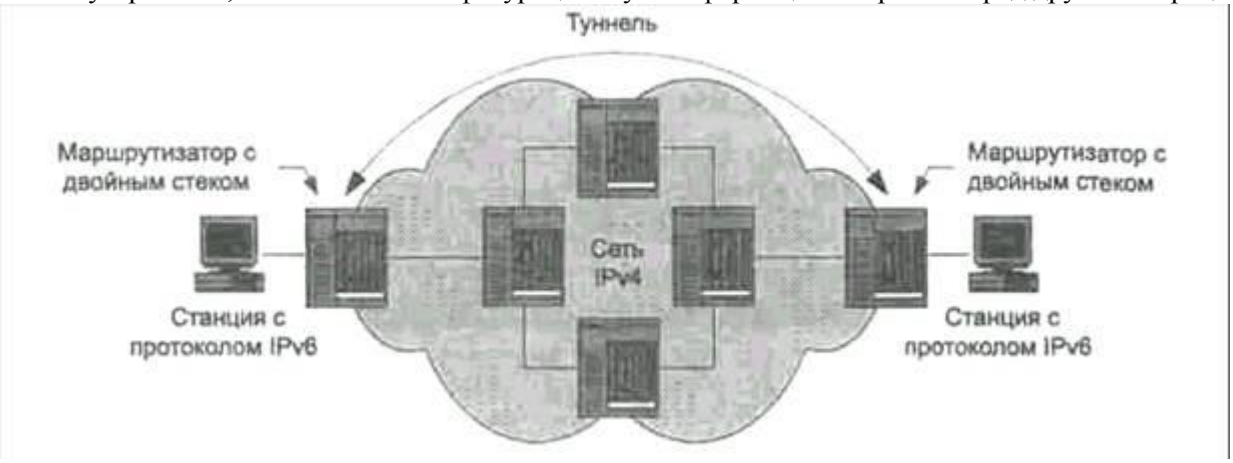


Рис. 7.1 Туннелирование дейтаграммы IPv6, инкапсулированной в дейтаграмме IPv4

Нужно учитывать, что маршрутизаторам может потребоваться дополнительная оперативная память, так как таблицы маршрутизации протокола IPv6 больше по объему. На рис.7.2 показано распределение уровней узла с двойным стеком IPv4/IPv6.

Прикладной уровень	
Транспортный уровень (протоколы TCP и UDP)	
IPv4	IPv6
Уровень сетевого интерфейса	

Рис. 7.2 Уровни двойного стека TCP/IP

Протокол IPv6 поддерживается практически всеми современными операционными системами и производителями сетевого оборудования. Естественно, развитие протокола IP повлекло за собой модернизацию всего стека TCP/IP. Уменьшился объем маршрутной и служебной информации. Много внимания уделено функциональным составляющим TCP/IP, которые напрямую влияют на загрузку маршрутизаторов:

- реализована гибкая схема разделения адресного пространства с использованием технологий CIDR и масок подсетей переменной длины. Изменение адресной схемы позволяет сократить объем таблиц маршрутизации и ускорить их просмотр и обновление;
- введено повсеместное использование (физического адреса устройства в качестве номера узла. При этом снижается нагрузка на сеть за счет отказа от протокола ARP;
- уменьшен заголовок дейтаграмм IP;
- проведение фрагментации перенесено на конечные узлы. Узлы, поддерживающие протокол IPv6, сами определяют размер MTU, который устраивает все транзитные узлы и каналы на пути следования дейтаграммы.

Схема адресации IPv6 существенно отличается от схемы адресации протокола IP. Адреса получателя и отправителя в протоколе IPv6 задаются 128 битами. Такая длина адресного пространства позволяет на достаточно большой период времени снять проблему дефицита адресов в сети Internet. Основным

механизмом, заложенным в схему адресации протокола IPv6, является введение иерархического разделения адресного пространства на уровни. Вместо прежних двух уровней – номера сети и номера устройства, – в протоколе IPv6 используется пять уровней, включая два уровня идентификации провайдеров и три уровня идентификации абонентов в сети (рис. 7.3)

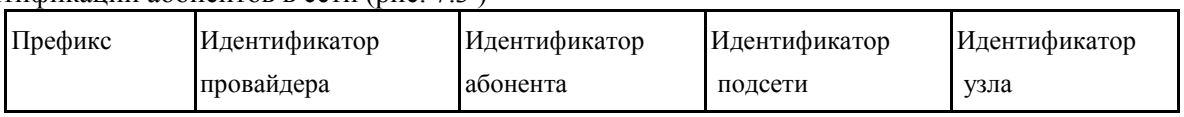


Рис. 7.3 Уровни адресации протокола IPv6

Префикс определяет тип используемого адреса. Приведем пример адреса с идентификацией провайдера. Такой адрес имеет префикс 010. Этот префикс выбран согласно табл. 7.1, в которой приведено исходное распределение адресов протокола IPv6. В табл. 7.1 строка с адресом идентификации провайдера выделена.

Таблица 7.1.

Исходное распределение адресов IPv6

Назначение блока адресов	Двоичный префикс	Доля адресного пространства
Резервный	0000 0000	1/256
Незанятый	0000 000	11/258
Зарезервирован для IPX	0000 010	1/128
Незанятый	0000 011	1/128
Незанятый	0000 1	1/32
Незанятый	0001	1/16
Незанятый	001	1/8
Адреса идентификации провайдера	010	1/8
Незанятый	011	1/8
Зарезервирован для адресов по географической принадлежности	100	1/8
Незанятый	101	1/8
Незанятый	110	1/8
Незанятый	1110	1/16
Незанятый	11110	1/32
Незанятый	1111 10	1/64
Незанятый	1111 110	1/128

Незанятый	1111 1110 0	1/512
Локальные адреса для линии	1111 1110 10	1/1024
Локальные адреса для узла	1111 1110 11	1/1024
Групповые адреса	1111 1111	1/256

На рис. 7.4 показан формат адреса с идентификацией провайдера.

Пре-фикс	Идентифи-катор орга-низации	Идентифи-катор провайдера	Зарезерви-ровано	Идентификатор абонента	Зарезерви-ровано	Адрес сети и устройств
010	5 бит	16 бит	8 бит	24 бита	8 бит	64 бита

Рис. 7.4 Формат адреса с префиксом 010

Поле «Идентификатор организации» определяет организацию, ответственную за выделение адресов провайдерам. Например, InterNIC в Северной Америке, RIPE в Европе или APNIC в Азии. Поле «Идентификатор провайдера» определяет непосредственно провайдера. Провайдер назначает поле «Идентификатор абонента». Данное поле идентифицирует конкретного пользователя. За полями «Идентификатор провайдера» и «Идентификатор абонента» следуют резервные поля, необходимые для будущего расширения. Оставшиеся 64 бита в адресе по принадлежности к провайдеру определяют номер сети и номер устройства. Идентификация сети и устройства происходит примерно по тем же правилам, что и в случае использования протокола IPv4 с классами A, B и C. Данное поле предоставляет достаточно пространства для разбиения выделенного блока адресов на адреса подсетей и рабочих станций в каждой подсети.

Такая структура адреса по принадлежности к провайдеру значительно упрощает маршрутизацию. Поле «Идентификатор провайдера» сразу определяет сеть другого провайдера. После определения сети провайдера маршрутизатор анализирует поле «Идентификатор абонента» и определяет непосредственного абонента, которому должна быть передана информация. Абонентом может выступать любая организация, которая, при необходимости, может организовать несколько уровней иерархии в своей сети.

В протоколе IPv6 отменено разделение адресов на классы. В основе распределения адресного пространства лежит технология CIDR. При этом адреса сетей каждого провайдера имеют одинаковое значение сетевого префикса и все устройства в этой сети поддерживают его передачу. С использованием технологии CIDR деление IP-адреса на адрес подсети и адрес устройства производится на основе маски подсети переменной длины, которая назначается провайдером и уже не зависит от класса адреса. Использование технологии CIDR совместно с технологией «наибольшего совпадения», реализуемой маршрутизаторами, позволяет значительно уменьшить объем их таблиц маршрутизации. Технология CIDR уже используется с четвертой версией протокола IP и поддерживается протоколами маршрутизации OSPF, RIP-2 и BGP-4. Эта технология позволяет избежать избыточного назначения адресного пространства, которое может иметь место при использовании классов адресов.

Протокол IPv6 вводит несколько типов адресов:

Unicast – индивидуальный (единичный) адрес. Адрес определяет отдельное устройство в сети или порт маршрутизатора. В свою очередь, индивидуальный адрес подразделяется на:

Global – глобальный. Основной тип адресов в Internet;

Link-Local и Site-Local – адреса для линии и узла. Адреса используются в сетях, не связанных с Internet. Для того чтобы эти адреса можно было использовать в Internet, поле «Идентификатор провайдера» заполняется нулями. Термин «Link» относится к сетям Frame Relay и ATM, то есть к прямой выделенной линии или соединению с сетью Ethernet, FDDI и т. д. Локальный адрес для линии описывает устройство, не имеющее соединения с маршрутизатором или с Internet. С использованием этих адресов можно подключать к Internet сети без присвоения им новых адресов. На рис. 7.5 показаны примеры локальных адресов для линий и узлов. В формате локального адреса для линии поле уникального адреса линии содержит физический адрес локальной сети. Префикс имеет длину 10 бит, а оставшаяся часть – 118 бит. Если локальной адрес для линии используется для подключения к сети Ethernet, то физический адрес займет 48 бит (6 байт). Таким образом, использование локальных адресов для линий позволяет подключать сети Ethernet, Token Ring и FDDI без реконфигурации сетевых адресов.

Префикс (1111 1110 10)	00 ... 00	Уникальный адрес линии
	...	
Префикс (1111 1110 11)	00 ... 00	Уникальный адрес узла

Рис. 7.5 Адреса для линии и узла

В каждом из адресов префикс необходимо дополнить нулями, число которых определяется требуемым остатком бит для задания адреса линии или узла. Адреса для узла присваиваются узлам, имеющим маршрутизаторы, но не подключенным к сети Internet через провайдера. При подключении этого узла к сети Internet маршрутизатор настраивается с новым префиксом. По сути дела, маршрутизатор создает адрес по принадлежности к провайдеру открытием полей «Идентификатор организации», «Идентификатор провайдера» и «Идентификатор абонента».

Multicast – адрес набора узлов (групповой адрес). В протоколе IPv6 отсутствует понятие широковещательного адреса. Широковещательная адресация заменена поддержкой групповой передачи данных. Такой механизм необходим протоколу IPv6 для регулирования пропускной способности сети при распространении мультимедийного трафика;

Anycast – адрес набора узлов. Этот тип адресов используется для обеспечения прохождения своего трафика через маршрутизаторы отдельных провайдеров. В отличие от групповых адресов, такая дейтаграмма должна быть доставлена любому члену группы. В протоколе IPv6 широко используется маршрутизация от источника. Этот вид маршрутизации освобождает маршрутизаторы от функции анализа своих таблиц маршрутизации, уменьшает время задержки дейтаграммы для ее обработки и, естественно, повышает пропускную способность сети в целом;

При назначении адресов каждому порту маршрутизатора вместе с физическим адресом присваивается еще один адрес, общий для всех портов всех маршрутизаторов в сети данного провайдера, который является any-cast-адресом;

При указании нечеткого адреса устройству не надо знать конкретный адрес маршрутизатора, так как оно является членом группы маршрутизаторов с этим адресом. Положительным моментом здесь является то, что в случае изменения местоположения этого маршрутизатора адрес его менять не надо, так как дейтаграмма будет отправляться по-прежнему ближайшему члену этой нечеткой группы.

Для плавного перехода к протоколу IPv6 введен специальный тип адресов – IPv4-compatible (совместимые адреса). В этих адресах старшие 96 бит содержат нули, а младшие 32 бита – обычный адрес IPv4. Такие адреса позволяют решить проблему совместимости частей Internet, работающих с протоколами IP разных версий.

Для упрощения обработки заголовка дейтаграммы в протоколе IPv6 введены основной и дополнительный заголовки. Основной заголовок присутствует всегда и имеет длину 40 байт (рис. 7.6). Дополнительный заголовок определяет некоторые необязательные параметры.

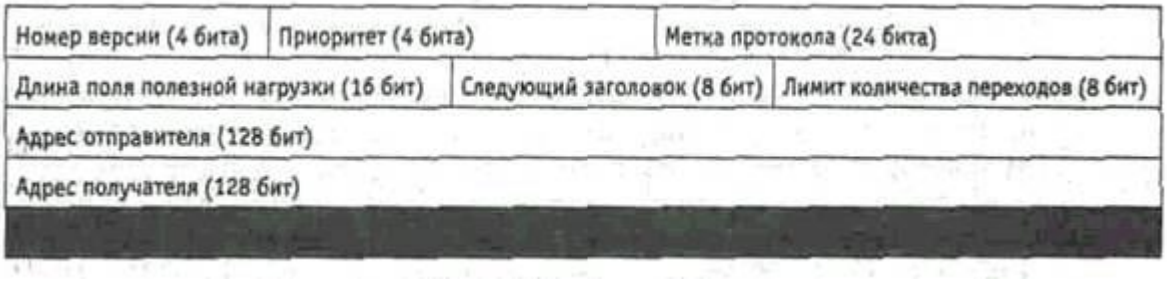


Рис. 7.6 Формат основного заголовка

Поле «Следующий заголовок» по своему назначению соответствует полю «Протокол» в версии 4 и определяет тип заголовка, который следует за данными. Каждый следующий дополнительный заголовок содержит это поле. Если дейтаграмма не содержит дополнительных заголовков, то значение этого поля определяет протокол – TCP, UDP, RIP или OSPF. Поле «Лимит переходов» введено для более эффективного определения времени жизни дейтаграмм. В протоколе IPv4 поле времени жизни дейтаграммы уменьшается на единицу (по крайней мере) при прохождении каждого маршрутизатора. При этом время ожидания в очереди не учитывается. Поле «Приоритет» позволяет отправителю задать приоритет своих дейтаграмм. Возможные 16 значений этого поля разделены на две категории: значения от 0 до 7 определяют трафик, которым маршрутизатор при необходимости может пренебречь, а значения от 8 до 15 указывают на трафик, к которому эти меры применяться не могут (аудио- и видеоинформация, передаваемая с постоянной скоростью в реальном времени). Используя поля «Приоритет» и «Метка потока» устройства могут идентифицировать дейтаграммы, которым требуется нестандартное обслуживание на маршрутизаторах. В протоколе IPv6 определены следующие типы дополнительных заголовков:

Routing – определяет полный маршрут при маршрутизации от источника. Данный заголовок позволяет отправителю указать список IP-адресов, которые диктуют путь передачи;

Fragmentation – содержит сведения о проведении фрагментации на конечных узлах сети. В протоколе IPv6 фрагментацию не разрешается выполнять на промежуточных узлах; это значительно повышает производительность при маршрутизации. В том случае, если распределенная сеть состоит из сегментов с различными значениями MTU, отправитель использует дополнительный заголовок Fragmentationдля разделения дейтаграммы на произвольное число небольших фрагментов. В этом дополнительном заголовке содержатся поля, которые идентифицируют фрагменты исходной дейтаграммы по присвоенным им последовательным номерам. Так как промежуточные маршрутизаторы не выполняют фрагментацию, то вся ответственность за выбор правильного размера дейтаграммы возлагается на отправителя, которому необходимо определить значения MTU каждой промежуточной сети в пути до получателя. Например, если две сети FDDI со значением MTU 4500 байт связываются через сетьEthernet с MTU, равным 1500 байт, то отправитель должен посылать дейтаграмму с длиной не более 1500 байт или разделить ее на фрагменты той же длины. Конечные системы могут определять минимальное значение MTU в пути между ними, используя механизм MTU path discovery process (процесс выяснения значений MTU на пути), описанный в документе RFC 1191. При этом отправитель посылает дейтаграмму с длиной, равной значению MTU той сети, к которой он подключен. Если выбранный размер дейтаграммы слишком велик для некоторых промежуточных сетей, то отправителю будет послано сообщение протокола ICMP «Datagram Too Big» (Дейтаграмма слишком велика) с указанием рекомендованного значения MTU. После получения этого сообщения отправитель скорректирует размер дейтаграммы (например, с помощью фрагментации) и повторит процесс. Это будет продолжаться до тех пор, пока дейтаграмма не сможет пройти все промежуточные сети в пути до получателя;

Authentication – служит для идентификации конечных узлов и обеспечения целостности дейтаграмм;

Encryption – служит для шифрования и дешифровки передаваемых данных;

Hop-by-Hop Option – используется алгоритмом «переход за переходом» при обработке дейтаграмм. Данный заголовок переносит дополнительные параметры, которые проверяются промежуточными узлами. Заголовок должен следовать первым после основного заголовка. Так как заголовок проверяется всеми маршрутизаторами, его полезно использовать для передачи управляющей или отладочной информации. В настоящее время может использоваться параметр Router Alert, который информирует маршрутизаторы, что дейтаграмма должна быть обработана целиком до ее передачи следующему маршрутизатору в пути. Данный параметр применяется, например, при работе протокола RSVP;

Destination Option – содержит дополнительную информацию для узла назначения.

Каждый дополнительный заголовок содержит тип следующего за ним заголовка, что позволяет создать цепочку заголовков. Основной заголовок является первым в цепочке и не содержит дополнительных заголовков. Поле «Следующий заголовок» указывает, какой дополнительный заголовок следует за основным. Поле «Следующий заголовок» первого дополнительного заголовка указывает на тип второго дополнительного заголовка и т. д. Это продолжается до тех пор, пока в поле «Следующий заголовок» очередного дополнительного заголовка не встретится запись о том, что далее, например, следует заголовок протокола TCP (рис7.7).

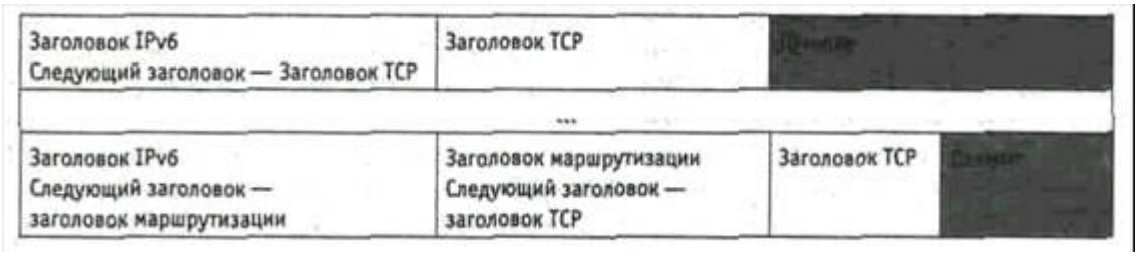


Рис 7.7 Формирование цепочки заголовков

Для поддержки качества обслуживания протокол IPv6 работает с «меткой потока» (flow label). Метка потока – это признак, который размещается в поле заголовка «Метка протокола» дейтаграммы IPv6. Метка указывает на принадлежность данной дейтаграммы к последовательности дейтаграмм – потоку, который требует определенных параметров обслуживания. Маршрутизаторы обрабатывают потоки на основании значения метки и идентификатора отправителя дейтаграмм. Для предоставления нестандартного качества обслуживания потоков разработан дополнительный протокол RSVP – протокол резервирования ресурсов.

В отличие от протокола IPv4, механизмы разрешения адресов и их настройки в протоколе IPv6 реализованы на сетевом уровне, а не на канальном (протоколы ARP для широковещательных сетей иATMARP и NHRP для сетей ATM). То есть, протокол обнаружения (Neighbor Discovery), который используется для нахождения маршрутизаторов и соседей, является интегральной частью IPv6 и любые механизмы, предназначенные для адаптации технологии ATM к IPv6, прежде всего, должны работать именно с этим протоколом. Здесь наблюдается принципиальное отличие от протокола IPv4, в котором механизмы разрешения адресов не являются частью основного протокола, а реализуются на канальном уровне – протоколом ARP для широковещательных сетей и механизмами ATMARP и NHRP для сетей ATM.

8 ФУНКЦИИ СЕТЕВОЙ МАРШРУТИЗАЦИИ

8.1 Таблицы маршрутизации.

Как модуль IP узнает, какой именно сетевой интерфейс нужно использовать для отправления IP-пакета? Модуль IP осуществляет поиск в таблице маршрутов. Ключом поиска служит номер IP-сети, выделенный из IP-адреса места назначения IP-пакета.

Таблица маршрутов (рис. 8.1) содержит по одной строке для каждого маршрута.

Сеть назначения	Флаг вида маршрутиз.	Шлюз	Интерфейс (выход)	…….	Метрика
Имя/номер	Прямая/ косвенная	Имя/ адрес	номер	…….	Взвешенное расстояние

Рис. 8.1 Таблица маршрутизации (сокращенная)

Основными столбцами таблицы маршрутов являются номер сети, флаг прямой или косвенной маршрутизации, IP-адрес шлюза и номер сетевого интерфейса.

Эта таблица используется модулем IP при обработке каждого отправляемого IP-пакета.

В большинстве систем таблица маршрутов может быть изменена с помощью команды "route". Содержание таблицы маршрутов определяется менеджером сети, поскольку менеджер сети присваивает машинам IP-адреса.

8.2. Прямая IP-маршрутизация.

На рис.8. показана небольшая IP-сеть, состоящая из 3 машин: A, B и C.

Каждая машина имеет такой же стек протоколов TCP/IP как на рис.5.1. Каждый сетевой адаптер этих машин имеет свой Ethernet-адрес. Менеджер сети должен присвоить машинам уникальные IP-адреса.

Сеть development

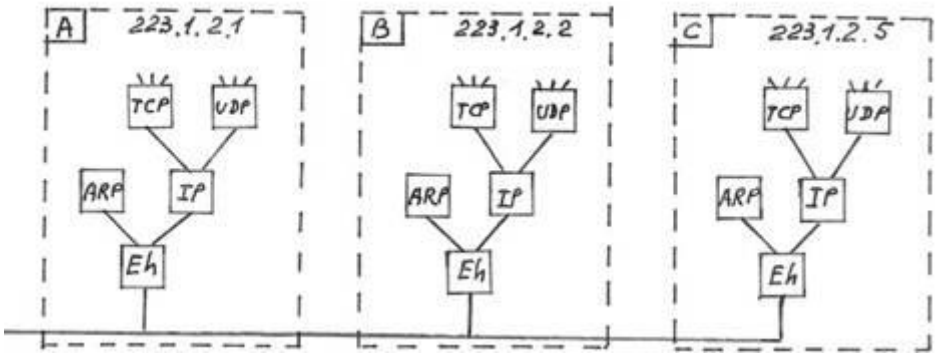


Рис. 8.2 Простая IP-сеть

Когда A посылает IP-пакет B, то заголовок IP-пакета содержит в поле отправителя IP-адрес узла A, а заголовок Ethernet-кадра содержит в поле отправителя Ethernet-адрес A. Кроме этого, IP-заголовок содержит в поле получателя IP-адрес узла B, а Ethernet-заголовок содержит в поле получателя Ethernet-адрес B.

Табл.8.1.

Адреса в Ethernet-кадре, передающем IP-пакет от A к B

адрес	отправитель	получатель
IP-заголовок	A	B
Ethernet-заголовок	A	B

В этом простом примере протокол IP является излишеством, которое мало что добавляет к услугам, предоставляемым сетью Ethernet. Однако протокол IP требует дополнительных расходов на создание, передачу и обработку IP-заголовка. Когда в машине B модуль IP получает IP-пакет от машины A, он сопоставляет IP-адрес места назначения со своим и, если адреса совпадают, то передает датаграмму протоколу верхнего уровня.

В данном случае при взаимодействии A с B используется прямая маршрутизация.

Рассмотрим более подробно, как происходит маршрутизация в одной физической сети.(см. рис 8.2)

Таблица маршрутов (табл. 8.2) в узле A выглядит так:

Табл.8.2.

Пример таблицы маршрутов

Сеть назначения	Флаг вида маршрутиз.	Шлюз	Интерфейс (выход)	…….	Метрика
B	Прямая	<пусто>	1	…….	0

В данном простом примере все узлы сети имеют одинаковые таблицы маршрутов.

Для сравнения ниже (табл. 8.3) представлена та же таблица, но вместо названия сети указан ее номер.

Табл.8.3.

Пример таблицы маршрутов с номерами сетей

Сеть назначения	Флаг вида маршрутизации	Шлюз	Интерфейс (выход)	…….	Метрика
223.1.2	Прямая	<пусто>	1	…….	0

Процедура прямой маршрутизации

Узел A посылает IP-пакет узлу B. Этот пакет находится в модуле IP узла alpha, и IP-адрес места назначения равен IP-адресу beta(223.1.2.2). Модуль IP с помощью маски подсети выделяет номер сети из IP-адреса и ищет соответствующую ему строку в таблице маршрутов. В данном случае подходит первая строка.

Остальная информация в найденной строке указывает на то, что машины этой сети доступны напрямую через интерфейс номер 1. С помощью ARP-таблицы выполняется преобразование IP-адреса в соответствующий Ethernet-адрес, и через интерфейс 1 Ethernet-кадр посылается узлу B.

Если прикладная программа пытается послать данные по IP-адресу, который не принадлежит сети development, то модуль IP не сможет найти соответствующую запись в таблице маршрутов. В этом случае модуль IP отбрасывает IP-пакет. Некоторые реализации протокола возвращают сообщение об ошибке "Сеть не доступна".

8.3. Косвенная маршрутизация

На рис.8.3 представлена более реалистичная картина сети Internet. В данном случае сеть Internet состоит из трех сетей Ethernet, на базе которых работают три IP-сети, объединенные шлюзом D. Каждая IP-сеть включает четыре машины; каждая машина имеет свои собственные IP- и Ethernet-адреса.

За исключением D все машины имеют стек протоколов, аналогичный показанному на рис.1. Шлюз D соединяет все три сети и, следовательно, имеет три IP-адреса и три Ethernet-адреса. Машина D имеет стек протоколов TCP/IP, похожий на тот, что показан на рис.5.1, но вместо двух модулей ARP и двух драйверов, он содержит три модуля ARP и три драйвера Ethernet.

Обратим внимание на то, что машина D имеет только один модуль IP.

Менеджер сети присваивает каждой сети Ethernet уникальный номер, называемый IP-номером сети. На рис.8.3 IP-номера не показаны, вместо них используются имена сетей.

Когда машина A посылает IP-пакет машине B, то процесс передачи идет в пределах одной сети. При всех взаимодействиях между машинами, подключенными к одной IP-сети, используется прямая маршрутизация, обсуждавшаяся в предыдущем примере.

Когда машина D взаимодействует с машиной A, то это прямое взаимодействие. Когда машина D взаимодействует с машиной E, то это прямое взаимодействие. Когда машина D взаимодействует с машиной H, то это прямое взаимодействие. Это так, поскольку каждая пара этих машин принадлежит одной IP-сети.

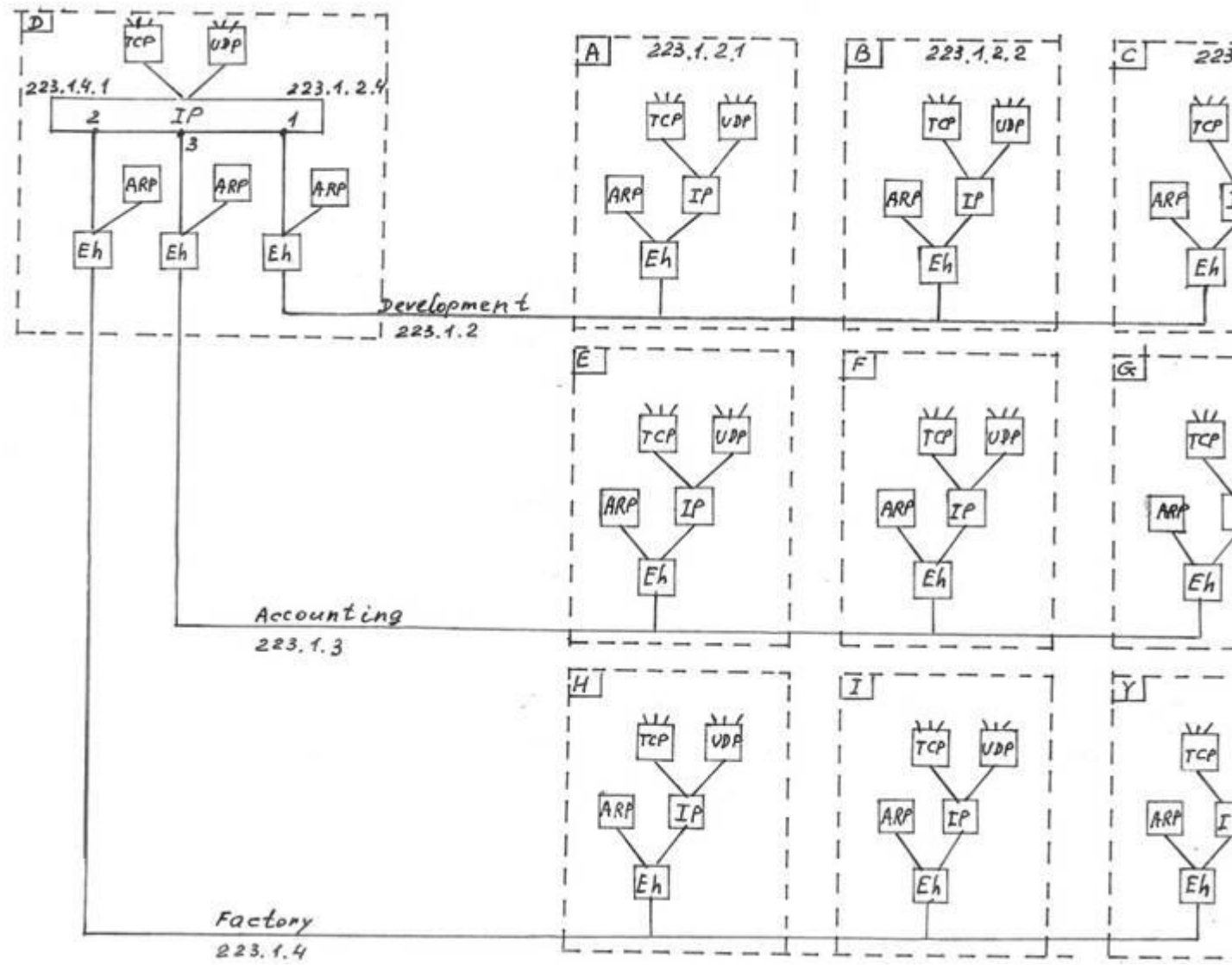


Рис.8.3. Сеть Internet, состоящая из трех IP-сетей

Однако, когда машина A взаимодействует с машинами, включенными в другую IP-сеть, то взаимодействие уже не будет прямым. Машина A должна использовать шлюз D для ретрансляции IP-пакетов в другую IP-сеть. Такое взаимодействие называется "косвенным".

Маршрутизация IP-пакетов выполняется модулями IP и является прозрачной для модулей TCP, UDP и прикладных процессов.

Если машина A посылает машине E IP-пакет, то IP-адрес и Ethernet-адрес отправителя соответствуют адресам A. IP-адрес места назначения является адресом E, но поскольку модуль IP в A посылает IP-пакет через D, Ethernet-адрес места назначения является адресом D.

Табл.8.4.

Адреса в Ethernet-кадре, передающем IP-пакет от A к E через шлюз D.

адрес	отправитель	получатель
IP-заголовок	A	E
Ethernet-заголовок	A	D

Модуль IP в машине D получает IP-пакет и проверяет IP-адрес места назначения. Определив, что это не его IP-адрес, шлюз D посылает этот IP-пакет прямо к E.

Табл.8.5.

Адреса в Ethernet-кадре, содержащем IP-пакет от A к E (после шлюза D)

адрес	отправитель	получатель
IP-заголовок	A	E
Ethernet-заголовок	D	E

Итак, при прямой маршрутизации IP- и Ethernet-адреса отправителя соответствуют адресам того узла, который послал IP-пакет, а IP- и Ethernet-адреса места назначения соответствуют адресам получателя. При косвенной маршрутизации IP- и Ethernet-адреса не образуют таких пар.

В данном примере сеть Internet является очень простой. Реальные сети могут быть гораздо сложнее, так как могут содержать несколько шлюзов и несколько типов физических сред передачи. В приведенном примере несколько сетей Ethernet объединяются одним шлюзом для того, чтобы локализовать широковещательный трафик в каждой сети.

Процедура косвенной маршрутизации

Рассмотрим более сложный порядок маршрутизации в IP-сети, изображенной на рис.8.3.

Таблица маршрутов (табл.8.6.) в узле A выглядит так:

Табл.8.6.

Таблица маршрутов в узле A

Сеть назначения	Флаг вида маршрутизации	Шлюз	Интерфейс (выход)	…….	Метрика
development	Прямая	<пусто>	1	…….	0
accounting	косвенная	D	1	……..	1
factory	косвенная	D	1	…….	1

Та же таблица в узле A с IP-адресами вместо названий (табл.8.7).

Табл.8.7.

Таблица маршрутов в узле A

Сеть назначения	Флаг вида маршрутизации	Шлюз	Интерфейс (выход)	…….	Метрика
223.1.2	Прямая	<пусто>	1	…….	0
223.1.3	косвенная	223.1.2.4	1	……..	1
223.1.4	косвенная	223.1.2.4	1	…….	1

В столбце "шлюз" таблицы маршрутов узла A указывается IP-адрес точки соединения узла D с сетью development.

Процедура косвенной маршрутизации

Узел A посылает IP-пакет узлу E. Этот пакет находится в модуле IP узла A, и IP-адрес места назначения равен IP-адресу узла e (223.1.3.2). Модуль IP выделяет сетевой номер из IP-адреса (223.1.3) и ищет соответствующую ему строку в таблице маршрутов. Соответствие находится во второй строке.

Запись в этой строке указывает на то, что машины требуемой сети доступны через шлюз D. Модуль IP в узле A осуществляет поиск в ARP-таблице, с помощью которого определяет Ethernet-адрес, соответствующий IP-адресу D. Затем IP-пакет, содержащий IP-адрес места назначения E, посылается через интерфейс 1 шлюзу D.

IP-пакет принимается сетевым интерфейсом в узле D и передается модулю IP. Проверяется IP-адрес места назначения, и, поскольку он не соответствует ни одному из собственных IP-адресов D, шлюз решает ретранслировать IP-пакет.

Модуль IP в узле D выделяет сетевой номер из IP-адреса места назначения IP-пакета (223.1.3) и ищет соответствующую запись в таблице маршрутов. Таблица маршрутов в узле D (табл.8.8) выглядит так:

Табл.8.8.

Таблица маршрутов в узле D

Сеть назначения	Флаг вида маршрутизации	Шлюз	Интерфейс (выход)	…….	Метрика
development	Прямая	<пусто>	1	…….	0
accounting	Прямая	<пусто>	3	……..	0
factory	Прямая ая	<пусто>	2	…….	0

Та же таблица с IP-адресами в узле D (табл. 8.9) вместо названий.

Табл.8.9.

Таблица маршрутов в узле D

Сеть назначения	Флаг вида маршрутизации	Шлюз	Интерфейс (выход)	…….	Метрика
223.1.2	Прямая	<пусто>	1	…….	0
223.1.3	Прямая	<пусто>	3	……..	0
223.1.4	Прямая ая	<пусто>	2	…….	0

Соответствие находится во второй строке. Теперь модуль IP напрямую посылает IP-пакет узлу E через интерфейс номер 3. Пакет содержит IP- и Ethernet-адреса места назначения равные E.

Узел E принимает IP-пакет, и его модуль IP проверяет IP-адрес места назначения. Он соответствует IP-адресу E, поэтому содержащееся в IP-пакете сообщение передается протокольному модулю верхнего уровня.

8.4 Формирование таблиц IP-маршрутизации

Установка маршрутов

До сих пор рассматривались функции использования таблиц маршрутизации для передачи IP-пакетов сети. Далее рассмотрим методы, позволяющие поддерживать корректность таблиц маршрутов.

Фиксированные маршруты - простейший способ проведения маршрутизации состоит в установке маршрутов при запуске системы с помощью специальных команд. Этот метод можно применять в относительно маленьких IP-сетях, в особенности, если их конфигурации не часто меняются.

На практике большинство машин автоматически формирует таблицы маршрутов. Например, UNIX добавляет записи о IP-сетях, к которым есть непосредственный доступ. Стартовый файл может содержать команды:

```
ifconfig ie0 128.6.4.4 netmask 255.255.255.0
ifconfig ie1 128.6.5.35 netmask 255.255.255.0
```

Они показывают, что существуют два сетевых интерфейса, и устанавливают их IP-адреса. Система может автоматически создать две записи в таблице маршрутов (табл 8.10):

Табл.8.10.

Автоматически создаваемые записи маршрутов

Сеть назначения	Флаг вида маршрутизации	Шлюз	Интерфейс (выход)	…….	Метрика
128.6.4	Прямая	<пусто>	0	…….	0
128.6.5	Прямая	<пусто>	1	……..	0

Эти записи определяют, что IP-пакеты для локальных подсетей 128.6.4 и 128.6.5 должны посылаться через указанные интерфейсы.

В стартовом файле могут быть команды, определяющие маршруты доступа к другим IP-сетям. Например,

```
route add 128.6.2.0 128.6.4.1 1
route add 128.6.6.0 128.6.5.35 0
```

Эти команды показывают, что в таблицу маршрутов должны быть добавлены две записи. Первый адрес в командах является IP-адресом сети, второй адрес указывает шлюз, который должен использоваться для доступа к данной IP-сети, а третий параметр является метрикой. Метрика показывает, на каком "расстоянии" находится описываемая IP-сеть. В данном случае метрика - это количество шлюзов на пути между двумя IP-сетями. Маршруты с метрикой 1 и более определяют первый шлюз на пути к IP-сети. Маршруты с метрикой 0 показывают, что никакой шлюз не нужен - данный маршрут задает дополнительный сетевой номер локальной IP-сети.

Таким образом, команды, приведенные в примере, показывают, что для доступа к IP-сети 128.6.2 должен использоваться шлюз 128.6.4.1, а IP-сеть 128.6.6 - это просто дополнительный номер для физической сети, подключенной к интерфейсу 128.6.5.35.

Табл. 8.11

Записи, добавляемые в таблицу маршрутов

Сеть назначения	Флаг вида маршрутизации	Шлюз	Интерфейс (выход)	…….	Метрика
128.6.2	косвенная	128.6.4.1	0	…….	0
128.6.6	Прямая	<пусто>	1	……..	0

Можно определить маршрут по умолчанию, который используется в тех случаях, когда IP-адрес места назначения не встречается в таблице маршрутов явно. Обычно маршрут по умолчанию указывает IP-адрес шлюза, который имеет достаточно информации для маршрутизации IP-пакетов со всеми возможными адресами назначения.

Если рассматриваемая IP-сеть имеет всего один шлюз, тогда все, что нужно сделать, - это установить единственную запись в таблице маршрутов, указав этот шлюз как маршрут по умолчанию. После этого можно не заботиться о формировании маршрутов в других узлах. (Конечно, сам шлюз требует больше внимания.)

9. МАРШРУТИЗАЦИЯ ПРОТОКОЛА IP

9.1. Сетевая маршрутизация

Большинство сетей, работающих в государственном или частном секторе в нашей стране, давно превратились в серьезный инструмент. За сравнительно быстрое время они выросли из небольших локальных сетей в большие распределенные сети с огромным числом пользователей. В таких сетях ключевой характеристикой является надежность, так как задачи, решаемые с их помощью, зачастую напрямую влияют на успешность работы всей организации.

Существует несколько методов повышения надежности сети.

Например, разделение всей сети на подсети и введение маршрутизации способно обеспечить требуемую надежность и достаточно простую схему поиска нужного абонента.

Другой метод использует виртуальные локальные сети, которые позволяют повысить надежность за счет объединения пользователей в широковещательные домены. Для расширения таких доменов (сегментов или подсетей) с включением в них разнообразных сетевых устройств обычно применяются коммутаторы и протоколы остовых деревьев (spanning tree). И хотя такие схемы по заявлениям производителей позволяют практически неограниченно расширять подсети, при этом получается сеть, которой намного труднее управлять и в которой сложнее устранять неисправности.

В реальной жизни именно маршрутизаторы являются основными узлами сложных сетей, например Internet. В подавляющем большинстве случаев именно они, а не широко рекламируемые коммутаторыATM, Ethernet или FDDI, являются базовыми устройствами распределенных корпоративных сетей. Это особенно справедливо в российских условиях. В данной связи разговор о маршрутизации чрезвычайно актуален.

Как известно, протокол IP работает на сетевом уровне. Именно на этом уровне реализуется межсетевое взаимодействие, в частности, маршрутизация дейтаграмм в Internet. Но главное, что именно на сетевом уровне принимается решение о маршрутизации.

Маршрутизатор соединяет несколько различных физических сетей. Для каждого поступающего пакета маршрутизатор принимает решение о том, куда его переслать. Пакет «летает» от маршрутизатора к маршрутизатору, пока не достигнет сети, в которой находится станция-адресат. В роли маршрутизатора может выступать специальный компьютер. В большинстве реализаций стека TCP/IP рабочая станция с несколькими сетевыми интерфейсами также может выполнять функции маршрутизатора, однако для этого она должна быть специальным образом сконфигурирована.

Программное обеспечение протокола IP выполняет функции маршрутизации, выбирая путь для передачи информации в сложной схеме физических сетей. На маршрутизаторах и на конечных станциях для определения маршрута поддерживаются специальные *таблицы маршрутизации.* Выбор маршрута осуществляется на основе адреса сети назначения, который определяется по адресу получателя. Протокол IP определяет маршрут отдельно для каждой дейтаграмммы, не гарантируя надежной доставки в нужном порядке. Он непосредственно отображает данные на нижележащий физический уровень. Тем самым достигается высокая эффективность доставки дейтаграмм.

Распределенную сеть можно рассматривать как набор сетевых устройств и сетей, связанных между собой маршрутизаторами. Стек протоколов TCP/IP разработан для взаимодействия удаленных систем в сложных, распределенных сетях. Отдельные сети с коммутацией пакетов связываются маршрутизаторами.

Два устройства, подключенные к одной сети, могут посылать пакеты друг другу. Кроме того, сеть получает пакеты из удаленной сети и доставляет их определенному получателю в локальной сети или передает дальше, другим сетям. Если два устройства, расположенные в разных сетях, хотят переслать друг другу информацию, отправитель посылает пакеты определенному маршрутизатору. Тот передает пакет через систему маршрутизаторов и сетей до тех пор, пока пакет не достигнет маршрутизатора, который подключен напрямую к сети получателя. Этот конечный маршрутизатор затем передаст пакет получателю по известному физическому адресу. Маршрутизаторы передают пакеты, основываясь на номере (адресе)

сети получателя, а не на его физическом адресе. Поэтому информация, необходимая маршрутизатору, зависит от числа сетей, составляющих общую распределенную сеть, но не от числа устройств.

Выделяют два типа маршрутизации: прямую и косвенную. При прямой маршрутизации отправитель в определенной IP-сети может напрямую передавать кадры любому получателю в той же сети. При этом не требуется функциональность IP-маршрутизации.

Для передачи дейтаграммы с использованием прямой маршрутизации, отправитель инкапсулирует эту дейтаграмму в кадр канального уровня, определяет с помощью протокола ARP физический адрес получателя по известному IP-адресу и, используя сетевое аппаратное обеспечение, доставляет дейтаграмму.

Косвенная маршрутизация происходит в том случае, если отправитель и получатель находятся в разных IP-сетях. Косвенная маршрутизация требует, чтобы отправитель передавал дейтаграммы маршрутизатору для доставки их через распределенную сеть. Косвенная маршрутизация — это процесс более сложный, чем прямая маршрутизация, ввиду следующих двух причин:

- Отправитель должен определить маршрутизатор, которому необходимо адресовать дейтаграммы для доставки;
- Маршрутизатор должен уметь доставлять дейтаграммы к целевой сети, в которой располагается получатель.

Поясним на простом примере отличия прямой и косвенной маршрутизации. Предположим, что какой-либо маршрутизатор связывает две сети и, следовательно, он имеет два IP-адреса и два физических адреса для каждого из своих портов, присоединенных к этим сетям. Когда отправитель в любой из сетей направляет свой пакет маршрутизатору, то это будет прямой маршрутизацией. Если маршрутизатор отправляет пакет получателю в любой из сетей — это также прямая маршрутизация. Однако, если рассматривать взаимную работу отправителя и получателя через маршрутизатор, то их взаимодействие осуществляется с помощью косвенной маршрутизации. Маршрутизация выполняется маршрутизатором на уровне протокола IP. Этот процесс полностью прозрачен для протоколов TCP, UDP и сетевых приложений.

Перед отправкой пакета отправитель проверяет сетевой префикс IP-адреса получателя, сравнивая его с префиксом своей сети. Совпадение означает, что дейтаграмма может быть послана напрямую. Если номера сетей не совпадают, отправитель должен послать дейтаграмму маршрутизатору.

Обычно параметр «маршрутизатор по умолчанию» (default router) настраивается на каждой рабочей станции сетевым администратором. Маршрутизатор по умолчанию отвечает за доставку дейтаграмм всем устройствам, которые не подключены к сети отправителя.

Маршрутизатор принимает решение о передаче каждой дейтаграммы на основании своей таблицы маршрутизации. В качестве индекса таблицы используется номер сети, полученный из поля «Адрес получателя» в заголовке IP-дейтаграммы. Если получатель располагается в сети, подключенной к одному из портов маршрутизатора, последний может доставить дейтаграмму напрямую, не посылая ее другим маршрутизаторам. В противном случае маршрутизатор должен отослать дейтаграмму другому маршрутизатору, который находится ближе к получателю.

Существует два подхода к выбору маршрута:

- одношаговый подход;
- маршрутизация от источника.

Согласно методу одношаговой маршрутизации каждый маршрутизатор и конечный узел принимает участие в выборе только одного шага передачи дейтаграммы. В каждой строке таблицы маршрутизации указывается не весь маршрут (в виде последовательности IP-адресов маршрутизаторов, через которые должна пройти дейтаграмма), а только один IP-адрес следующего маршрутизатора (маршрутизатора на том пути, по которому нужно передать дейтаграмму). Вместе с дейтаграммой этому маршрутизатору передается и ответственность за выбор следующего шага. Такой подход распределяет задачу выбора

маршрута и снимает ограничение на максимальное количество маршрутизаторов в пути. Кроме того, за счет использования маршрутизатора по умолчанию (который обычно занимает в таблице маршрутизации последнюю строку) существенно сокращается объем таблицы. Все дейтаграммы, номера сетей которых отсутствуют в таблице маршрутизации, передаются маршрутизатору по умолчанию. Подразумевается, что маршрутизатор по умолчанию передает дейтаграмму в магистральную сеть, а маршрутизаторы, подключенные к магистральной сети, имеют полную информации о ее топологии.

Существуют различные алгоритмы построения таблиц для одношаговой маршрутизации. Их делят на три класса:

- **Алгоритмы фиксированной маршрутизации.** Они применяются в сетях с простой топологией и основаны на составлении таблиц маршрутизации «вручную» администратором сети.
- **Алгоритмы простой маршрутизации.** Они разделяются на три подкласса:
  - случайная маршрутизация (дейтаграммы передаются в любом случайном направлении, кроме исходного);
  - лавинная маршрутизация (дейтаграммы передаются во всех направлениях, кроме исходного);
  - адаптивная маршрутизация (таблица маршрутизации составляется на основании данных, содержащихся в проходящих через маршрутизатор дейтаграммах).
- **Алгоритмы адаптивной маршрутизации.** Основные алгоритмы, применяемые в современных сетях. Маршрутизаторы периодически обмениваются между собой информацией о сетевой топологии.

При маршрутизации от источника выбор маршрута производится конечным узлом или первым маршрутизатором на пути следования дейтаграммы. Все остальные маршрутизаторы только отрабатывают выбранный маршрут. Этот метод в сетях IP применяется только в целях отладки.

Управление таблицей маршрутизации на маршрутизаторах в большой распределенной сети является сложной задачей. Таблицы маршрутизации для отображения текущей сетевой топологии должны быть динамическими. Маршрутизатор обменивается с другими маршрутизаторами информацией о маршрутах. К протоколам маршрутизации, обменивающимися информацией о маршрутах в сетях IP, относятся: Routing Information Protocol (RIP), Open Shortest Path First Protocol (OSPF), Integrated Intermedia te System to Intermediate System (ISIS), Exterior Gateway Protocol (EGP) и Border Gateway Protocol(BGP).

Сегодня Internet значительно отличается от той сети, которая была создана в 1980 году. Internet стала мировой, общедоступной информационной сетью, которая удваивается в размерах каждые девять месяцев. Однако столь бурный рост Internet неизбежно привел к выявлению большого количества проблем, связанных с маршрутизацией. При этом:

- Производительность протоколов маршрутизации значительно снизилась;
- Размер сообщений, которыми обмениваются маршрутизаторы для поддержания своих таблиц маршрутизации растет, что требует все больших ресурсов маршрутизаторов и все большей пропускной способности сети;
- Большое число маршрутизаторов, работающих с протоколами маршрутизации, делают поддержку механизмов обнаружения и изоляции сбоев практически невозможной.

Для того чтобы в какой-то степени снизить влияние этих факторов, сеть Internet разделили на отдельные *автономные системы* (Autonomous System). Автономная система (АС) представляет собою группу сетей и маршрутизаторов, управляемую уполномоченным. АС позволяет независимо управлять различными частями Internet и разрешает маршрутизаторам внутри различных АС использовать разные протоколы маршрутизации. Каждая сеть внутри АС должна быть доступна из Internet.

Маршрутизаторы внутри одной АС взаимодействуют, используя динамические протоколы маршрутизации. Протоколы, управляющие маршрутной информацией внутри АС, относятся к классу так называемых протоколов IGP (Interior Gateway Protocol, внутренний шлюзовой протокол). Главной задачей

протоколов класса IGP является поддержание необходимой производительности. Эти протоколы должны немедленно подстраиваться под изменения в сетевой топологии и находить маршрут с наименьшей стоимостью. Отдельные протоколы класса IGP это: RIP, NLSP, OSPF, IGRP, EIGRP и IS-IS.

Взаимодействие между маршрутизаторами, принадлежащими к различным АС, требует дополнительного протокола, который называется EGP (Exterior Gateway Protocol, внешний шлюзовой протокол). Так как различные АС управляются совершенно независимо, в протоколе предприняты меры, предотвращающие влияние сбоев в других АС. Необходимо учитывать, что для связи с маршрутизаторами внутри АС корневой маршрутизатор (см. ниже), кроме поддержки EGP, должен также поддерживать протоколы класса IGP, так как всю информацию о своей АС протокол EGP получает через IGP.

9.2 Протокол RIP

Протокол RIP (Routing Information Protocol) описан в документе RFC 1058. Протокол RIP относится к классу протоколов IGP. Этот протокол является одним из первых протоколов обмена маршрутной информацией между маршрутизаторами в IP-сети. Существует также версия этого протокола для сетей IPX/SPX компании Novell. Впервые протокол RIP появился в 1982 году как часть стека протокола TCP/IP для UNIX, разработанной Berkley Software Distribution. Исторически протокол RIP близко связан с семейством сетевых протоколов фирмы Xerox. Преимуществом протокола RIP является его простота. Недостатком — увеличение трафика за счет периодической рассылки широковещательных сообщений.

Протокол RIP стал стандартным протоколом маршрутизации внутри отдельной автономной системы (АС), хотя он существенно ограничивает размер автономной системы. Это связано с тем, что протокол RIP не поддерживает длинные пути, которые содержат более 15 переходов.

Протокол RIP использует алгоритм длины вектора (Distance vector algorithm). Этот алгоритм основывается на тех же принципах, что и алгоритм Беллмана-Форда, который был применен в первом протоколе маршрутизации для сетей ARPANET и исходит из предположения, что каждый маршрутизатор может вычислить самый короткий маршрут и соответствующее расстояние до каждой сети.

То есть, каждый маршрутизатор выбирает ближайший соседний маршрутизатор, который расположен на этом самом коротком маршруте до получателя. Выбор осуществляется на основании информации о стоимости путей (выбирается путь с меньшей стоимостью).

Стоимость вычисляется по информации, имеющейся в таблицах маршрутизации всех соседних маршрутизаторов (маршрутизаторы регулярно обмениваются между собой таблицами маршрутизации). Этот алгоритм хорошо работает в небольших сетях. В больших сетях он заполняет сеть широковещательным трафиком.

Протокол не всегда точно и быстро учитывает изменения сетевой топологии, так как маршрутизаторы не имеют точного представления о топологии сети, а располагают только информацией, полученной от своих соседей.

Протокол RIP использует в качестве метрики маршрута количество переходов, то есть число маршрутизаторов, которые должна миновать дейтаграмма, прежде чем она достигнет получателя.

Маршрутизаторы с поддержкой протокола RIP всегда выбирают маршрут с наименьшим числом переходов.

Таблица маршрутизации RIP содержит следующие поля:

- IP-адрес целевой сети;
- Количество переходов до целевой сети;
- Адрес первого маршрутизатора на пути к целевой сети;
- Идентификатор соседнего маршрутизатора, который является источником данной адресной информации в таблице маршрутизации;

- Таймер для отслеживания времени, прошедшего с момента последнего обновления записи.

При включении маршрутизатора таблица маршрутизации заполняется описанием сетей, которые напрямую подключены к данному маршрутизатору.

Затем соседние маршрутизаторы шлют ему свою информацию. Периодически каждый маршрутизатор посылает сообщения об обновлении маршрута всем своим соседям. Эти сообщения содержат информацию из таблицы маршрутизации.

Если объем информации слишком велик и не помещается в одно сообщение протокола RIP, она будет разделена на части и помещена в несколько сообщений. Перед передачей сообщений об обновлении маршрута соседним маршрутизаторам маршрутизатор увеличивает количество переходов до получателя на единицу.

Когда сообщения об обновлении маршрута приходят на маршрутизатор, он обновляет свою таблицу маршрутизации в соответствии со следующими *правилами*:

- Если новое количество переходов меньше, чем текущее (для конкретной записи), маршрутизатор примет новый маршрут. Эта новая запись будет храниться в таблице до тех пор, пока не появится маршрут с еще меньшей метрикой;
- Если передающий маршрутизатор является источником информации для существующей записи, то принявший сообщение маршрутизатор будет использовать новое значение количества переходов, даже если оно больше, чем старое.

Реакция протокола RIP на изменения в топологии сети зависит от того, как маршрутизатор информирует своих соседей о модификации его таблицы маршрутизации.

Однако следует учитывать, что если сетевая топология изменяется, то соседи могут перестать быть соседями. Кроме того, если маршрутизатор выходит из строя, он не может известить своих соседей об изменениях в топологии.

Таким образом, в случае отсутствия сообщений об обновлении маршрутизации предполагаемый маршрут может не отражать произошедшие изменения.

Все маршрутизаторы, участвующие в обмене сообщениями об обновлении маршрута по протоколу RIP, посылают эти сообщения через определенный интервал, который по умолчанию составляет 30 с.

Если маршрутизатор не получает сообщения от маршрутизатора, который отвечает за определенную запись в таблице маршрутизации за временной интервал, равный увеличенному в шесть раз интервалу обмена (по умолчанию 180 с), он предполагает, что либо его соседний маршрутизатор вышел из строя, либо между ними нарушилась связь.

Затем маршрутизатор помечает отказавший маршрут как некорректный и, в конечном счете, удаляет его из своей таблицы маршрутизации.

Когда маршрутизатор получит информацию о новом маршруте от другого своего соседа, он будет использоваться вместо старого удаленного маршрута. Шестикратное увеличение интервала необходимо для того, чтобы избежать исключения маршрута при случайной потере одного сообщения об обновлении.

Кроме того, маршрутизатору чрезвычайно важно оповестить своих соседей о том, что не существует корректного маршрута к определенному получателю. Протокол RIP позволяет выполнить это с помощью стандартных сообщений об обновлении маршрутизации.

Для обозначения недостижимости получателя количество переходов устанавливается равным 16. Это значение можно считать «бесконечностью», так как допустимые маршруты не могут иметь более 15 переходов. Если какая-либо сеть становится недостижимой, все соседние маршрутизаторы установят метрику для этой сети равной 16.

В следующем цикле посылки сообщений об обновлении маршрутизаторы передадут эти данные всем своим соседям, указав для них число переходов к недостижимой сети равным 16. На рис. 9.1 показан пример сетевой топологии с вышедшим из строя каналом связи.

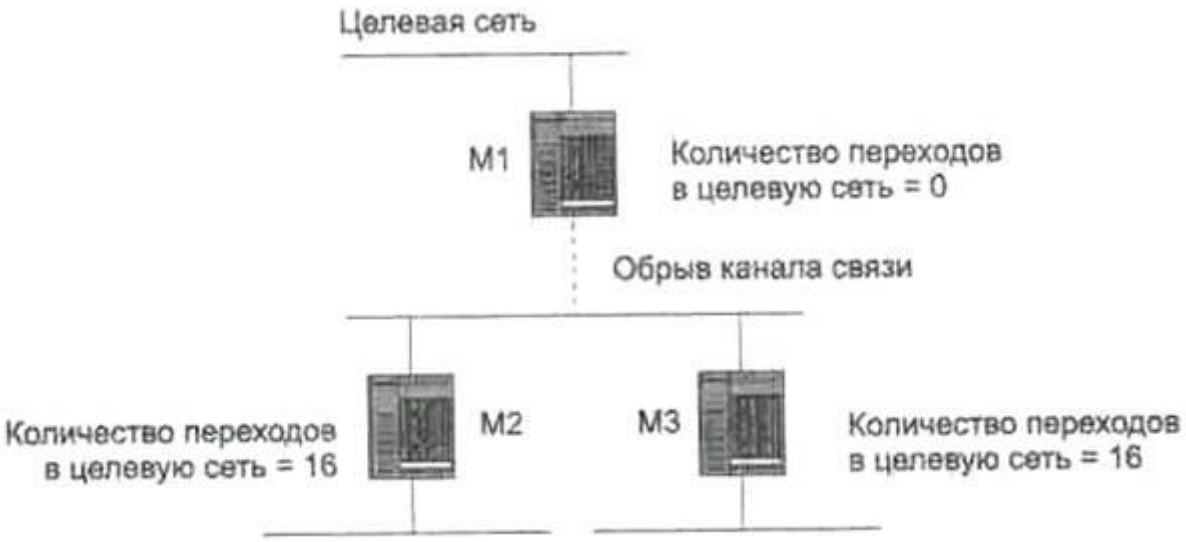


Рис 9.1 Обрыв канала связи

Протокол RIP гарантирует, что таблицы маршрутизации за определенное время (время сходимости) станут правильными. Однако алгоритм в текущем своем состоянии не гарантирует, что время сходимости будет мало.

Может оказаться так, что до истечения времени сходимости в сеть будут внесены изменения, и тогда все начнется заново. Вопрос о том, сколько времени требуется для сходимости процесса извещения об изменении маршрутов, является достаточно сложным. Когда происходит выход из строя канала связи или возникают другие проблемы в сети, некоторые из существующих маршрутов становятся недоступными или менее подходящими для передачи. Проблема в том, что, в принципе, соседние маршрутизаторы могут обмениваться между собой не вполне правильной информацией. Эта информация будет передана дальше по сети. Она будет исправлена только при следующих итерациях. Медленная сходимость может иметь достаточно серьезные последствия. В частности, может увеличиться объем трафика сообщений об изменениях маршрутов, которыми обмениваются маршрутизаторы, могут образовываться логические петли маршрутизации и т. д.

На рис. 9.2 показана логическая петля, образованная маршрутизаторами M2 и M3.

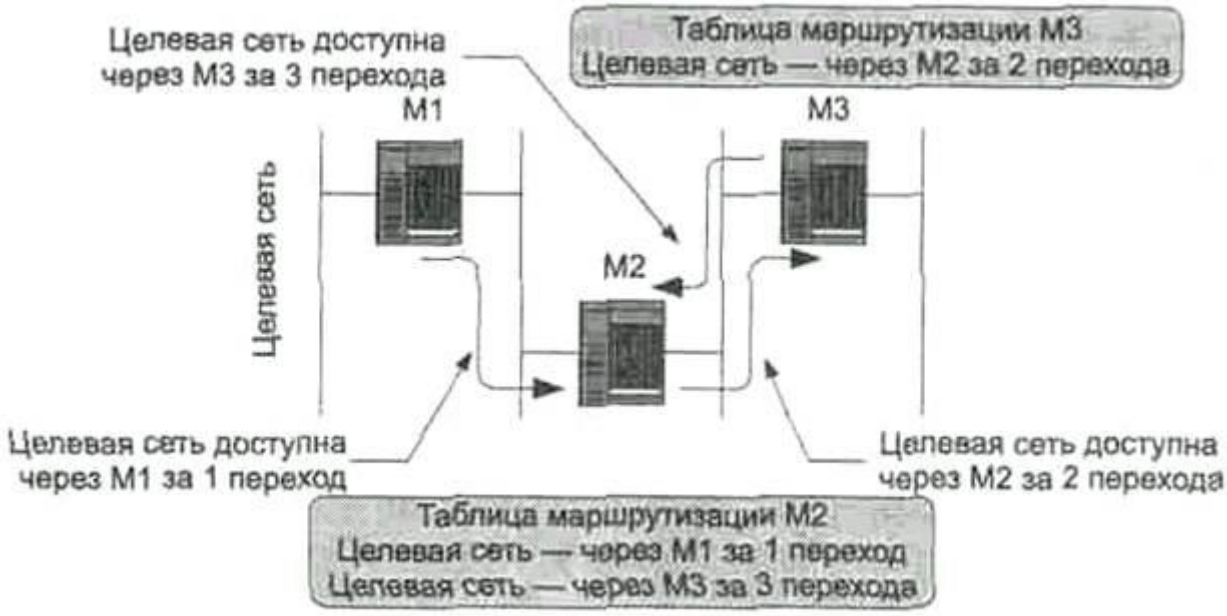


Рис. 9.2. Пример образования логической петли маршрутизации

Маршрутизатор M2 может достигнуть целевой сети через маршрутизатор Ml за 1 переход.

Маршрутизатор М3 получает эту информацию из периодических сообщений об обновлении от маршрутизатора М2. Поэтому М3 может достигнуть целевую сеть через маршрутизатор М2 за 2 перехода.

В следующем цикле посылки сообщений об обновлении маршрутизатор М3 известит маршрутизатор М2 о достижимости целевой сети за 3 перехода. В результате маршрутизатор М2 будет иметь два маршрута в целевую сеть: первый маршрут с использованием маршрутизатора Ml и количеством переходов 1, а второй маршрут с использованием маршрутизатора М3 и количеством переходов 3.

Маршрутизатор М2 выберет маршрут через маршрутизатор Ml, так как он имеет наименьшую метрику.

В случае, если канал связи между маршрутизаторами Ml и М2 выйдет из строя, М2 не получит сообщение об обновлении в требуемый интервал времени и удалит из своей таблицы запись о маршруте в целевую сеть через Ml. В результате у маршрутизатора М2 будет сформирована запись о маршруте в целевую сеть через маршрутизатор М3 за 3 перехода. Следовательно, М2 будет пересылать весь трафик маршрутизатору М3, тот его пошлет обратно М2 и т. д. Таким образом образуется логическая петля маршрутизации. Обмен пакетами между маршрутизаторами будет продолжаться до тех пор, пока полеTTL в заголовке IP-дейтаграммы не станет равно 0. Тогда дейтаграмма будет удалена одним из маршрутизаторов.

Существует несколько технологий, которые ускоряют сходимость протокола RIP IP и повышают его производительность. К ним относятся:

- расщепление горизонта (Split-Horizon),
- обратное исправление (Poison Reverse),
- мгновенное изменение (Triggered Update)
- временный отказ от приема информации Hold-Down (приостановка)
- Garbage-Collection (сборка мусора).

Описанная проблема «обоюдного обмана» может быть решена путем определения направления посылки маршрутной информации. Согласно технологии Split-Horizon маршрутизатор не будет распространять информацию об определенном маршруте через порт, который явился источником данной информации. Другими словами, маршрутизатор не будет информировать о достижимости получателя своего соседа, от которого была получена информация о маршруте к получателю. На рис. 9.3 показано, каким образом Split-Horizon устранит образовавшуюся петлю.

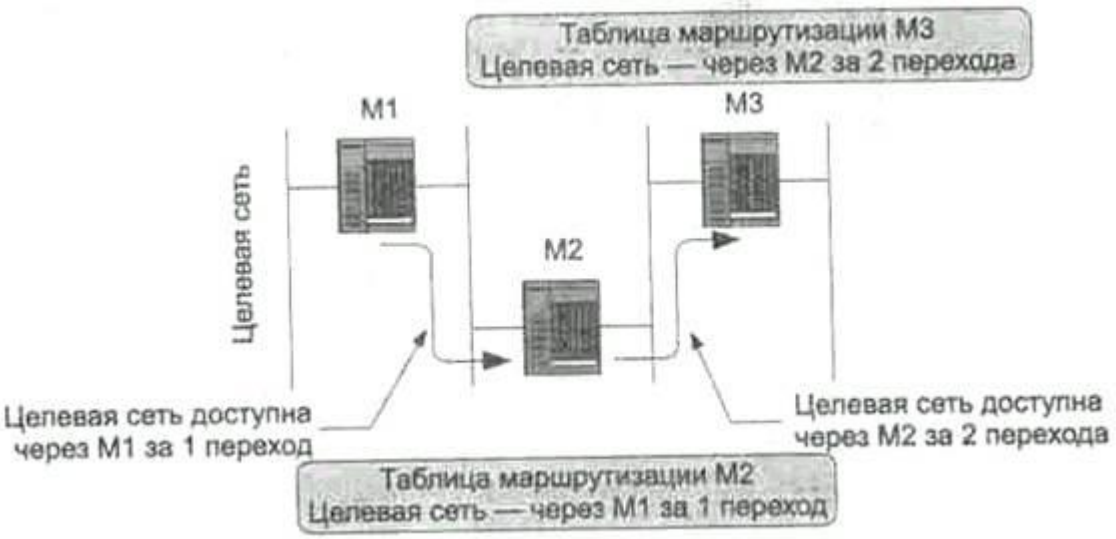


Рис. 9.3 Результат работы технологии Split-Horizon

Процедура обмена сообщениями об обновлении маршрута остается такой же, как и выше, за исключением того, что маршрутизатор М3 не будет посылать информацию о маршруте к целевой сети маршрутизатору М2. В результате маршрутизатор М2 имеет только один маршрут в целевую сеть, и, если

произойдет обрыв канала связи между маршрутизаторами Ml и М2, он удалит из своей таблицы маршрутизации маршрут в целевую сеть, и петля не возникнет.

Технология Poison Reverse решает те же задачи, что и Split-Horizon, однако немного другим способом. Маршрутизаторы распространяют информацию о маршрутах через порты, которые явились источниками информации. Но эти маршруты указываются как недостижимые — количество переходов устанавливается равным 16 (рис. 9.4).

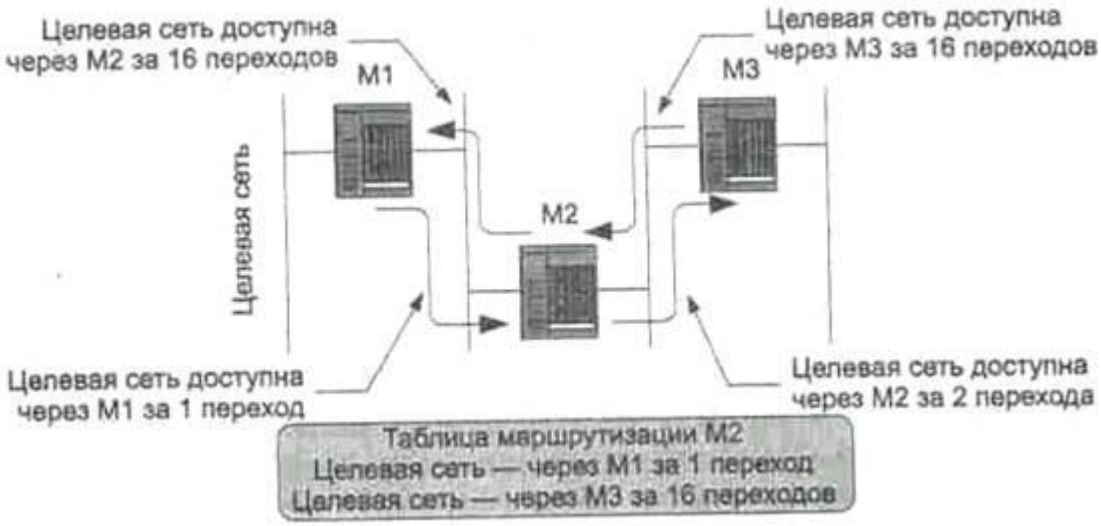


Рис. 9.4 Результат работы технологии Poison Reverse

По сути, сообщения о маршрутах с числом переходов, равным 16, — это то же самое, что отсутствие сообщений (не считая увеличения нагрузки на сеть).

Однако при изменении сетевой топологии скорость сходимости в этой схеме может увеличиться, так как упоминаются и достижимые, и недостижимые маршруты.

Основным недостатком этой технологии является то, что она увеличивает нагрузку на сеть. Во многих случаях администратор может согласиться с медленной сходимостью ради уменьшения загрузки сети, вызываемой потоком сообщений об обновлении.

Технологии Split-Horizon и Poison Reverse хорошо работают в случае двух маршрутизаторов в петле. Однако возможны ситуации, когда три или более маршрутизаторов «обманывают друг друга». Например, маршрутизатор Ml полагает, что имеет маршрут к какой-либо сети через маршрутизатор М2, маршрутизатор М2 — через М3, М3 — через М4, а М4 — снова через Ml. Для ускорения сходимости в подобных ситуациях служит технология Triggered Update, которая требует, чтобы маршрутизатор немедленно посылал сообщения об обновлении своим соседям, если он обнаружил изменение в метрике маршрута. Сообщение должно быть послано, даже если не пришло время для регулярных сообщений. Естественно, это ускорит сходимость, но увеличит трафик в сети. Поэтому технология Triggered Update может вызвать чрезмерную загрузку сети с ограниченной пропускной способностью. Все реализации протокола RIP должны предусматривать предел частоты немедленной посылки сообщений об обновлении (скажем, не чаще чем раз в секунду), чтобы не загружать сеть. Простым решением данной проблемы является установка таймера на случайное число между одной и пятью секундами. После обнуления таймера происходит посылка сообщения об обновлении. Если произошли другие изменения в сети, при которых немедленно высылаются дополнительные сообщения, маршрутизатор должен выждать обнуление таймера и только затем посылать новое сообщение. Таймер после этого устанавливается в другое случайное число в заданном интервале.

Например, маршрутизатор М будет устанавливать тайм-аут для своего маршрута в целевую сеть. Это заставит его, после обнуления таймера, сформировать сообщения об обновлении и посылать их через свои порты. Сообщения будут распространяться через все пути, обновляя метрику для целевой сети до

бесконечности. Такой каскад сообщений об обновлении останавливается только тогда, когда он достигает маршрутизатора, который использует путь до целевой сети, не проходящий через маршрутизатор М.

Маршруты в таблице маршрутизации, создаваемой с помощью протокола RIP, могут иметь различные состояния. Например, для маршрутизаторов фирмы 3Com маршруты могут находиться в одном из следующих состояний:

- **UP** — маршрут достижим с определенной метрикой, меньшей 16. Маршрут остается в данном состоянии в течение шестикратного интервала времени между посылками сообщений об обновлении — это время называется таймером маршрута. Данный таймер сбрасывается каждый раз, когда поступает новое сообщение об обновлении этого маршрута. По обнулении таймера маршрут считается недействительным и переводится в состояние Garbage- Collection;
- **Hold-Down** — маршрут имеет метрику, равную бесконечности (больше или равно 16). Маршрут будет оставаться в данной стадии в течение времени, равного четырехкратному интервалу между посылками сообщений об обновлении. В этом состоянии с маршрутом связывают таймер Hold-Down. Когда таймер Hold-Down обнулится, маршрутизатор перейдет в состояние Garbage-Collection. Если до обнуления таймера для этого маршрута будет получено сообщение с метрикой меньше 16, маршрут перейдет в стадию UP. Цель состояния Hold-Down — оповестить все маршрутизаторы в автономной системе о том, что маршрут не функционирует. Таймер препятствует маршрутизатору в этом состоянии принимать сообщения, содержимое которых устарело;
- **Garbage-Collection** — маршрут с обнуленным таймером, который до этого был в стадии UP. Маршрут может оставаться в этом состоянии на протяжении четырехкратного интервала времени между посылками об обновлении. Это время отсчитывает таймер Garbage-Collection. Если до обнуления таймера Garbage-Collection для этого маршрута не будет получено сообщений об обновлении, маршрут удаляется из таблицы маршрутизации. Если до обнуления таймера соседний маршрутизатор информирует об изменении этого маршрута с метрикой меньше 16, маршрут будет изменен, а его удаление отменено. Таймер будет сброшен, а маршрут перейдет в состояние UP.

На рис. 9.5 показана смена состояний маршрута в таблице маршрутизации.

Рис. 9.5 Алгоритм смены состояний маршрута

Сетевому администратору в автономной системе требуется контроль за использованием ресурсов сети. В частности, желательно ограничить поток сообщений о маршрутизации. Существует набор средств,

которые позволяют администратору контролировать содержимое сообщений об обновлении маршрута. Перечислим их на примере маршрутизатора NetBuilder II фирмы 3Com. В частности:

- Для уменьшения размера сообщений об обновлении маршрутизатор может не включать информацию о локальных сетях, к которым подключен он или его соседи;
- Если маршрутизатор поддерживает технологию Split-Horizon, он не будет оповещать о маршрутах, через которые была получена соответствующая информация;
- Устанавливая параметр Poison/No Poison, маршруты можно указывать как недостижимые или просто не включать их в сообщения;
- Задав сетевые правила (Network Policy), можно четко очертить список сетей, о которых оповещает протокол RIP. По умолчанию протокол должен сообщать обо всех сетях;
- Внутренние правила (Interior Policy) могут указывать на появление сообщений протокола RIP, информирующих о маршрутах, за которые отвечали другие протоколы класса IGP (например, OSPF илиIS-IS);
- Внешние правила (Exterior Policy) служат для управления списком сетей, оповещаемых протоколом RIP, который получен от протоколов политики маршрутизации, таких как EGP или BGP. По умолчанию информация о таких маршрутах не должна рассылаться;
- Настраиваемые статические правила (Static Policy) служат для определения того, будут ли рассылаемые сообщения об обновлении включать информацию о статических маршрутах. Следует отметить, что все маршруты, определяемые сообщениями протокола RIP IP, будут рассматриваться как «собственность» этого протокола;
- Настраиваемые метрики по умолчанию (Default Metric) помогают отслеживать сообщения RIP о маршруте по умолчанию. Сетевой администратор может выбрать метрику, которая будет использоваться с маршрутом по умолчанию;
- Правила получения (Receive Policy) служат для фильтрации сообщений от соседних маршрутизаторов. Это позволяет администратору принимать не всю информацию, получаемую от соседей и предназначенную для обновления таблицы маршрутизации.

Протокол RIP для передачи сообщений использует дейтаграммы UDP и протокольный порт 520. Полное сообщение протокола RIP, включая заголовок и данные, инкапсулируется в поле данных дейтаграммы UDP, которая, в свою очередь, инкапсулируется в IP-дейтаграмму (рис. 9.6).

Рис 9.6 Инкапсуляция сообщений протокола RIP

Все сообщения протокола RIP состоят из заголовка фиксированной длины и следующей за ним таблицы маршрутизации (точнее, ее подмножества) передающего маршрутизатора. В таблице представлен список достижимых сетей (рис. 9.7). Цифра 0 на рис. 9.7 означает, что это поле, согласно спецификации, должно быть нулевым.

Команда (8 бит)	Версия (8 бит)	0(16 бит)
Идентификатор адресной схемы (16 бит)		0(16 бит)
IP-адрес (32 бита)		

0(32 бита)	
0(32 бита)	
Количество переходов (32 бита)	
Идентификатор адресной схемы (16 бит)	0(16 бит)
IP-адрес (32 бита)	
0(32 бита)	
0(32 бита)	
Количество переходов (32 бита)	
...	
Идентификатор адресной схемы (16 бит)	0(16 бит)
IP-адрес (32 бита)	
0(32 бита)	
0(32 бита)	
Количество переходов (32 бита)	

Рис. 9.7 . Формат сообщения протокола RIP

Часть сообщения (запись из таблицы маршрутизации), начинающаяся с поля «Идентификатор адресной схемы» и заканчивающаяся полем «Количество переходов», может повторяться в сообщении до 25 раз. Это позволяет каждому сообщению протокола RIP переносить до 25 адресов, которые занимают до 500 байтов. Это ограничение вызвано тем, что максимальный размер сообщений составляет 521 байт, не включая заголовков протоколов IP и UDP. Маршрутизатору может потребоваться передать несколько сообщений для рассылки всей таблицы маршрутизации своим соседям. Не существует специальных требований к последовательности посылки сообщений, так как все маршруты обрабатываются независимо.

Поля в сообщении протокола RIP имеют следующие значения:

- **Поле «Команда»** указывает назначение дейтаграммы. Команда может быть либо запросом протокола RIP (1), либо ответом (2). Существуют и другие команды, однако в настоящее время они не используются;
- **Поле «Версия»** — версия протокола RIP;

- **Поле «Идентификатор адресной схемы»** указывает тип адреса для каждой записи. Сообщения RIP могут переносить адреса любого формата различных протоколов. Идентификатор IP-адреса равен 2;
- **Поле «IP-адрес»** занимает четыре байта. В нем могут указываться адрес устройства, номер сети, номер подсети или ноль, что означает маршрут по умолчанию;
- **Поле «Количество переходов»** указывает текущую метрику для данного сетевого адреса.

Поля, которые должны быть заполнены нулями, являются частью адресного поля, но не используются, так как адреса IP занимают только 4 байта. Протокол RIP может работать с сетевыми адресами длиной до 12 байт.

Выше рассматривался протокол маршрутизации RIP версии 1 (RIP-1 IP). Однако существует версия 2 этого популярного протокола (RIP-2 IP), описанная в документе RFC 1388. Версия 2 поддерживаетCIDR, аутентификацию, подсети и групповую передачу. На рис. 9.8 показан формат сообщений протокола RIP-2.

Команда (8 бит)	Версия (8 бит)	Домен маршрутизации (16 бит)
Идентификатор адресной схемы (16 бит)		Метка маршрута (16 бит)
IP-адрес (32 бита)		
Маска подсети (32 бита)		
Следующий переход (32 бита)		
Метрика (32 бита)		

Рис. 9.8 . Формат сообщения протокола RIP-2 IP

Сообщения версии 1 содержат нулевые неиспользуемые поля. Это позволяет задействовать их для расширений, предлагаемых в версии 2. Протокол RIP-2 IP наследует все поля первой версии и добавляет следующие:

- **Поле «Домен маршрутизации»** используется вместе с полем «Следующий переход» для совместной работы нескольких АС в едином домене маршрутизации;
- **Поле «Маска подсети»** — двоичное число, содержащее единицы в тех разрядах, которые относятся к расширенному сетевому префиксу. Маска подсети делит IP-адрес на номер подсети и номер устройства в этой подсети и позволяет выполнять маршрутизацию в сформированной структуре подсетей;
- **Поле «Метка маршрута»** служит для идентификации внешних маршрутов и задействуется в протоколах политики маршрутизации (EGP или BGP).

Номер версии для протокола RIP-2 IP равен 2. При получении такого сообщения маршрутизатором, поддерживающим протокол RIP-1 IP, он просто проигнорирует любые поля, которые согласно версии 1 протокола должны содержать нули. Поэтому он будет корректно обрабатывать все записи, не использующие расширений RIP-2 IP (например, простое поле с IP-адресом получателя без указания маски подсети).

Протокол RIP-1 IP не поддерживает безопасность. Любое устройство (рабочая станция или сервер), посылающее сообщение UDP через порт 520, будет рассматриваться маршрутизатором как его сосед. Отсутствие аутентификации возлагает на администратора дополнительные задачи по управлению правилами. Например, администратору может понадобиться вручную настроить список авторизованных соседей.

Протокол RIP-2 IP поддерживает аутентификацию. Стандарт допускает замену первой записи в сообщении на сегмент аутентификации. Таким образом, сообщение может содержать сегмент аутентификации и 24 записи из таблицы маршрутизации в стандартной форме RIP-2 IP. На рис. 9.9 показан формат сегмента аутентификации протокола RIP-2 IP.

%FFFFF	Тип аудентификации (16 бит)
Аудентификация (16 бит)	

Рис. 9.9 Формат сегмента ауденитфикации RIP -2 IP

Сегмент аутентификации определяется тем, что поле «Идентификатор адресной схемы» равно %FFFFF. Затем указывается тип схемы аутентификации, и следующие 16 байт отводятся под данные аутентификации. После получения сообщения маршрутизаторы проверят сегмент аутентификации (если протокол RIP-2 IP работает в безопасном режиме) и будут игнорировать любые сообщения с некорректной аутентификацией.

Так как протокол RIP был разработан достаточно давно и практически не изменялся за это время, он обладает существенными недостатками, которые ограничивают его применение в сложных сетях:

- RIP допускает 15 переходов между отправителем и получателем. Если количество переходов превышает 15, получатель рассматривается как недостижимый, что очень сильно ограничивает размеры автономной системы.
- Использование в качестве метрики маршрута количества переходов приводит к тому, что протокол RIP не всегда выбирает самый эффективный и экономный маршрут. Например, может оказаться так, что выбранный маршрут содержит медленный и дорогой канал связи. Протоколы маршрутизации, которые используют в качестве метрики количество переходов, не принимают во внимание такие важные характеристики, как скорость канала, его надежность и т. д.
- Механизм обмена сообщениями RIP основан на широковещательной рассылке всей таблицы маршрутизации. В средних и больших сетях это может сильно загрузить каналы связи, особенно если распределенная сеть состоит из удаленных сетей, соединяемых по коммутируемым каналам связи.
- Так как протокол RIP работает по алгоритму вектора расстояния, он обладает медленной сходимостью. В случае, если канал связи выйдет из строя, потребуется несколько минут для того, чтобы маршрутизаторы скорректировали свои таблицы. В течение этого времени могут образоваться петли маршрутизации.

10. ПРОТОКОЛ OSPF

Спецификация протокола OSPF (Open Shortest Path First) описана в документе RFC 1247. Протокол принят в 1991 году. Он ориентирован на применение в больших распределенных сетях. OSPF вычисляет маршруты в сетях IP, работая совместно с другими протоколами обмена маршрутной информацией. Протокол OSPF основан на алгоритме состояния канала. Суть этого алгоритма состоит в том, что он должен вычислить кратчайший путь. При этом «кратчайший» не означает, что путь физически самый короткий. Имеется в виду, что информация пройдет по этому пути быстрее, чем по другим. Маршрутизатор, работающий с этим протоколом, отправляет запросы всем соседним маршрутизаторам, находящимся в одном с ним домене маршрутизации, для выявления состояния каналов до них и далее от них. Состояние канала при этом характеризуется несколькими параметрами, которые называются *метриками.* Метрикой может быть пропускная способность канала, его загрузка на текущий момент, задержка информации при ее прохождении по этому каналу и т. д. Обобщив полученные сведения, этот маршрутизатор сообщает их всем соседям. После этого им строится ориентированный граф, который повторяет топологию домена маршрутизации. Каждому ребру этого графа назначается оценочный параметр (метрика). После построения графа используется алгоритм Дейкстры, который по двум заданным узлам находит набор ребер с наименьшей суммарной стоимостью, то есть, по сути, выбирает оптимальный маршрут. По совокупной информации (полученной и найденной в результате вычислений) создается таблица маршрутизации.

Сообщения протокола OSPF передаются в IP-дейтаграммах с полем «Протокол» равным 89.

Протокол OSPF отвечает за IP-маршрутизацию и относится к классу протоколов IGP. Он может заменить протокол маршрутизации RIP в больших и сложных сетях.

Протокол OSPF решает многие из перечисленных выше проблем, присущих протоколу RIP. Кроме того:

- Протокол OSPF обладает большей скоростью сходимости, что предотвращает возникновение петель маршрутизации;
- При работе протокола не генерируется большой сетевой трафик;
- Информация, которой обмениваются маршрутизаторы, проходит процедуру аутентификации. Это позволяет участвовать в процессе маршрутизации только авторизованным маршрутизаторам. Аутентификация устраняет возможность случайного или преднамеренного искажения маршрутной информации;
- Протокол OSPF использует групповую передачу вместо широковещательной. Такой подход позволяет исключить передачу маршрутной информации тем устройствам, которым она не нужна;
- Протокол поддерживает распределение нагрузки (load balancing) по маршрутам, имеющим одинаковые стоимости;
- Протокол поддерживает маски подсетей переменной длины (VLSM). Данные о масках подсетей передаются в сообщениях LSA (Link-State Advertisement,- объявление о состоянии канала), таким образом маршрутизаторы получают эту информацию динамически, что позволяет более гибко использовать выделенное организации адресное пространство;
- Протокол поддерживает области маршрутизации. Это позволяет администраторам разделять автономную систему на отдельные части с изоляцией сетевого трафика и маршрутизации;
- Протокол OSPF поддерживает передачу внешней маршрутной информации через автономные системы.

Протокол OSPF хорошо подходит для современных, больших, динамически изменяющихся сетей.

Например, в отличие от протокола RIP, который по умолчанию рассылает всю свою таблицу маршрутизации каждые 30 с, протокол OSPF посылает информацию о состоянии каналов каждые 30 мин.

При обнаружении изменения сетевой топологии протокол OSPF сразу же посылает небольшие (порядка 75 байт) сообщения.

В начале работы каждый маршрутизатор передает сообщения LSA на все свои порты. Эти сообщения позволяют однозначно идентифицировать передающий маршрутизатор и определить состояние всех его портов: IP-адрес, маску подсети, метрику, присвоенную каналу связи порта, и статус канала связи.

Сообщение LSA передается во всем домене маршрутизации. Поэтому маршрутизаторы обладают информацией о состоянии каналов других маршрутизаторов в домене. Каждый маршрутизатор на основе сообщений LSA формирует базу данных состояния каналов (Link-State Database). Эта база данных одна и та же на всех маршрутизаторах в одном домене. Алгоритм маршрутизации поддерживает синхронизацию этих баз данных.

После того как все базы данных синхронизированы, каждый маршрутизатор начинает создавать свою таблицу маршрутизации. Эта таблица базируется на информации, содержащейся в базе данных состояния каналов.

Сначала создается карта сетевой топологии. Непосредственно связанные маршрутизаторы по терминологии протокола OSPF называются *соседями.* Каждый маршрутизатор хранит информацию о том, в каком состоянии, по его мнению, находится сосед. Маршрутизатор полагается на соседние маршрутизаторы и передает им дейтаграммы только в том случае, если он уверен, что они полностью работоспособны.

После создания карты сетевой топологии маршрутизатор формирует дерево кратчайших путей ко всем возможным получателям.

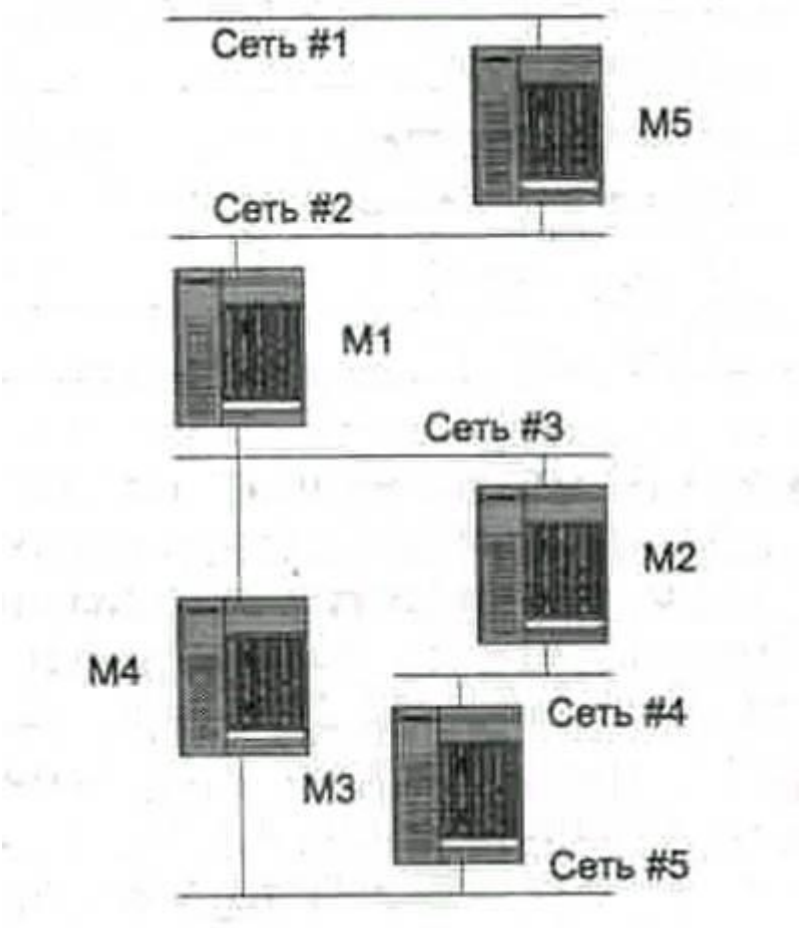


Рис. 10.1 Множество сетей

Это дерево строится таким образом, чтобы путь от корня — маршрутизатора, формирующего дерево, — до конечных объектов — целевых сетей — имел наименьшую стоимость. На рис.10.1 показано множество сетей, для которого формируется дерево кратчайших путей, изображенное на рис. 10.2 . Мы считаем, что метрики пути до каждой из сетей равны 10.

Рис. 10.2 Дерево кратчайших путей

После того как дерево построено, маршрутизаторы формируют локальные таблицы маршрутизации. Дерево путей показывает, что маршрутизатор Ml напрямую подключен к сетям #2 и #3. В таблицу маршрутизации для сетей, подключенных напрямую, в качестве значения метрики заносится 0. В табл.10.1 показаны данные таблицы маршрутизации для маршрутизатора Ml.

Таблица10.1.

Таблица маршрутизации маршрутизатора M1

Сеть	**Следующий маршрутизатор в сети**	**Метрика маршрута**
Сеть #1	Маршрутизатор M5	20
Сеть #2	Подключена напрямую	0
Сеть #3	Подключена напрямую	0
Сеть #4	Маршрутизатор M2	20
Сеть #5	Маршрутизатор M4	20

Так как маршрут в сеть #4 состоит из двух путей (от маршрутизатора Ml к M2, а затем от M2 к сети #4), а метрика каждого пути равна 10, то метрика маршрута до сети #4 равна 20. Аналогично вычисляются метрики остальных маршрутов. Следует отметить, что к сети #5 от маршрутизатора Ml имеются два пути — через маршрутизаторы M4 и M2. Путь через маршрутизатор M4 имеет метрику 20, а путь через маршрутизатор M2 — 30. Алгоритм состояния канала будет всегда выбирать кратчайший маршрут в целевую сеть. В результате в таблицу маршрутизации в качестве следующего маршрутизатора в пути будет включен маршрутизатор M4.

Если маршрутизатор обнаружит изменения в состоянии одного из подключенных к нему каналов, он должен разослать сообщения LSA всем своим соседям, которые ретранслируют их далее по всему домену маршрутизации. С момента посылки сообщения базы данных маршрутизаторов теряют синхронизацию и, следовательно, должны быть синхронизованы вновь. После выполнения синхронизации каждый маршрутизатор должен заново построить карту сетевой топологии, дерево кратчайших путей и таблицу маршрутизации.

Хотя протокол OSPF решает все проблемы, присущие RIP, он тоже далек от идеала. В очень больших сетях протокол OSPF порождает чрезвычайно много маршрутной информации. Например, в распределенной сети с сотнями маршрутизаторов изменение состояния одного канала (скажем, обрыв линии связи) вызывает распространение тысяч сообщений LSA через всю сеть. База данных состояния канала в таких сетях может достигать нескольких мегабайт.

В то же время, при изменениях в сетевой топологии все маршрутизаторы должны заново сформировать таблицу маршрутизации. Поэтому в очень больших сетях время сходимости может значительно возрасти (хотя медленная сходимость неизбежна в больших распределенных сетях при любом протоколе маршрутизации). Кроме того, значительный объем вычислений предъявляет серьезные требования к аппаратным ресурсам маршрутизатора (скорости обработки и объему памяти).

Частично протокол OSPF решает эти проблемы, вводя понятие *области маршрутизации,* иначе называемые *областями OSPF*. Тем самым большая сеть как бы разбивается на несколько областей с независимой маршрутизацией. Такой областью может быть сеть в одном здании, корпусе и т. д. В сети может существовать практически неограниченное число областей маршрутизации.

Маршрутизаторы внутри одной области OSPF не обмениваются информацией с маршрутизаторами другой области. Это уменьшает объем служебного трафика между маршрутизаторами и сокращает размер баз данных маршрутизаторов. Например, в большой распределенной сети, состоящей из 500 маршрутизаторов, создание. 10 областей, в каждой из которых работает 50 маршрутизаторов, означает, что каждому маршрутизатору необходимо поддерживать информацию о состоянии каналов только для 50 маршрутизаторов, а не для 500.

Области OSPF связываются друг с другом с помощью специально выделенных маршрутизаторов, которые должны содержать базу данных с информацией об обеих областях. Эти маршрутизаторы называются граничными маршрутизаторами областей (Area Border Routers). Они работают как фильтры для

сообщений маршрутизации, не выпуская их из области. Граничные маршрутизаторы взаимодействуют между собой, используя специальные сообщения LSA, которые содержат краткую информацию об IP-адресах, содержащихся в области. Граничные маршрутизаторы сохраняют эти сообщения в специальной базе данных, которая используется для определения маршрута между областями (такой способ взаимодействия называется inter-area). Граничные маршрутизаторы должны быть достаточно мощными.

В протоколе OSPF введено понятие *автономных систем маршрутизации,* которые представляют собой домены маршрутизации, находящиеся под общим административным управлением и использующие единый протокол маршрутизации. Маршрутизатор, соединяющий две автономные системы, одна из которых использует протокол маршрутизации, отличный от OSPF, называется*пограничным маршрутизатором автономной системы* — ASBR (Autonomous System Boundary Router). Внешние маршруты обрабатываются в два этапа. Сначала маршрутизатор выбирает оптимальный внешний маршрут. Если таких маршрутов несколько (то есть метрики этих маршрутов примерно одинаковы), то выбирается маршрут, который имеет наименьшую стоимость внутреннего пути до ASBR.

Протокол OSPF поддерживает *настраиваемые* метрики, предоставляя администратору возможность присваивать метрику маршрута, основываясь на различных характеристиках, таких как стоимость передачи, надежность, задержка или количество переходов. Число, используемое в качестве метрики пути, может быть задано произвольным образом. По умолчанию в качестве метрики используется время передачи бита, измеряемое в десятках наносекунд. Так линии Ethernet назначается значение 10, а линии со скоростью 56 Кбит/с — 1785. Вычисляемая протоколом OSPF метрика маршрута представляет собой сумму метрик по всему маршруту.

Рис. 10.3 Топология с настраиваемыми метками

Рассмотрим пример (рис. 10.3). Предположим, что станции А необходимо передать данные станции Б и администратор вручную настроил метрики для протокола OSPF. Протокол RIP не учитывает при выборе маршрута скорость каналов связи. Поэтому при использовании протокола RIP данные будут передаваться через низкоскоростной (56 Кбит/с) канал связи, так как он имеет наименьшее количество переходов. В отличие от протокола RIP протокол OSPF принимает во внимание скорость канала связи (если она выбрана в качестве метрики). Поэтому будут задействованы именно высокоскоростные каналы, так как их суммарная стоимость составит 20.

Протокол OSPF способен работать в сетях следующих типов:

- **Сети точка-точка.** Используются для связи двух маршрутизаторов. Обычно, для организации такой связи используются коммутируемые или выделенные каналы связи;

- **Широковещательные сети.** Примером может служить Ethernet. В этой сети с помощью широковещательной адресации можно передавать информацию одновременно всем маршрутизаторам.
- **Нешироковещательные сети** (NBMA, Non-Broadcast Multiple Access Networks, нешироковещательные сети со множественным доступом). Примером таких сетей могут служить X.25, Frame Relayили ATM.

При включении маршрутизатору необходимо синхронизовать свою базу данных со всеми маршрутизаторами локальной сети. Так как база данных на всех маршрутизаторах одинакова, достаточно провести синхронизацию с одним из них. Этот маршрутизатор называется *назначенным маршрутизатором* — Designated Router (DR). Его «заместитель» называется *резервным назначенным маршрутизатором* — Backup Designated Router (BDR). При этом DR и BDR будут единственными маршрутизаторами, с которыми новый маршрутизатор должен синхронизировать свою базу. После синхронизации базы с назначенным маршрутизатором, новый маршрутизатор будет синхронизован со всеми маршрутизаторами данной сети. Кроме того, назначенный маршрутизатор делает объявления о сетевых связях, перечисляя своих соседей по подсети. Остальные маршрутизаторы просто объявляют о своей связи с назначенным маршрутизатором. BDR занимает место DR в тех случаях, когда последний выходит из строя. Достоинство этого метода в том, что после назначения DR значительно уменьшается количество маршрутной информации, передаваемой по сети. Протокол OSPF состоит из трех внутренних протоколов: Hello, Exchange и Flooding.

Протокол Hello использует сообщения Hello. Периодически маршрутизаторы обмениваются между собой этими сообщениями. Сообщение Hello посылается через все порты маршрутизатора и содержит следующую информацию:

- приоритет маршрутизатора, который используется для выбора DR и BDR;
- интервал между сообщениями Hello в секундах;
- интервал ожидания сообщения;
- список маршрутизаторов, от которых недавно были получены сообщения Hello;
- маршрутизаторы, которые в настоящий момент являются DR и BDR.

Наиболее важны интервал между сообщениями Hello и интервал ожидания сообщения. В сообщениях Hello маршрутизатор передает некоторые сведения о себе и говорит о том, кого он рассматривает в качестве своих ближайших соседей. Маршрутизаторы с разными рабочими параметрами игнорируют сообщения Hello друг от друга, поэтому маршрутизаторы с некорректно настроенными параметрами не влияют на остальные. Маршрутизаторы посылают сообщения Hello всем своим соседям, по крайней мере, один раз на протяжении интервала времени Hello. Если интервал ожидания сообщения истек, а сообщение Hello от соседа не пришло, то считается, что сосед неработоспособен, и рассылается объявление о сетевых каналах, для того чтобы в сети произошел пересчет маршрутов. На рис. 10.4 показан формат сообщения Hello.

<table>
<tr><td colspan="3">Общий заголовок OSPF (поле «Тип» равно)</td></tr>
<tr><td colspan="3">Маска подсети (32 бита)</td></tr>
<tr><td>Интервал Hello (32 бита)</td><td>Опции (8 бит)</td><td>Приоритет (8 бит)</td></tr>
<tr><td colspan="3">Интервал Dead (32 бита)</td></tr>
<tr><td colspan="3">DR 32 бита)</td></tr>
<tr><td colspan="3">BDR (32 бита)</td></tr>
<tr><td colspan="3">Сосед (32 бита)</td></tr>
<tr><td colspan="3">Сосед (32 бита)</td></tr>
</table>

Рис. 10.4 Формат сообщения Hello

Поле «Маска подсети» указывает маску подсети порта, через который посылается сообщение. Это поле устанавливается в шестнадцатеричное значение %FF000000 для сетей класса A, %FFFF0000 для сетей класса B и %FFFFFF00 для сетей класса C.

Значения «Интервал Hello» и «Интервал Dead» устанавливаются администратором. Поле «Приоритет» используется при выборе DR и BDR. Каждому маршрутизатору назначается приоритет в пределах от 0 до 255. Выбирается маршрутизатор, имеющий наибольший приоритет. Если маршрутизатор с наибольшим приоритетом выходит из строя или просто отключен, выбирается другой маршрутизатор с наибольшим приоритетом. Он будет оставаться маршрутизатором DR, даже если старый DR вновь будет функционировать. Но маршрутизаторы, у которых приоритет равен нулю, никогда не будут избраны DR. В сетях точка-точка выбор DR не производится.

Начальная синхронизация баз данных нового маршрутизатора и DR выполняется с помощью протокола Exchange. Работу данного протокола можно разделить на два этапа. На первом этапе определяются роли двух маршрутизаторов: один становится доминирующим, другой — доминантным. После этого на втором этапе происходит взаимный обмен базами данных с помощью пакетов DDP (DatabaseDescription Packet, пакет описания базы данных). После начальной синхронизации с помощью протокола Exchange ответственность за поддержку синхронизации баз данных состояния каналов связи всех маршрутизаторов несет протокол Flooding.

В широковещательных сетях и сетях точка-точка сообщения Hello посылаются соседям с использованием групповой IP-адресации. В этом случае соседние маршрутизаторы обнаруживаются динамически. В сети, в которой широковещательная передача не поддерживается, сообщения Hello посылаются по списку адресов маршрутизаторов в сети. Данный список ведется на каждом маршрутизаторе, который может (потенциально) стать DR.

Для распространения по сети данных о состоянии каналов связи маршрутизаторы обмениваются специальными сообщениями LSA, называемыми Network Links Advertisement (объявления о каналах сети, NLA), — объявлениями о каналах и их состоянии. Маршрутизаторы обмениваются не только своими, но и чужими объявлениями о каналах, получая, в конце концов, информацию о состоянии всех каналов в сети. По этой информации строится ориентированный граф связей сети (он один и тот же для маршрутизаторов). Кроме информации о соседях маршрутизаторы в своих объявлениях перечисляют IP-подсети, с которыми они связаны непосредственно. Маршрутизатор вычисляет путь не до подсети, а до маршрутизатора, к которому эта подсеть подключена. Каждый маршрутизатор имеет уникальный идентификатор, который передается в объявлении о состоянии каналов.

Для обмена маршрутной информацией протокол OSPF использует групповую адресацию. В этих целях зарезервированы два IP-адреса класса D:

- **224.0.0.5** — все маршрутизаторы, работающие по протоколу OSPF, должны поддерживать передачу и получение дейтаграмм по этому групповому адресу;
- **224.0.0.6** — все DR и BDR должны поддерживать получение дейтаграмм с данным групповым адресом.

В соответствии с уже рассмотренными правилами преобразования групповых адресов протокола IP в физические адреса канального уровня используемые групповые адреса преобразуются в следующие физические адреса (табл. 10.2).

Таблица 10.2.

Соответствие между групповыми и физическими адресами протокола OSPF

Адрес класса D	**Физический адрес**

224.0.0.5	%01-00-5E-00-00-05
224.0.0.6	%01-00-5E-00-00-06

Необходимо отметить, что протокол OSPF не использует групповую передачу в сетях NBMA. Все сообщения протокола OSPF начинаются со стандартного заголовка (рис.10.5).

Версия (8 бит)	Тип (8 бит)	Длина сообщения (16 бит)
Идентификатор маршрутизатора (32 бита)		
Идентификатор области (32 бита)		
Контрольная сумма (16 бит)		Идентификатор алгоритма аутентификации (16 бит)
Аутентификация (32 бита)		
Аутентификация (продолжение предыдущего поля — 32 бита)		

Рис. 10.5 . Формат заголовка сообщений OSPF

Поля в заголовке имеют следующие значения:

- **«Версия»** указывает на версию протокола OSPF. Текущая версия — 2;
- **«Тип»** указывает на тип сообщения протокола OSPF;
- **«Длина сообщения»** — длина сообщения в байтах;
- **«Идентификатор маршрутизатора»** — это может быть IP-адрес, выбранный для идентификации маршрутизатора;
- **«Идентификатор области»** используется для указания области. Общей практикой является выбор IP-адресов в качестве идентификаторов;
- **«Контрольная сумма»** вычисляется для всего сообщения, исключая 8- байтовое поле «Аутентификация»;
- **«Идентификатор алгоритма аутентификации»** может принимать только два значения: 0 — аутентификация не используется, 1 — простая аутентификация.

При простой аутентификации поле «Аутентификация» содержит пароль, состоящий из восьми символов. Администратор может задать различные пароли для каждой сети. Это является хорошей гарантией защиты от случайных ошибок, например, от некорректной настройки маршрутизатора. Однако это не очень надежный метод защиты от намеренного вмешательства.

Практика показывает, что не существует сетей, в которых используется исключительно протокол OSPF. Совместно с ним применяются протоколы EGP, BGP, RIP IP или фирменные протоколы производителей.

11. ЗАГОЛОВОК И ПРОТОКОЛ UDP

11.1 Протокол UDP

Протокол пользовательских дейтаграмм (User Datagram Protocol, UDP) описан в документе RFC 768. Протокол разработан для предоставления прикладным программам транспортных услуг. UDP, так же, как и IP, обеспечивает негарантированную доставку дейтаграмм получателю и не поддерживает установку соединений. В модели стека TCP/IP он располагается над протоколом IP.

Взаимодействие между прикладными программами и протоколом UDP осуществляется через так называемые *протокольные порты.* Протокольные порты определяют соответствие между абстрактными точками доступа к протоколу UDP и конкретными прикладными программами. Механизм протокольных портов позволяет рабочей станции одновременно поддерживать несколько сеансов связи с удаленными компьютерами и программами в сети. Можно также сказать, что протокольный порт служит для указания программы-получателя информации. Когда рабочая станция получает дейтаграмму с ее IP-адресом, она направляет эту дейтаграмму конкретной программе, используя номер протокольного порта, который определяется во время установки сеанса связи. Следует отметить, что протокольные порты протокола UDP отличаются от протокольных портов протокола TCP.

Назначение портов происходит при участии сетевой операционной системы. Большинство операционных систем обеспечивают параллельный доступ к протокольным портам. Протокольный порт идентифицируется целым положительным числом. Для связи с протокольным портом на другой рабочей станции, отправитель должен знать IP-адрес получателя и номер порта на этой рабочей станции. Каждое сообщение содержит также номер протокольного порта отправляющей рабочей станции. Таким образом, прикладная программа, получающая сообщения, может напрямую ответить отправителю.

В стеке протоколов TCP/IP протокол UDP обеспечивает транспортный механизм, используемый прикладными программами для передачи дейтаграмм другим приложениям. Протокол UDP предоставляет протокольные порты, используемые для раздельной работы нескольких приложений, выполняющихся на одной рабочей станции. При этом приложения, использующие транспорт протокола UDP, должны сами обеспечивать надежность передачи сообщений. Каждое сообщение протокола UDP называется *пользовательской дейтаграммой.* Она состоит из двух частей: заголовка и области данных. Заголовок содержит четыре 16-битных поля, которые определяют протокольный порт отправителя, протокольный порт получателя, длину сообщения и контрольную сумму (рис. 11.1).

Рис. 11.1 Формат пользовательской дейтаграммы протокола UDP

Поля «Порт отправителя» и «Порт получателя» содержат 16-битные номера портов. Поле «Порт отправителя» может не использоваться, тогда оно должно содержать нули. Поле «Длина сообщения» указывает количество байтов в пользовательской дейтаграмме. При этом учитывается длина заголовка протокола UDP и длина поля данных.

Контрольная сумма пользовательской дейтаграммы может вычисляться, а может и не вычисляться. Нулевое поле «Контрольная сумма» означает, что контрольная сумма не вычислялась. Контрольная сумма, как правило, не вычисляется при работе протокола UDP в высоконадежной локальной сети. Однако в ненадежной сети только контрольная сумма может указать на достоверность и целостность пришедших данных — ведь протокол IP не вычисляет контрольную сумму поля данных IP-дейтаграмм. Для расчета контрольной суммы пользовательской дейтаграммы необходима дополнительная информация. Для этой цели к началу пользовательской дейтаграммы приписывают псевдозаголовок и добавляют в конец этой дейтаграммы байт, заполненный нулями, так, чтобы число бит во всем сообщении было кратно 16. После этого вычисляется контрольная сумма полученной дейтаграммы. Концевое дополнение из нулей и псевдозаголовок не передаются вместе с пользовательской дейтаграммой. Для вычисления контрольной суммы полученной пользовательской дейтаграммы сначала сохраняется ноль в поле «Контрольная сумма», затем вычисляется 16-битная сумма, включая псевдозаголовок, заголовок самой дейтаграммы и данных. При получении пользовательской дейтаграммы контрольная сумма должна проверяться. При этом

используется IP-адрес назначения, полученный из заголовка IP-дейтаграммы, которая содержала пользовательскую дейтаграмму протокола UDP. Если контрольные суммы одинаковы, то пользовательская дейтаграмма действительно достигла нужного получателя и нужный протокольный порт на станции получателя. На рис. 11.2 показан формат псевдозаголовка.

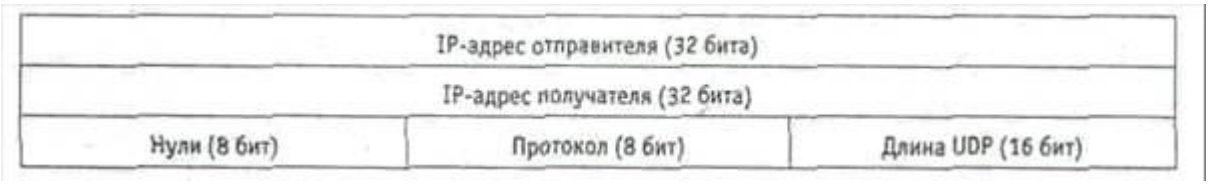


Рис. 11.2 Формат псевдозаголовка

Псевдозаголовок имеет длину 12 байт. Поле «Протокол» содержит код типа протокола. Для проверки контрольной суммы получатель должен извлечь эти поля из IP-заголовка, сформировать свой псевдозаголовок и вычислить контрольную сумму. Данные протокола UDP инкапсулируются в IP-дейтаграммах при передаче их по сети (рис. 11.3).

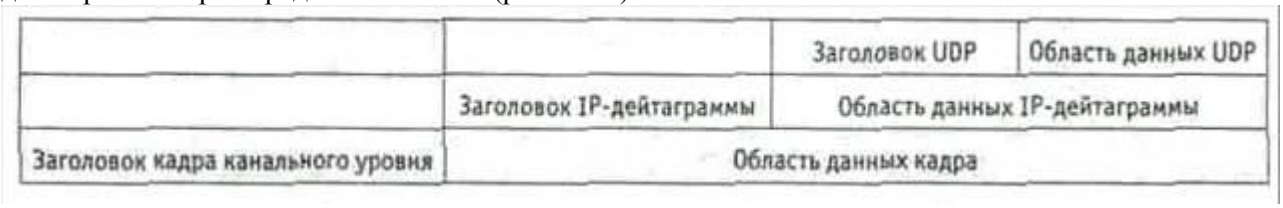


Рис. 11.3 Схема инкапсуляции сообщений UDP

Это означает, что только IP-заголовок определяет отправителя и получателя, а заголовок пользовательской дейтаграммы протокола UDP определяет протокольные порты приложений.

Протокол UDP принимает дейтаграммы от многих прикладных программ и передает их соответствующим прикладным программам на устройствах-получателях. Программное обеспечение UDP обеспечивает мультиплексирование и демультиплексирование дейтаграмм. Операционная система должна выделить каждой прикладной программе порт и сообщить ей его номер. После этого прикладная программа может посылать дейтаграммы с указанием номера протокольного порта. Протокол UDP принимает приходящие с уровня IP (в работе UDP принимают участие и сами дейтаграммы IP, см. выше) пользовательские дейтаграммы протокола UDP и демультиплексирует их по протокольным портам назначения. На рис. 11.4 показан пример демультиплексирования.

Рис. 11.4 общая схема демультиплексирования пользовательской дейтаграммы

Порт UDP можно представить в виде очереди. Операционная система создает внутреннюю очередь, которая хранит приходящие сообщения. Если поступило сообщение с номером протокольного порта, которого нет среди используемых протокольных портов, оно удаляется и высылается сообщение протокола ICMP «Получатель недостижим» с кодом «Порт недостижим».

Некоторые номера протокольных портов стандартизированы. Эти номера выделяет центральный орган — организация IANA (Internet Assigned Numbers Authority). Она регулярно публикует список назначений. Эти протокольные порты зарезервированы и их использование контролируется IANA. В большинстве систем они могут использоваться только системными процессами или программами, выполняемыми привилегированными пользователями. Несколько лет назад эти протокольные порты использовали диапазон номеров от 0 до 255. Однако недавно IANA расширила этот диапазон — теперь она отвечает за назначение портов с номерами от 0 до 1023.

Остальные порты могут назначаться динамически. Сетевое программное обеспечение назначает протокольный порт, когда программа в нем нуждается. Такие протокольные порты не контролируются IANA и могут свободно использоваться пользовательскими процессами. Номера этих протокольных портов лежат в диапазоне от 1024 до 65 535. Протокольные порты в диапазоне от 1024 до 5000 называются *временными* (ephemeral). Хотя IANA не контролирует использование этих протокольных портов, она поддерживает информацию о них в интересах сообщества пользователей Internet.

Для получения информации о текущем назначении протокольных портов необходимо послать соответствующий запрос.

12. ПРОТОКОЛ TCP

Для достижения высокой производительности конечных систем и сети в целом очень важно правильно выбрать транспортный протокол. Транспортный протокол обеспечивает интерфейс между приложениями и сетевым оборудованием. Он выполняет несколько функций, в том числе, позволяет приложениям запрашивать желаемое качество обслуживания. Сетевые приложения фирмы Microsofttребуют, чтобы сетевые драйверы обеспечивали как режим передачи данных с гарантией доставки, так и негарантированную доставку. При передаче с гарантией доставки потеря данных исключается, так как постоянно проводится подтверждение успешности приема. Если подтверждение не получено, производится повторная посылка данных. Ориентированные на предварительное установление соединения транспортные протоколы, такие как TCP (Transmission Control Protocol), делят общий поток данных приложения на раздельные логические потоки и могут дифференцированно распределять ресурсы между ними, что позволяет эффективно использовать общую пропускную способность.

Протокол управления передачей данных TCP описан в документе RFC 793. Протокол TCP — это основной транспортный протокол стека протоколов TCP/IP. Он обеспечивает надежную передачу данных, базируясь на услугах протокола IP. На протокол TCP, в частности, возложена задача управления потоками и перегрузками, что подразумевает согласование скорости передачи данных с техническими возможностями рабочей станции-получателя и промежуточных устройств.

Поступающие к получателю данные буферизуются средствами протокола TCP. Перед отправкой данные также буферизуются.

Для достижения необходимой для данного логического соединения пропускной способности важно правильно выбрать реализацию протокола TCP и оптимально настроить его параметры, влияющие на производительность. В конечном счете пропускная способность логических соединений определяет, насколько быстро приложения обмениваются данными по сети. Такие протоколы прикладного уровня, как HTTP и FTP, передают протоколу TCP свои данные для транспортировки. Поэтому скоростные характеристики TCP оказывают непосредственное влияние на производительность приложений.

Кроме того, на уровень перегрузок, возникающих в сети, очень влияют правила транспортного протокола для передачи и повторной передачи данных. Протокол TCP активно используется и поддерживается

большинством приложений, которые опираются на стек протоколов TCP/IP. Протокол TCP входит в состав транспортных протоколов, поддерживаемых операционной системой MicrosoftWindows NT 5.0 (Windows 2000). На рис. 12.1 схематично показана сетевая модель этой операционной системы и место протокола TCP в ней.

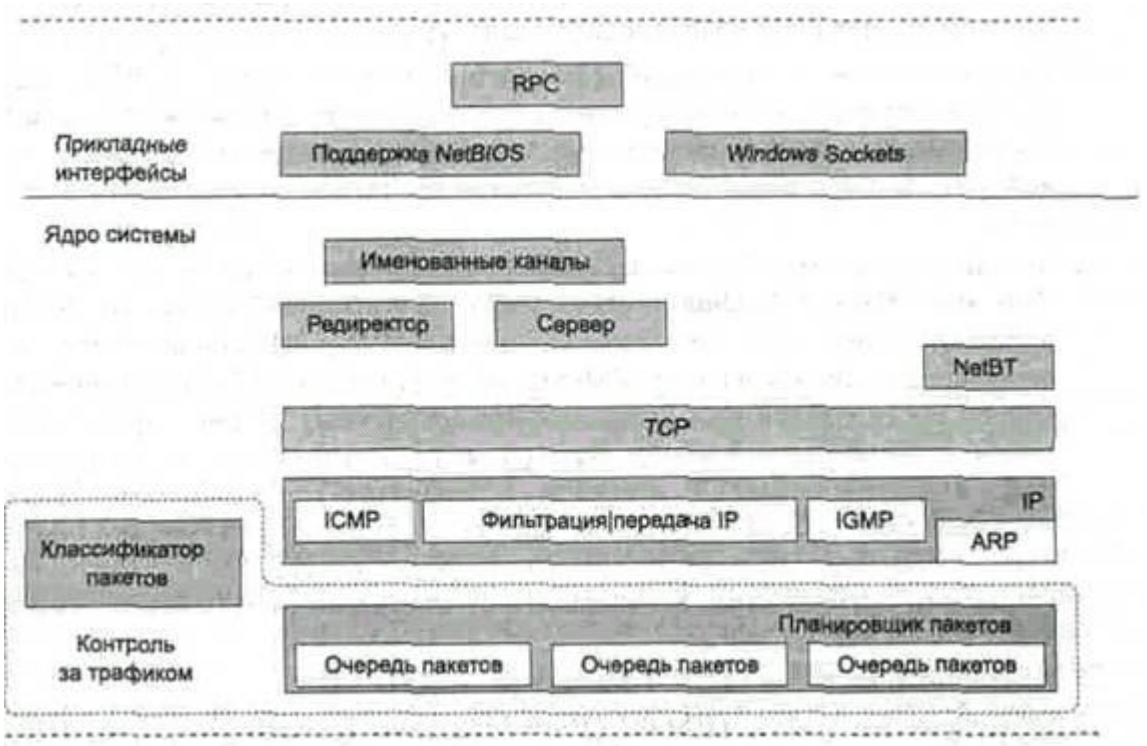


Рис. 12.1 Протокол TCP в операционной системе Microsoft Windows NT 5.0

Поддержка TCP позволяет пользователям Microsoft Windows NT совместно работать через сеть с использованием всех доступных сервисов. В операционной системе Microsoft Windows NT 5.0 поддержкаTCP значительно расширена. Реализация стека протоколов TCP/IP для операционных систем Microsoft Windows NT 4.0/5.0 пользуется параметрами, которые хранятся в системном реестре. Эти параметры можно корректировать вручную, но, вообще говоря, данные реализации протоколов являются самонастраивающимися.

Коротко рассмотрим основные расширения протокола TCP в операционной системе Microsoft Windows NT 5.0:

- **Выборочное подтверждение (TCP Selective Acknowledgment, SACK — RFC 2018).** Это расширение сравнительно недавно утвердил консорциум IETF. Оно позволяет подтверждать прием данных не в порядке их поступления, как это было раньше, а выборочно. Этот подход имеет два главных преимущества. Во-первых, повышается эффективность повторных передач дейтаграмм протоколаTCP благодаря сокращению времени повторной передачи. Протокол TCP использует алгоритм повторной передачи, который учитывает, в каком порядке поступают подтверждения. Такой подход вполне приемлем, однако в этом случае на восстановление каждого потерянного сегмента уходит примерно один цикл обращения. Механизм SACK позволяет осуществлять повторную передачу сразу нескольких потерянных сегментов в одном цикле. Во-вторых, благодаря выборочным подтверждениям, протокол TCP, как показали проведенные исследования, точнее оценивает доступную ширину полосы пропускания в условиях нескольких последовательных потерь пакетов и способен обойтись без выполнения алгоритма «Медленного старта»;

- **Подтверждение с задержкой (Delayed Acknowledgments — RFC 1122).** В соответствии с этим алгоритмом подтверждения высылаются, если не было выслано подтверждение на предыдущий принятый сегмент, или если после получения сегмента в течение 200 мс не поступил следующий;
- **Механизмы масштабирования окон (TCP Receive Window Size Calculation and Window Scaling — RFC 1323).** Протокол не использует жестко заданный размер окна. Он может увеличивать его размер на величину максимального размера сегмента (Maximum Segment Size, MSS), величина которого определяется при установлении соединения. Размер окна приема по умолчанию равен 8 Кбайт. Этот размер выставляется в настройках реестра для протокола TCP, а именно, в параметре TcpWindowSize. Максимальный размер окна — 64 Кбайт. Для сетей Ethernet размер окна обычно равен 8760 байт для версии Microsoft Windows NT 4.0 и 17 520 байт (16 Кбайт, размещенные в двенадцати сегментах по 1460 байт) для версии Microsoft Windows NT 5.0;
- **Временные штампы (TCP Timestamps - RFC 1323);**
- **Обнаружение PMTU (PMTU (Path Maximum Transmission Unit) Discovery — RFC 1191).** Обычно величина MSS равна величине MTU за вычетом 40 байт на заголовки IP и TCP. Сегмент в распределенную сеть отправляется с запретом фрагментации. На отдельных участках сети может быть принят другой MTU. Маршрутизатор, который настроен на этот размер, отсылает сообщение протокола ICMP о недоступности пункта назначения с указанием действующего размера MTU. На основании этого сообщения отправитель изменит значение MTU так, чтобы сегменты смогли достичь получателя. Максимальный размер MTU составляет 68 байт. Предусмотрена возможность работы с маршрутизаторами, которые не совместимы с алгоритмом определения MTU. Для работы с ними в реестре предусмотрены два параметра. PMTU можно определить и вручную — командой ping с ключом запрета фрагментации;
- **Обнаружение неработающего шлюза — маршрутизатора по умолчанию — (Dead Gateway Detection).** Метод описан в документе RFC 816. Проводятся необходимые действия для поиска работающего маршрутизатора взамен отключенного с соответствующей корректировкой таблицы маршрутизации;
- **Политика повторной передачи (TCP Retransmission Behavior).** Число попыток повторной передачи определяется параметром реестра TcpMaxDataRetransmission. По умолчанию этот параметр равен 5. Добавление единицы к этому параметру удваивает значение таймера повторной передачи, которое в начальный период работы соединения равно 3 с;
- **Алгоритмы медленного старта и предотвращения перегрузки (Slow Start Algorithm and Congestion Avoidance — RFC 1122);**
- **Предотвращение синдрома «глупого» окна (Silly Window Syndrome, SWS — RFC 1122).** Получатель всегда старается увеличить окно приема, если имеет свободное буферное пространство. Отправитель также старается увеличить окно передачи при малейшей возможности. В такой ситуации говорить о стабильном потоке сегментов не приходится. Для того чтобы уберечь отправителя и получателя от соблазна «втиснуть в канал лишний сегмент», используется алгоритм предотвращения SWS. Большой объем данных не отправляется до тех пор, пока получатель не объявит размер окна, достаточный для посылки полного сегмента. Кроме того, может производиться настройка, не позволяющая увеличивать окно приема меньше, чем на сегмент;
- **Алгоритм Nagle (Nagle Algorithm, RFC 896).** Алгоритм предназначен для уменьшения количества небольших сегментов в сети. Предпочтение при передаче отдается полноразмерным сегментам.

В данном разделе подробно рассматриваются лишь некоторые из этих расширений. Для того чтобы понять основные механизмы работы всех существующих расширений, остановимся на базовых принципах протокола. За подробностями заинтересованный читатель может обратиться к документам RFC, приведенным в табл. 12.1. Следует отметить, что глубокое понимание существующих механизмов

протокола TCP и того, как они влияют на производительность сети, необходимо для разработки, внедрения, эксплуатации или сопровождения сетей. Это обусловлено тем, что TCP лежит в основе всех современных сетевых операционных систем — и в первую очередь, Windows NT. Так, если внедряется крупная, распределенная сеть на базе этой операционной системы, то проектировщику стоит обратить самое пристальное внимание на планирование загрузки сетевых каналов.

Таблица12.1.

Стандарты RFC, поддерживаемые операционной системой Microsoft Windows NT 5.0

Документ RFC	**Название**
768	User Datagram Protocol (UDP)
783	Trivial File Transfer Protocol (TFTP)
791	Internet Protocol (IP)
792	Internet Control Message Protocol (ICMP)
793	Transmission Control Protocol (TCP)
816	Fault Isolation and Recovery
826	Address Resolution Protocol (ARP)
854	Telnet Protocol (TELNET)
862	Echo Protocol (ECHO)
863	Discard Protocol (DISCARD)
864	Character Generator Protocol (CHARGEN)
865	Quote of the Day Protocol (QUOTE)
867	Daytime Protocol (DAYTIME)
894	IP over Ethernet
919,922	IP Broadcast Datagrams (broadcasting with subnets)
950	Internet Standard Subnetting Procedure
959	File Transfer Protocol (FTP)
1001,1002	NetBIOS Service Protocols

1009	Requirements for Internet Gateways
1034,1035	Domain Name System (DNS)
1042	IP over Token Ring
1055	Transmission of IP over Serial Lines (IP-SLIP)
1112	Internet Group Management Protocol (IGMP)
1122,1123	Host Requirements (communications and applications)
1134	Point-to-Point Protocol (PPP)
1144	Compressing TCP/IP Headers for Low-Speed Serial Links
1157	Simple Network Management Protocol (SNMP)
1179	Line Printer Daemon Protocol
1188	IP over FDDI
1191	Path MTU Discovery
1201	IP over ARCNET
1231	IEEE 802.5 Token Ring MIB (MIB-II)
1256	ICMP Router Discovery Messages
1323	TCP Extensions for High Performance
1332	PPP Internet Protocol Control Protocol (IPCP)
1334	PPP Authentication Protocols
1518	An Architecture for IP Address Allocation with CIDR
1519	Classless Inter-Domain Routing (CIDR): An Address Assignmentand Aggregation Strategy
1533	DHCP Options and BOOTP Vendor Extensions
1534	Interoperation Between DHCP and BOOTP
1541	Dynamic Host Configuration Protocol (DHCP)

1542	Clarifications and Extensions for the Bootstrap Protocol
1547	Requirements for Point-to-Point Protocol (PPP)
1548	Point-to-Point Protocol (PPP) .
1549	PPP in High-level Data Link Control (HDLC) Framing
1552	PPP Internetwork Packet Exchange Control Protocol (IPXCP)
1825	Security Architecture for the Internet Protocol
1826	IP Authentication Header (AH)
1827	IP Encapsulating Security Payload (ESP)
1828	IP Authentication using Keyed MD5
1829	ESP DES-CBC Transform
1851	The ESP Triple DES-CBC Transform
1852	IP Authentication using Keyed SHA
2014	HMAC: Keyed Hashing for Message Authentication
2085	HMAC-MD5 IP Authentication with Replay Prevention
2136	Dynamic Updates in the Domain Name System (DNS UPDATE)
2205	Resource Reservation Protocol (RSVP) — Version 1 Functional Specification
2236	Internet Group Management Protocol, Version 2

12.1 Формат заголовка TCP

Протокол TCP — это основной транспортный протокол в стеке протоколов TCP/IP. Он обеспечивает надежную передачу потока данных, опираясь при этом на ненадежный сервис транспортировки дейтаграмм, предоставляемый протоколом IP. В сетях IP протокол TCP используется для обработки запросов на вход в сеть, разделения ресурсов (файлов и принтеров), репликации информации между контролерами доменов, передачи списков ресурсов и т. д. На протокол TCP, в частности, возложена задача управления потоками и перегрузками. Он отвечает за согласование скорости передачи данных с техническими возможностями рабочей станции-получателя и промежуточных устройств в сети.

Рассмотрим пример использования TCP в неоднородной сети. На рис. 7.2 показана схема движения информации по цепочке: отправитель, сеть Frame Relay, маршрутизатор, получатель. Маршрутизатор является связующим звеном между сетью Frame Relay и сетью Ehternet 802.2, в которой работает получатель.

Как видно из рис. 12.2, протокол TCP выполняет основную задачу по доставке информации.

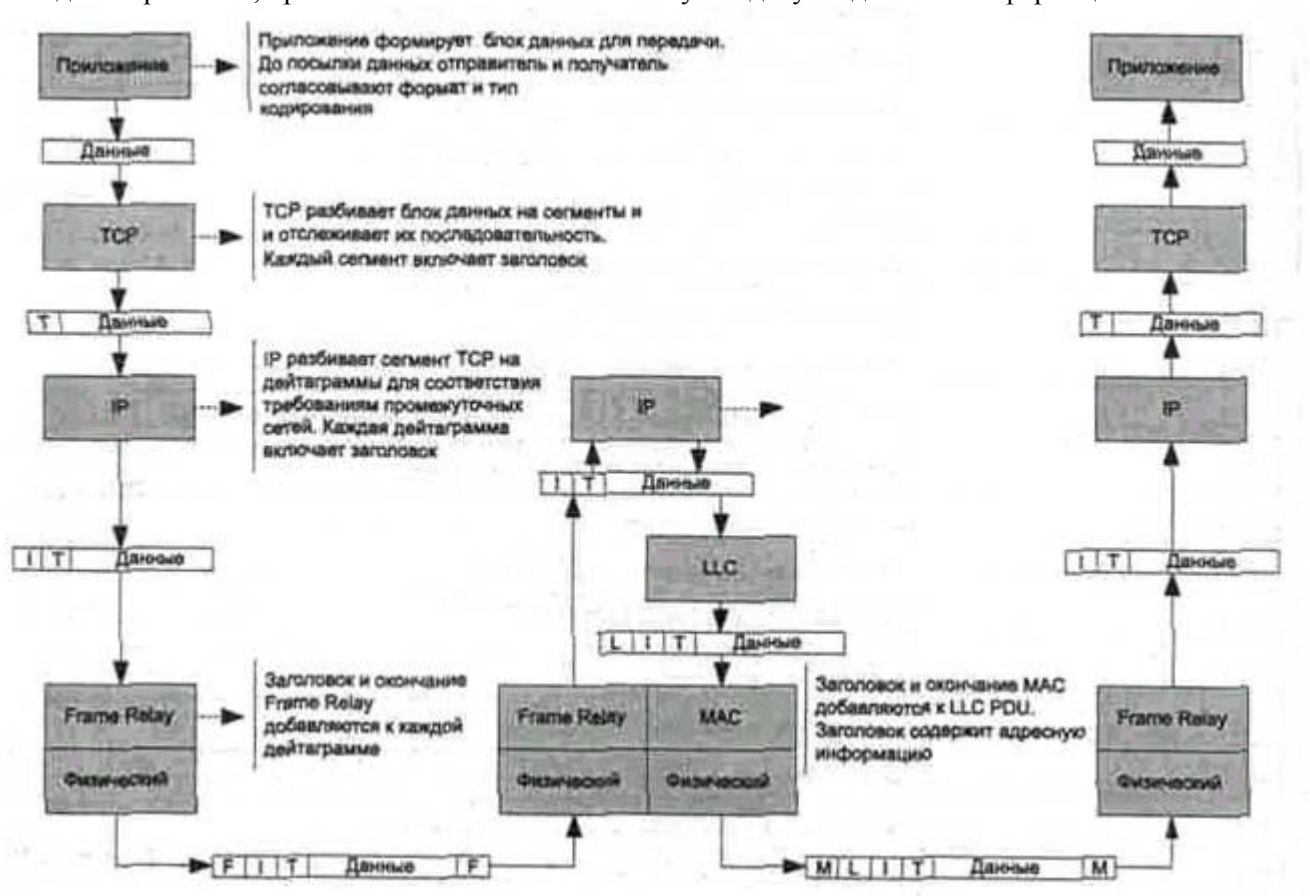


Рис. 12.2 Схема прохождения информации при использовании протокола TCP

С течением времени в протокол TCP постоянно добавляются расширения и появляются более современные его реализации. Этот протокол использует только один тип протокольного блока данных (PDU — Protocol Data Unit), который называется *сегментом TCP*. Сегмент состоит из заголовка и поля данных (полезной нагрузки). На рис. рис. 12.3 показан формат заголовка сегмента TCP.

Минимальная длина составляет 20 байт. Такая большая величина обусловлена тем, что один и тот же заголовок используется протоколом для самых различных целей. Для определения назначения большинства полей предназначены контрольные биты. Формат и значения поля «Контрольные биты» описаны в табл.11.2.

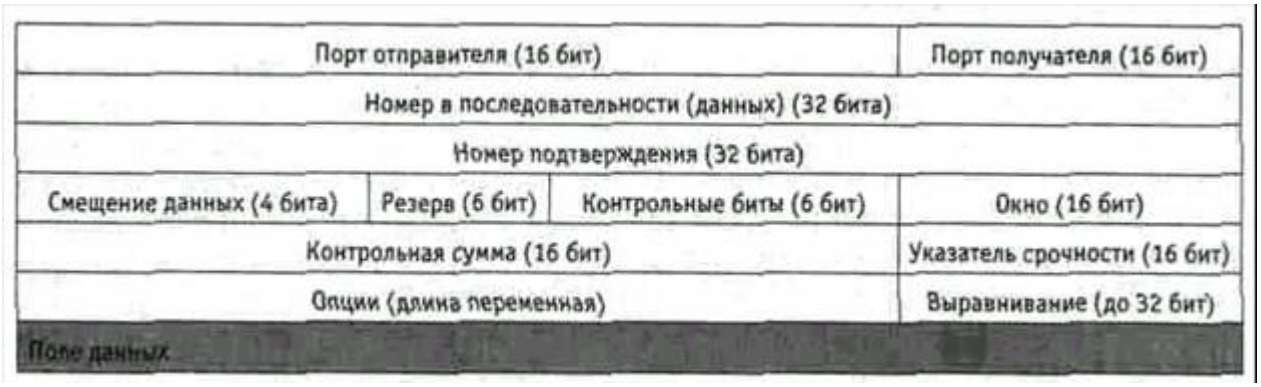


Рис. 12.3 Формат заголовка протокола TCP

Таблица 11.2.

Формат и значения поля «Контрольные биты»

Бит	1	2	3	4	5	6
Сокращение	URG	ACK	PSH	RST	SYN	FIN
Назначение	Поле «Указатель срочности» задействовано	Поле «Номер подтверждения» задействовано	Включена функция проталкивания	Перезагрузка данного соединения	Синхронизация номеров в очереди	Данных для передачи нет

Из рис.12.3 видно, что протокол TCP использует следующие поля:

- **Поле «Номер в последовательности» (Sequence Number)** определяет номер первого байта в очереди (последовательности) байтов в текущем сегменте. Исключение составляют случаи, когда установлен бит (флаг) синхронизации SYN. Тогда это поле обозначает начальный номер в последовательности (Initial Sequence Number, ISN), и первый байт данных имеет номер в очереди ISN+1.
- **Поле «Номер подтверждения» (Acknowledgment Number)** содержит следующий номер в последовательности получаемых подтверждений, который ожидает отправитель в ответ на отосланный сегмент. Иными словами, на отосланный сегмент с данными отправитель ожидает сегмент с подтверждением его успешного приема. В заголовке в поле «Номер в последовательности» занесен указанный номер подтверждения. При этом должен быть установлен контрольный бит подтверждения ACK. Подтверждения (или, как часто говорят, ACK) посылаются постоянно, как только соединение будет установлено.
- **Поле «Смещение данных» (Data Offset)** определяет количество 32-битных слов в заголовке TCP. Тем самым указывается начало поля данных. Заголовок протокола TCP всегда заканчивается на 32-битной границе, даже если он содержит опции.
- **Поле «Резерв» (Reserved)** должно быть заполнено нулями и предназначено для будущего расширения протокола.
- **Поле «Окно» (Window)** содержит объявляемый размер окна в байтах.
- **Поле «Контрольная сумма» (Checksum)** рассчитывается по сегменту, при этом определяется 16-битное дополнение суммы всех 16-битных слов в заголовке и в поле данных. Если сегмент содержит нечетное количество байтов, то он будет дополнен нулями справа до образования 16-битного слова. Этот выравнивающий байт не передается с сегментом по сети, так как может быть «восстановлен» получателем. Контрольная сумма учитывает также 96-битный псевдозаголовок, который ставится перед заголовком протокола TCP. Псевдозаголовок включает следующие поля из заголовка протокола IP: IP-адреса отправителя и получателя, протокол и длину сегмента. С помощью добавления псевдозаголовка протокол TCP защищает самого себя от ошибочной доставки протоколом IP. Так, если протокол IP доставляет сегмент, не предназначенный данному работающему приложению, то модуль протокола TCP на принимающей стороне обнаружит некорректность доставки.
- **Поле «Указатель срочности» (Urgent Pointer)** сообщает текущее значение указателя срочности. Эта положительная величина определяет смещение относительно номера в очереди данного сегмента. Этот указатель сообщает номер байта, следующего за срочными данными, то есть, начиная с этого байта, данные имеют обычный статус срочности. Поле используется совместно с контрольным битом URG.
- **Поле «Опции» (Options)** имеет переменную длину и может вообще отсутствовать. Опции располагаются в конце заголовка протокола TCP и их длина кратна 8 битам. Протокол TCP должен быть готов обрабатывать все виды опций. Опции используются для решения вспомогательных задач, например, для выбора максимального размера сегмента.

- **Поле «Выравнивание» (Padding)** может иметь переменную длину и представляет собой фиктивное поле, используемое для доведения размера заголовка до целого числа 32-битных слов.

Некоторые поля в заголовке сегмента протокола TCP могут быть использованы при его дальнейшем развитии. «Номер в последовательности» и «Номер подтверждения» выравниваются по числу байтов в поле данных, а не по длине всего сегмента. Например, если сегмент содержит в поле «Номер в последовательности» значение 1000 и несет в поле данных 600 байт данных, то поле «Номер в последовательности» указывает на номер первого байта в поле данных. Следующий (в логическом порядке) сегмент будет иметь в поле «Номер в последовательности» значение 1601. Мы видим, что протокол TCP изначально логически ориентирован на работу с потоком данных. Он принимает байты от пользовательского приложения, группирует их, а затем распределяет по сегментам и формирует поток сегментов с нумерацией каждого байта.

Флаги PSH (Push) и URG (Urgent) реализуют две службы протокола TCP: продвижение (проталкивание) потока данных и сигнализацию о срочных данных. Обычно протокол TCP передает данные, когда количество предназначенных для передачи байтов равно длине поля данных очередного сегмента. Приложение может потребовать, чтобы протокол передал все оставшиеся данные и пометил их флагом PSH. На принимающей стороне модуль протокола TCP доставит эти данные приложению сразу (то есть данные не будут ждать своей очереди в буфере). Приложение может прибегнуть к проталкиванию, если достигнут логический конец данных. Поле «Указатель срочности» позволяет информировать приложения на принимающей стороне о том, что в принимаемый ими поток помещены важные данные.

Связь модуля протокола TCP и пользовательского приложения осуществляется при помощи специального интерфейса. Приложение использует ряд команд для управления модулем протокола. Если пользовательское приложение формирует команду SEND для передачи блока данных модулю протокола TCP, то он помещает эти данные в буфер отправки. Если при этом был установлен флаг PSH, любые оставшиеся в буфере данные, включая данные, которые только что были помещены в него, будут немедленно посланы (в одном или нескольких сегментах), а последний сегмент будет помечен флагом PSH. Если этот флаг не установлен, то протокол TCP может держать данные в буфере и посылать их в подходящее время. Например, протокол может ожидать до тех пор, пока в буфере накопится такое количество данных, которое позволит отсылать сегменты с предельно большим полем данных, повышая тем самым эффективность передачи.

Данные в сегменте проходят через соединение к точке входа в модуль протокола TCP и сохраняются в буфере приема, который ассоциируется с этим соединением. Если входящие данные помечены флагом PSH, то они вместе с данными, уже имеющимися в буфере, немедленно доставляются приложению по команде RECEIVE. Если данные не помечены флагом PSH, то протокол TCP доставляет эти данные в подходящее время. Например, протокол может ожидать накопления большего количества данных для уменьшения прерываний в системе.

Документ RFC 793 определил только одну опцию — максимальный размер сегмента. Поле этой опции с размером 16 бит может быть использовано только в сегменте, который посылается при запросе на создание логического соединения. Опция определяет максимальный размер сегмента в байтах, который будет использоваться будущим соединением. За время, прошедшее после публикации документаRFC 793, широкое распространение получили две другие опции: «Параметр масштабирования окна» (Window scale factor) и «Временной штамп» (Timestamp).

Обычно, поле «Окно» в заголовке протокола TCP определяет некий лимит или кредит, выданный отправителю на размещение его данных в отсылаемых сегментах и измеряемый в байтах. Когда используется опция «Параметр масштабирования окна», количество байтов данных, размещаемых в поле «Окно» должно быть кратно 2^F, где F — это параметр масштабирования окна. Максимальное значение F, которое может быть использовано протоколом TCP, — 14. Следует отметить, что эта опция

включается только на время установления соединения. Полное описание этого метода приведено в разделе 2.2 документа RFC 1323.

В этом же документе описан алгоритм «Временной штамп». Этот алгоритм используется для соединений, работающих с окнами большого размера. Опция «Временной штамп» используется для того, чтобы проводить точное измерение времени прохождения сигнала до получателя и обратно, то есть времени обращения (Round Trip Time, RTT).

Такие замеры необходимы для корректировки времени повторной передачи при отсутствии подтверждения приема определенного сегмента. «Временной штамп» реально может быть использован в любом сегменте с данными и определяет два дополнительных поля: «Timestamp Value» (значение временного штампа) и «Timestamp Echo Reply» (ответ на временной штамп получен). Протокол TCP может включить поле «Timestamp Value» в любой исходящий сегмент. Когда этот сегмент подтверждается принимающей стороной, то отвечающий модуль протокола TCP включает в ответный сегмент поле «Timestamp Echo Reply» с тем же самым значением, что было в поле «Timestamp Value» в сегменте, который был подтвержден. Это позволяет протоколу TCP выполнять постоянный мониторинг времени обращения. В операционной системе Windows 2000 предусмотрен режим использования алгоритма «Временной штамп» по умолчанию.

12.2. Протокол TCP. Установление соединения

Состояние системы Соединения протокола TCP переходят из одного состояния в другое в ответ на определенные события — запросы клиента, приход сегментов с флагами SYN, ACK, RST, FIN — или по истечении заданного времени. Соединение может находиться в одном из следующих состояний:

- **LISTEN** — ожидание запроса на соединение со стороны внешних (чужих) портов и внешних (чужих) программ TCP.
- **SYN-SENT** — ожидание парного запроса на установление соединения (со стороны отправителя запрос уже сделан).
- **SYN-RECEIVED** — ожидание подтверждения после того, как запрос на установление соединения уже принят и отправлен.
- **ESTABLISHED** — соединение установлено. Принимаемые от приложения данные можно передать пользователю.
- **FIN-WAIT-1** — ожидание запроса от чужой программы TCP или подтверждение ранее отправленного запроса на закрытие соединения.
- **FIN-WAIT-2** — ожидание запроса на закрытие соединения со стороны чужой программы TCP.
- **CLOSE-WAIT** — ожидание запроса на закрытие соединения со стороны своего клиента.
- **CLOSING** — ожидание подтверждения запроса о закрытии соединения со стороны чужой программы TCP.
- **LAST-ACK** — ожидание ответного запроса на закрытие соединения на посланный запрос о закрытии соединения, который был ранее отправлен чужой программе TCP.
- **TIME-WATT** — соединение находится в этом состоянии на протяжении времени, достаточного для того, чтобы быть уверенным, что чужая программа TCP получила подтверждение своего запроса на закрытие соединения.
- **CLOSED** — соединение закрыто.

Основное состояние соединения — ESTABLISHED (соединение установлено). В этом состоянии происходит обмен данными между абонентами.

12.3 Протокол TCP. Передача данных.

Блок управления передачей Для обеспечения надежной передачи данных по установленным логическим соединениям между прикладными программами протокол TCP должен обеспечивать следующие функции:

- передачу данных;
- проверку достоверности данных при передаче;
- управление потоком данных;
- разделение каналов связи;
- обслуживание установленных соединений;
- соблюдение установленного приоритета пользователей;
- обеспечение соответствующего уровня безопасности.

Для одновременного использования протокола TCP несколькими прикладными программами на одном компьютере он представлен набором адресов и портов. Поскольку идентификаторы портов выбираются каждой программой протокола TCP независимо, то они не будут уникальны. Уникальна совокупность идентификатора порта и IP-адреса. Эта совокупность называется *сокет.* Соединение между отправителем и получателем полностью определяются двумя сокетами на его концах. Это соединение можно использовать для передачи данных в обоих направлениях, то есть оно поддерживает дуплексный режим передачи.

Существует несколько основополагающих концепций связи портов с прикладными программами при любой реализации протокола TCP.

Для сохранения всей совокупности информации, необходимой для создания и поддержки соединения, каждый раз при установлении соединения создается структура данных, называемая блоком управления передачей (Transmission Control Block, TCB). Блок управления передачей TCB хранит всю постоянную информацию по созданному соединению и текущие значения нескольких переменных, например, определяющих очередность отправления. К постоянной информации относятся: номера локального и удаленного сокетов, флаги безопасности и приоритета для данного соединения, указатели на буферы отправки и приема. Блок TCB поддерживает несколько переменных, определяющих очередность отправления и получения сегментов. К ним относятся переменные, связанные с отправкой:

- **SND.UNA** — посылка не подтверждена;
- **SND.NXT** — послать следующий сегмент;
- **SND.WND** — отправить окно;
- **SND.UP** — отправить срочный указатель;
- **SND.WL1** — номер в очереди сегмента, использованный для обновления последнего окна;
- **SND.WL2** — номер подтверждения в сегменте, используемый для обновления последнего окна;
- **ISS** — первоначальный номер в очереди отправки;

и переменные, связанные с получением:

- **RCV.NXT** — получить следующий сегмент;
- **RCV.WND** - получить окно;
- **RCV.UP** — получить срочный указатель;
- **IRS** — первоначальный номер в очереди получения.

Часто используются переменные, берущие свое значение из полей очередного сегмента. К ним относятся:

- **SEG.SEQ** — номер в очереди для сегмента;
- **SEG.ACK** — номер подтверждения для сегмента;
- **SEG.LEN** — длина сегмента;
- **SEG.WND** — окно для сегмента;
- **SEG.UP** — срочный указатель для сегмента;
- **SEG.PRC** — приоритет для сегмента.

На рис. 12.4 показана последовательность этапов отправки и приема данных.

На примере рис.12.4 рассмотрим принцип использования некоторых переменных. Отправитель данных с помощью переменной SND.NXT отслеживает следующий номер сегмента в очереди, подлежащего отправке. Получатель данных с помощью переменной RCV.NXT отслеживает номера прибывающих сегментов. В поле переменной SND.UNA отправитель данных помещает самый старый номер сегмента, который уже был отправлен, но на который еще не получено подтверждение (ACK). Когда отправитель создает и посылает новый сегмент, он увеличивает значение своей переменной SND.NXT. Адресат при получении этого сегмента увеличивает значение своей переменной RCV.NXT и отправляет подтверждение.

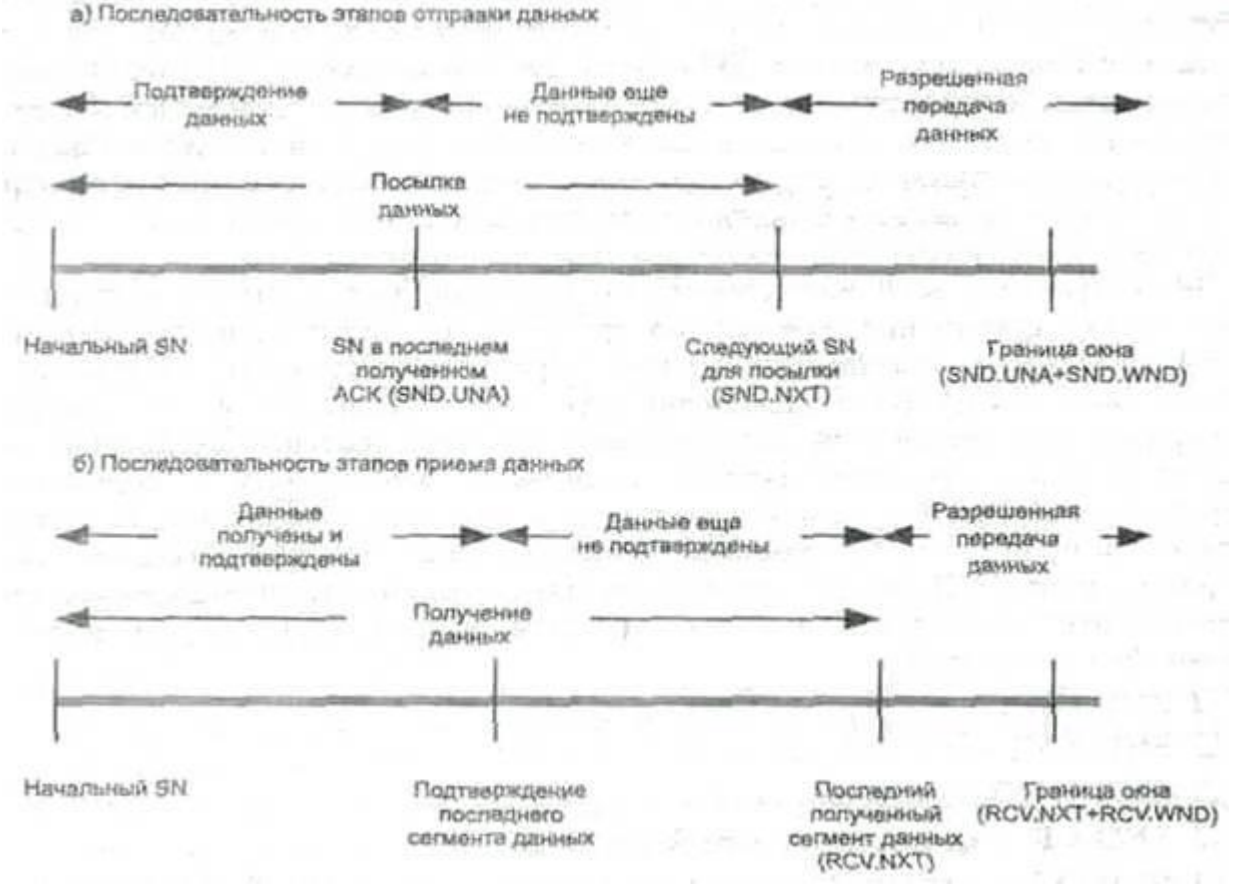


Рис. 12.4 Процесс: а) передачи данных; б) прием данных

При получении подтверждения увеличивается значение переменной SND.UNA отправителя. Разность значений переменных SND.NXT и SND.UNA может служить мерой задержки сегментов в сети. Переменные увеличиваются на длину поля данных в сегменте.

Установление и закрытие соединений При открытии и закрытии соединений в зависимости от поведения сторон могут возникать самые различные ситуации (но так или иначе соединение всегда будет находиться в одном из состояний, перечисленных выше). Упрощенно же процесс открытия соединения можно представить следующей последовательностью действий:

- Инициатор соединения посылает запрос к протоколу TCP на открытие порта для передачи.
- После открытия порта протокол TCP на стороне приложения-инициатора посылает запрос приложению, с которым требуется установить соединение (принимающей стороне).
- Протокол TCP на принимающей стороне открывает порт для приема данных и отсылает квитанцию, подтверждающую прием запроса.
- Принимающая сторона открывает порт для передачи и также передает запрос к противоположной стороне.
- Приложение-инициатор открывает порт для приема и возвращает квитанцию. С этого момента соединение считается установленным. После этого начинается обмен информацией по данному соединению.

При передаче данных по соединению каждый байт информации нумеруется. Нумерация ведется и в очереди отправления и в очереди получения. Первоначальный номер байта в очереди отправки указывается модулем TCP посылающей стороны, а первоначальный номер байта в очереди приема выясняется во время установления соединения. В это время оба модуля протокола TCP должны синхронизировать друг с другом первоначальные номера байтов в очередях. Синхронизация производится путем обмена сегментами, которые используются при установке соединения. Эти сегменты несут флаг синхронизации SYN и исходные номера для обоих очередей. Синхронизация требует, чтобы каждая сторона послала свой собственный первоначальный номер в очереди и получила подтверждение о принятии этого номера. Нумеруются и сами сегменты: номером сегмента считается номер первого байта в поле полезной нагрузки этого сегмента. Рассмотрим синхронизацию номеров на примере создания соединения между станцией А и станцией Б. Для синхронизации необходимо выполнить следующие действия:

1. Станция А посылает сегмент с флагом SYN и своим номером в очереди N станции Б;
2. Станция Б посылает подтверждение — «ваш номер в очереди N» — станции А;
3. Станция Б посылает сегмент с флагом SYN и своим номером в очереди (обозначим его К) станции А;
4. Станция А посылает подтверждение — «ваш номер в очереди К» — станции Б.

Шаги 2 и 3 можно объединить, поэтому такой обмен называется открытием соединения с подтверждением трех сообщений. Эту же процедуру открытия соединения с подтверждением трех сообщений можно показать с фиксацией состояний соединения и переходов между ними (рис. 7.5). Каждая строка на рис. 7.5 пронумерована и показывает состояние соединения. Стрелки «→» означают отправление сегмента от модуля TCP станции А в модуль TCP станции Б. Стрелки «←» показывают отправку сегментов в противоположном направлении. Промежуточное состояние соединения соответствует моменту посылки или получения сегмента. На рис. Рис. 12.5 показано не все содержание сегментов — приведены только номера в очереди, флаги управления и АСК.

Модуль TCP на станции А	Модуль TCP на станции Б
CLOSED	LISTEN
SYN-SENT → <SEQ=100><CTL=SYN>	O SYN-RECEIVED
ESTABLISHED ← <SEQ=300><ACK=101><CTL=SYN, ACK>	M SYN-RECEIVED
ESTABLISHED → <SEQ=101><ACK=301><CTL=ACK>	O ESTABLISHED
ESTABLISHED → <SEQ=101><ACK=301><CTL=ACK><DATA>	O ESTABLISHED

Рис. 12.5 Процедура подтверждения трех сообщений для синхронизации соединения

Станция А указывает, что она будет использовать номер в очереди 100. В ответ станция Б посылает свой номер в очереди 300 и говорит, что ожидает получения номера 101. В последней строке после установления соединения модуль TCP станции А передает некоторую порцию данных.

На рис. 12.6 показана нормальная, штатная процедура закрытия соединения.

Модуль TCP на станции А	Модуль TCP на станции Б
ESTABLISHED	ESTABLISHED
FIN-WAIT-1 → <SEQ=100><ACK=300><CTL=FIN, ACK>	O CLOSE-WAIT
FIN-WAIT-2 ← <SEQ=300><ACK=101><CTL= ACK>	M CLOSE-WAIT
TIME-WAIT ← <SEQ=300><ACK=101><CTL=FIN, ACK>	M LAST-ACK
TIME-WAIT → <SEQ=101><ACK=301><CTL= ACK>	O CLOSED
CLOSED	CLOSED

Рис. 12.6 Нормальная процедура закрытия соединения

Как видно, при закрытии соединения происходит обмен сегментами, несущими управляющий флаг FIN, указывающий на отсутствие данных для передачи.

12.4 Механизм окна TCP. Управление потоком данных

.Плавающее окно

Как и большинство протоколов, осуществляющих управление потоком данных (например, HDLC и X.25), протокол TCP использует механизм плавающих окон. Протоколы HDLC и X.25 используют этот механизм в классическом виде — на каждый отправленный блок данных должно быть получено подтверждение. Протокол TCP несколько отходит от классической схемы.

Основной недостаток классической схемы заключается в том, что только один сегмент может быть передан за один сеанс. В данном случае под сеансом понимается посылка сегмента и получение подтверждения на его успешный прием.

Такая схема не позволяет работать с максимальной производительностью. Эффективность может быть значительно повышена, если разрешить передачу множества сегментов за один сеанс с группировкой всех подтверждений для них в одно.

Рассмотрим схему работы этого метода на примере двух станций — А и Б, которым необходимо обмениваться данными.

Станция Б выделяет буферное пространство для приема *n* сегментов. Поэтому станция Б может принять *n* сегментов, а станция А может послать те же *n* сегментов без необходимости ожидания подтверждений об их приеме. Для отслеживания подтверждений о принятых сегментах каждый из них помечается номером в последовательности. Станция Б подтверждает получение сегмента посылкой подтверждения на него, которое содержит номер в последовательности следующего ожидаемого сегмента. Это подтверждение также косвенным образом извещает станцию А о том, что станция Б готова получить следующие *n* сегментов, начиная с указанного номера. Такая схема работы годится для подтверждения получения множества сегментов.

Например, станция Б получает сегменты 2, 3 и 4, но воздерживалась от отправки подтверждения на сегменты 2 и 3 до получения сегмента 4. Посылая подтверждение с номером в последовательности 5, станция Б подтверждает получение сегментов 2, 3 и 4 за один раз. Таким образом, можно сказать, что станция А ведет список номеров в последовательности сегментов, которые ей разрешено посылать, а станция Б поддерживает список номеров в последовательности сегментов, которые она готова принять.

Эти списки называют *окнами сегментов,* а такую схему передачи сообщений часто называют *управлением потоком с использованием плавающего окна.*

Так как номер в последовательности занимает одно поле в сегменте, то очевидно, что номер не может быть слишком большим. Например, если это поле занимает 3 бита, то номер в последовательности сегментов может иметь значения от 0 до 7. Соответственно, сегменты нумеруются по модулю 8, то есть за номером в последовательности 7 следует номер в последовательности 0.

Таким образом, для поля номера в последовательности, состоящего из *k* бит, границы изменения номера равны 0 и 2^k-l, а сегменты нумеруются по модулю 2^k. С учетом приведенных рассуждений на рис.12.7 показаны значения передаваемых и принимаемых сегментов на принимающей и передающей сторонах с фиксацией границ плавающего окна. В этом примере для простоты размер поля «Номер в последовательности» принят равным 3 битам. Серые прямоугольники указывают сегменты, которые могут быть посланы. В соответствии с рис. 12.7 отправитель может послать 5 сегментов, начиная с сегмента с номером 0. Каждый раз, когда сегмент посылается, ширина окна (серого прямоугольника) уменьшается.

При получении подтверждения ширина окна увеличивается. Сегменты, находящиеся между черной вертикальной чертой и серым прямоугольником (окном) уже были посланы, но еще не были подтверждены. Отправитель должен хранить копии этих сегментов в своем буфере на случай, если потребуется их повторная передача.

Рассмотрим следующий пример (рис. 12.8), позволяющий проследить последовательность обмена информацией. В этом примере для наглядности поле номера в последовательности имеет длину три бита, следовательно, максимальный номер равен 7 и окно не может содержать более семи номеров сегментов.

В начале работы станции А и Б имеют окна длиной семь сегментов. То есть, станция А может передать семь сегментов, начиная с сегмента SO, а станция Б — принять такое же количество сегментов. После передачи трех сегментов (SO, S1 и S2) без подтверждения станция А сокращает размер своего окна до четырех сегментов и сохраняет в буферной памяти копию трех посланных сегментов.

Новый размер окна указывает на то, что станция А может передать четыре сегмента, начиная с сегмента S3. Станция Б после получения сегментов передает сообщение RR3, в котором заложена следующая информация: «Я приняла все сегменты до номера S2 включительно и готова принять сегмент S3; в действительности я готова принять семь сегментов, начиная с сегмента S3».

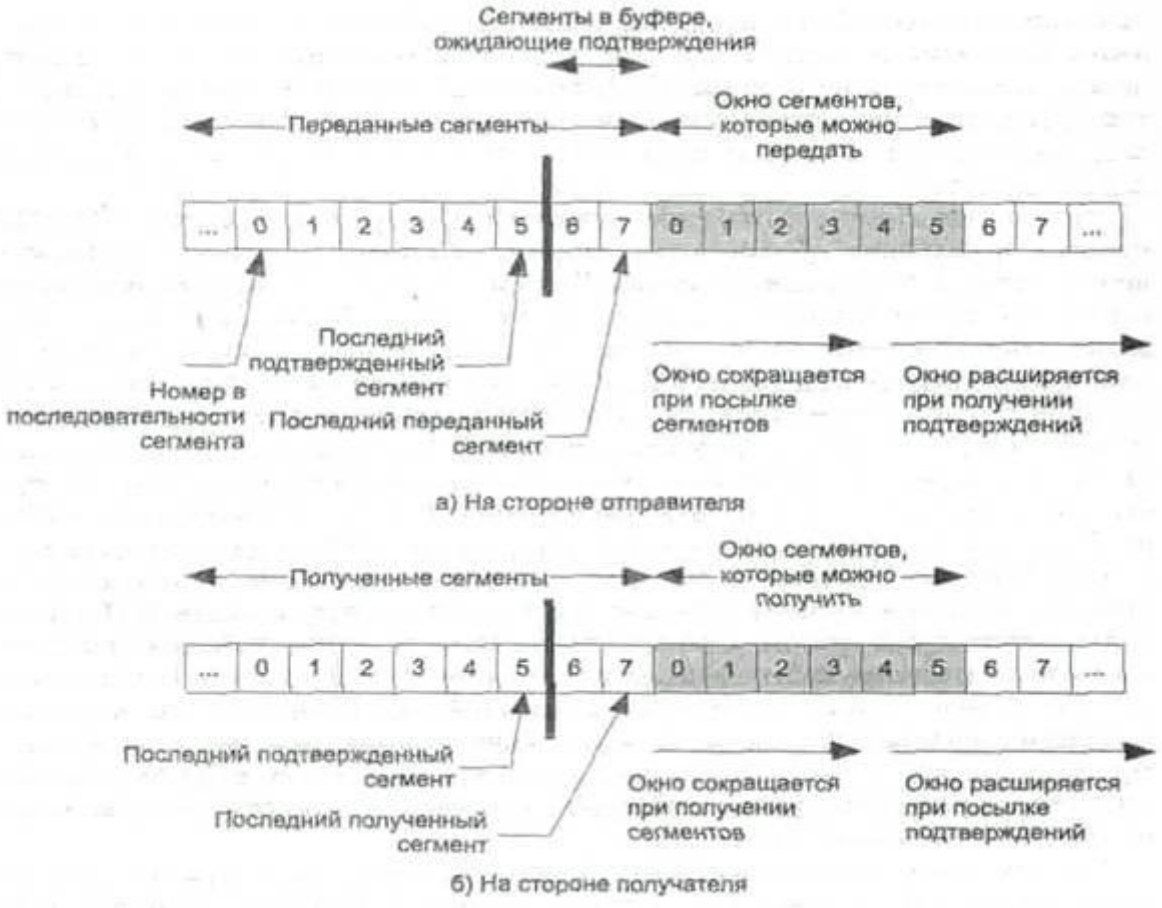


Рис.12.7 Сегменты в потоке данных на передающем и принимающем устройствах

После получения подтверждения с такой информацией станция А считает себя вправе передать семь сегментов, начиная с сегмента S3. Кроме того, станция А может очистить свою буферную память от копий первых трех сегментов, так как они были успешно приняты. Теперь станция А передает сегменты S3, S4, S5 и S6. Станция Б в ответ на получение сегмента S3 отправляет подтверждение RR4 и позволяет производить передачу сегментов с номерами от S4 до S2 (этот сегмент относится уже к следующей последовательности из семи сегментов). На момент получения станцией А этого подтверждения сегменты S4, S5 и S6 уже были посланы и, следовательно, станция А может расширить свое окно и послать четыре сегмента, начиная с сегмента S7.

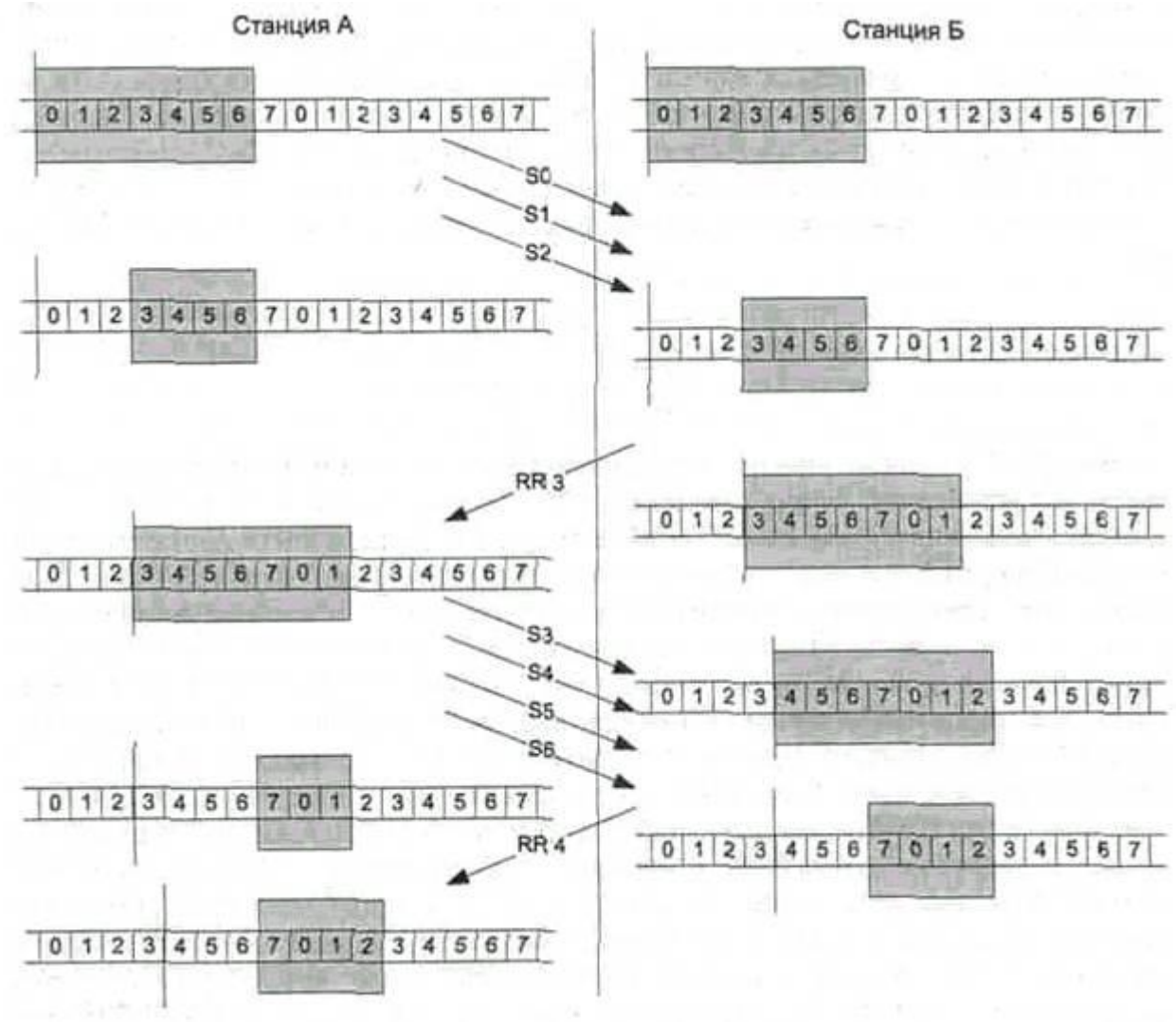


Рис. 12.8 Пример работы механизма плавающего окна

Пропускная способность

Рассмотрим методы определения максимально возможной пропускной способности соединения протокола TCP. Пропускная способность зависит от размера окна передачи, задержки и скорости пересылки данных. Используем следующие обозначения:

- *W* — размер окна передачи в байтах;
- *R* — скорость передачи данных (бит/с) по определенному соединению, доступная на стороне отправителя;
- *D* — задержка (в секундах) при передаче данных между отправителем и получателем через определенное соединение.

Для простоты рассуждений проигнорируем влияние служебных битов в сегменте TCP. Предположим, что отправитель начинает передавать последовательность байтов получателю через установленное соединение. Для того чтобы первый байт достиг получателя, потребуется время, равное *D.* Такое же время — *D* секунд — потребуется для получения подтверждения. В течение этого времени отправитель может передать *2RD бит,* или *RD/4* байт. На самом деле, отправитель ограничен размером окна в *W* байт и не может сдвигать окно, пока не получит подтверждение. Только при *W> RD/4* на этом соединении достигается максимально возможная пропускная способность. Если *W<RD/4,* то близость пропускной способности к максимальной определяется отношением *W к RD/4.* Следовательно, нормированная пропускная способность S может быть выражена как:

$$S = \begin{cases} 1, & W \geq RD/4 \\ \dfrac{4W}{RD}, & W < RD/4 \end{cases}.$$

На рис. 12.9 показан пример определения максимальной пропускной способности в зависимости от произведения *RD.* Максимальный размер окна может составлять 2^{16}-l=65 535 байт. Такой размер окна должен быть достаточен для большинства приложений. В качестве примера давайте разберем три различные технологии, применяемые для передачи сегментов TCP и использующие такой размер окна. Для технологии Gigabit Ethernet с длиной магистрали, равной 100 м, произведение *RD* будет меньше, чем 10^3 бит. На больших расстояниях, пусть даже при более низких скоростях, например, в случае использования канала T1 (1.544 Мбит/с), произведение *RD* становится большим, следовательно, эффективность падает. Тем не менее, она остается, как видно из рис. 7.9, приемлемой — около 0.8. На значительных расстояниях при дальнейшем увеличении скорости (допустим, речь идет о передаче информации с использованием технологии SDH (канал 155 Мбит/с) между двумя городами) рассматриваемая технология становится крайне неэффективной — как видим, нормализованная пропускная способность падает до 0.1. Очевидно, что в данном случае размер окна слишком мал. Необходимо использовать другой параметр масштабирования окна, который позволил бы эффективно задействовать пропускную способность канала. Достаточно увеличить параметр масштабирования окна до 4 — это приведет к значительному увеличению размера окна до 2^{20}-1 » 106 байт.

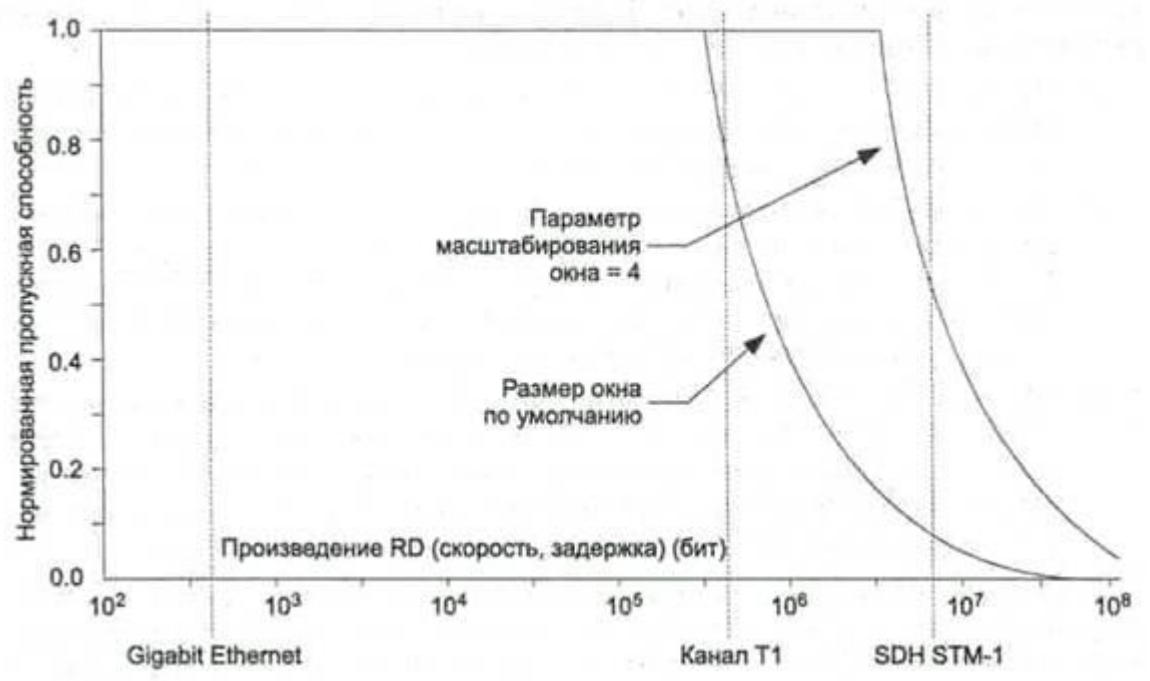


Рис. 12.9 Влияние параметра масштабирования окна на эффективность передачи

Как видим, перечисленные выше параметры оказывают основное влияние на эффективность передачи протокола TCP. Однако существует множество усложняющих факторов, которые также следует принять во внимание. Во-первых, в большинстве случаев соединения TCP мультиплексируются в один канал, так что каждое соединение получает часть его доступной пропускной способности. Это приводит к снижению скорости передачи *R* и, следовательно, к снижению эффективности работы протокола. Во-вторых, большинство соединений TCP проходят через маршрутизаторы. В этом случае время *D* будет равно сумме задержек в каждой сети и задержек на каждом маршрутизаторе в пути. При этом суммарная задержка на маршрутизаторах часто вносит основной вклад во время задержки *D*,особенно при возникновении

перегрузок. В-третьих, значение скорости *R,* используемое в приведенной выше формуле, определяет скорость передачи данных, доступную для соединения, только на стороне отправителя. Если на одном из переходов в пути от отправителя до получателя скорость передачи меньше этой скорости, то попытка передачи на максимальной скорости приведет к образованию узкого места, что неизбежно повысит время *D.* И наконец, в-четвертых, если сегмент теряется, он должен быть передан вновь, что приводит к снижению пропускной способности. Степень влияния потерь сегментов на эффективность передачи зависит от политики повторных передач. В современных распределенных сетях несколько сегментов протокола TCP могут быть потеряны из-за ошибок на линиях. Большинство же сегментов теряются при использовании механизмов сброса пакетов на маршрутизаторах или коммутаторах (например, Frame Relay) в моменты сетевых перегрузок.

Контроль за перегрузками в сетях IP достаточно сложно реализовать по целому ряду причин. К ним можно отнести следующие:

- Протокол IP не ориентирован на установление соединения. Он не обеспечивает обнаружение перегрузки и по этой причине не может быть использован для контроля за перезагрузками.
- Протокол TCP осуществляет контроль потока из конца в конец соединения и может лишь по косвенным признакам определить перегрузку в сети. Более того, так как задержки в распределенных сетях постоянно изменяются, то информация, полученная на основании косвенных признаков (например, размер окна), не является достоверной.
- Не существует распределенного алгоритма для связывания различных протоколов TCP. То есть, протоколы на разных компьютерах не могут взаимодействовать друг с другом для поддержания определенного уровня общего потока. Более того, на самом деле они ведут себя очень «эгоистично» по отношению к свободным ресурсам канала.

Сообщение «Подавление источника» (Source Quench) протокола ICMP можно рассматривать в качестве грубого инструмента, предназначенного для сдерживания потока трафика от отправителя, но его нельзя назвать эффективным методом контроля за перегрузками.

Задачу контроля за перегрузками можно возложить на протокол RSVP, но его широкое распространение — еще вопрос времени.

Протокол TCP может влиять на загрузку сети, управляя потоком данных с помощью плавающего окна, применяя различные методы отправки/приема данных и отсылки подтверждения, следя за уровнем ошибок и используя различные методы повторной передачи. Ниже будут рассмотрены алгоритмы медленного старта и контроля за перегрузками, реализованные в протоколе TCP. Основное предназначение этих алгоритмов — предотвращение перегрузки в сети.

Управление потоком данных использует механизм плавающего окна, но кроме этого, применяется также более гибкая схема приема/передачи данных и отсылки подтверждений на успешный прием данных. Управление потоком протокола TCP использует так называемую схему с выделением лимита на передачу данных. По этой схеме каждый передаваемый байт имеет свой собственный номер в последовательности (SN). Когда протокол TCP посылает сегмент, он выставляет в поле номера в последовательности номер первого байта в поле данных этого сегмента. На принимающей стороне пришедший сегмент подтверждается сообщением, в котором указывается (*A= i, W=j).* Такая запись имеет следующее значение: если величина *A (ACK)* равна г, это значит, что сообщение подтверждает получение всех байтов, вплоть до номера в последовательности *i—1;* следующие ожидаемые байты имеют номер в последовательности *i.* Кроме того, выдается разрешение на посылку дополнительного окна *W (Window)* из *j* байтов; то есть байтов с номерами в последовательности от i до *i +j - 1.* На рис. 12.10 иллюстрируется работа этого механизма. В отличие от рис. 12.9, окна передачи и приема указывают количество байтов данных.

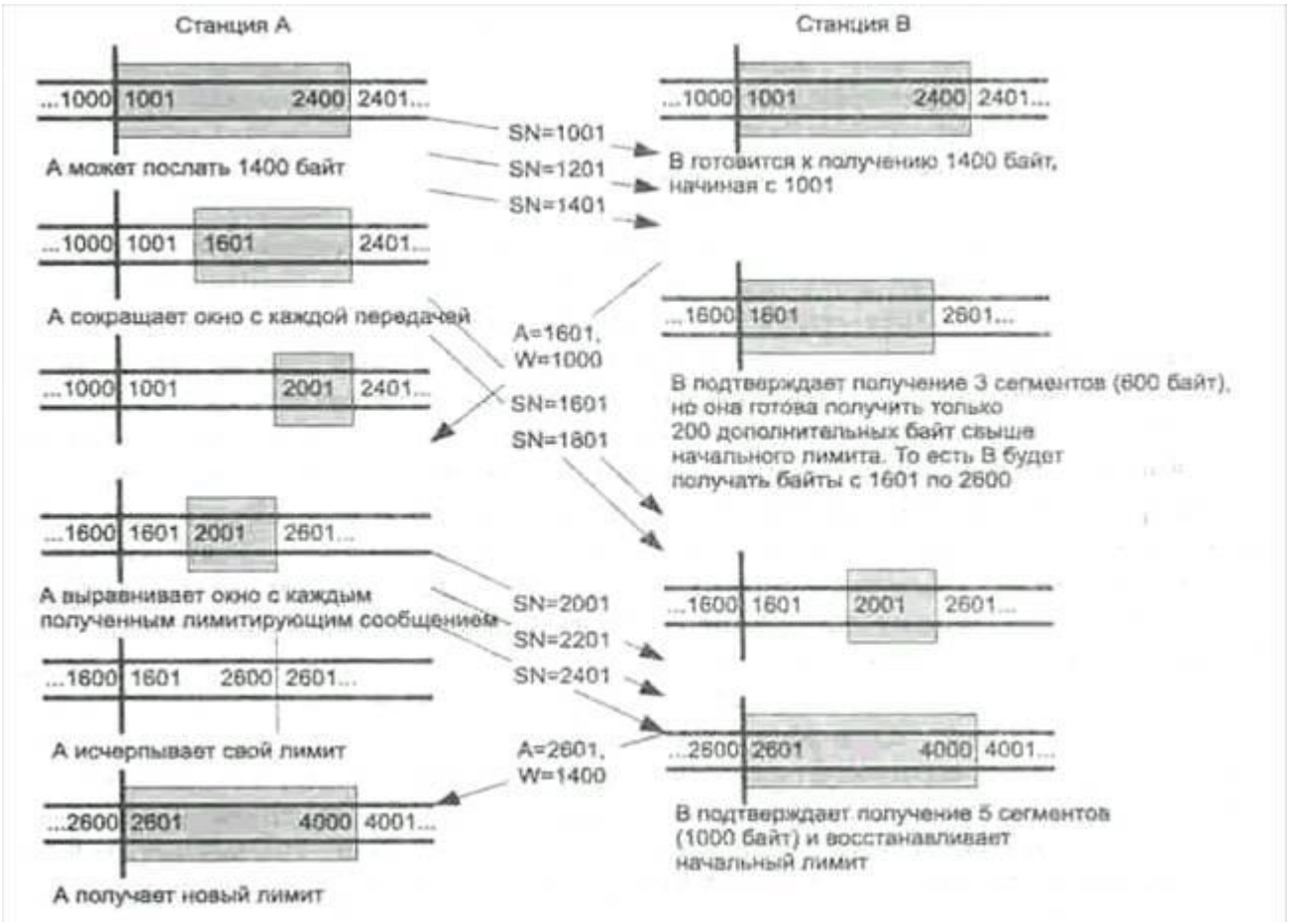


Рис. 12.10 Схема управления потоком данных

Для большей наглядности покажем поток данных, идущий только в одном направлении, и предположим, что в каждом сегменте посылаются 200 байт данных. Во время установления соединения номера в последовательностях отправителя и получателя синхронизированы и станция А имеет начальный лимит на отсылку данных 1400 байт, начиная с номера байта 1001. После посылки 600 байт в трех сегментах станция А уменьшает свое окно отсылки до 800 байт (номера с 1601 до 2400). После получения этого сегмента станция В подтверждает получение всех байтов, вплоть до 1601, и формирует свое окно приема на 1000 байт. Это означает, что станция А может посылать байты, начиная с номера 1601 и заканчивая номером 2600, то есть пять сегментов. Однако к тому моменту, когда сообщение от станции В доходит до станции А, последняя уже успела выслать два сегмента, содержащие байты 1601-2000, что позволял начальный лимит. Следовательно, оставшийся лимит станции А на этот момент составляет всего 400 байт или два сегмента. Во время обмена станция А продвигает левую границу своего окна каждый раз, когда осуществляет передачу. Правая граница передвигается только тогда, когда станция получает новый лимит.

На практике обе стороны одновременно задействуют режимы передачи и приема, так как данные могут передаваться в обоих направлениях (происходит полнодуплексная передача). Механизм выделения лимита является достаточно гибким. Например, рассмотрим ситуацию, при которой последнее сообщение, посланное станцией В, было *(A=i, W=j).* Последним байтом данных, полученным станцией В, был байт с номером *i -1.* Для увеличения лимита до значения *k,* при условии, что *kj* и дополнительные данные не поступали, станция В формирует сообщение (*A= i, W= k).* Для подтверждения входящего сегмента, содержащего *m* байт данных *(m)* без выделения дополнительного лимита, станция В формирует сообщение *(A =i+m, W =j-m).*

Следует отметить, что от получателя не требуется немедленного подтверждения приходящих сегментов. Он может ожидать некоторое время, а затем сформировать подтверждение сразу на несколько сегментов. Получатель должен проводить какую-то политику, регулирующую количество данных, которое он позволяет передавать отправителю. Можно выделить две политики получателя: консервативную и

оптимистическую. Консервативная схема управления потоком основана на том, что лимит выделяется в соответствии с имеющимся доступным буферным пространством. Если это правило применить к ситуации, показанной на рис. 7.10, то первое лимитирующее сообщение говорит о том, что станция B может разместить 1000 байт в своем буфере, а второе — о том, что станция B может разместить 1400 байт. Консервативная схема управления потоком может ограничить пропускную способность логического соединения в ситуации, когда в сети возникают большие задержки.

Получатель может более эффективно использовать пропускную способность канала с помощью оптимистического выделения лимита, сообщая о свободном буферном пространстве, которого он на данный момент фактически не имеет. Например, если буфер получателя заполнен, но он ожидает, что сможет освободить 1000 байт буферного пространства за время прохождения информации из конца в конец соединения, он может послать кредит на 1000 байт. Если получатель может поддерживать скорость, заданную отправителем, то такая схема способна повысить пропускную способность и не принесет вреда. Если же отправитель работает быстрее получателя, то некоторые сегменты будут отбрасываться из-за занятого буфера, что повлечет за собой повторную передачу. В таком случае оптимистическое управление потоком может усугубить ситуацию с перегрузкой в сети.

Политики отправки и приема сегментов. При отсутствии данных, помеченных флагом PSH, протокол TCP на стороне отправителя может сам решать, когда следует осуществить передачу. Когда данные передаются модулю протокола TCP пользовательским приложением, они записываются в передающий буфер. Протокол TCP может создавать сегмент для каждой группы данных, предоставленных приложением, или он может ожидать накопления определенного количества данных и только после этого формировать и отправлять сегмент. То, какая политика отправки лучше, всецело зависит от требований к производительности. Если передачи сегментов происходят редко, но при этом передаются большие объемы данных, то сегменты можно формировать сразу при поступлении данных — накладные расходы здесь будут невелики. С другой стороны, даже если объем передаваемых данных невелик, иногда имеет смысл слать их сразу же при поступлении — при этом накладные расходы велики, но такая система обеспечивает быструю реакцию на изменения состояния сети.

При отсутствии данных, помеченных флагом PSH, протокол TCP на стороне получателя также может самостоятельно решать, когда следует доставить данные пользовательскому приложению. Так, он может доставлять данные при получении каждого сегмента по порядку или заносить данные из нескольких сегментов в буфер приема. Как и в случае с отправкой, выбор политики доставки зависит от требований к производительности. Если данные приходят редко и имеют большой объем, то приложение получит их сразу. С другой стороны, если данные поступают часто и маленькими порциями, то немедленная их передача приложению приведет к неэффективному расходованию ресурсов приложения и протокола TCP.

Если сегменты поступают в порядке их отсылки по установленному соединению, протокол TCP помещает их данные в буфер получения для доставки пользовательскому приложению. Но вполне возможна ситуация, при которой определенный сегмент поступит не в порядке отправления. В этом случае протокол TCP на принимающей стороне может либо принимать только те сегменты, которые поступают в порядке отправления (другие сегменты просто отбрасываются), либо принимать все сегменты, номера которых зафиксированы в окне приема (вне зависимости от порядка их поступления).

Первый вариант действий упрощает реализацию протокола, но при этом возникает дополнительная нагрузка на сеть, так как отправитель будет ожидать некоторое время, а затем повторно передаст отброшенные сегменты, которые хотя и были успешно получены, но затем были отброшены из-за некорректного порядка поступления. Более того, если один сегмент был потерян при передаче, то необходимо посылать повторно все последующие сегменты.

При втором варианте может быть снижено количество повторных передач, но при этом требуется более сложный алгоритм приема и более сложная схема сохранения данных для буферизации и отслеживания порядка поступления данных.

Протокол TCP поддерживает очередь сегментов, которые были посланы, но еще не были подтверждены. Спецификация протокола говорит, что он будет повторно передавать сегмент, если на него не было получено подтверждение в определенный период времени. Реализация протокола TCP может поддерживать три режима повторной передачи:

- **Только первый.** Поддерживается один таймер повторной передачи для всей очереди. Если получено подтверждение, из очереди повторной передачи удаляется первый сегмент и таймер сбрасывается. Если таймер срабатывает (подтверждение не получено за указанное время), сегмент из начала очереди передается повторно, а таймер сбрасывается.
- **Пакетный.** Поддерживается один таймер повторной передачи для всей очереди. Если получено подтверждение, из очереди повторной передачи удаляются все сегменты и таймер сбрасывается. Если таймер срабатывает (подтверждение не получено за указанное время), все сегменты из очереди передаются повторно, а таймер сбрасывается.
- **Индивидуальный.** Для каждого сегмента в очереди поддерживается отдельный таймер. Если на определенный сегмент получено подтверждение, из очереди повторной передачи удаляется соответствующий сегмент и обнуляется его таймер. Если какой-либо из таймеров срабатывает, то соответствующий сегмент передается повторно, а его таймер сбрасывается.

Режим «Только первый» повышает эффективность передачи трафика, так как только потерянные сегменты (или сегменты, чьи подтверждения были потеряны) передаются повторно. Но из-за того, что таймер для второго сегмента в очереди не устанавливается до тех пор, пока не подтвержден первый сегмент, могут возникать некоторые задержки. Индивидуальный режим решает эти проблемы, но его реализация более сложна. Пакетный режим также снижает вероятность длительных задержек, но может привести к ненужным повторным передачам.

Реальная эффективность той или иной политики повторной передачи зависит от политики приема сегментов на стороне получателя. Если получатель использует политику приема, в соответствии с которой принимаются только сегменты, следующие в порядке отправления, то он будет отбрасывать сегменты, полученные после потерянного сегмента. Такая схема работы хорошо подходит к пакетной политике повторной передачи. Если получатель принимает все сегменты несмотря на порядок их прибытия, то оптимальными являются режимы «Только первый» или «Индивидуальный».

При поступлении сегмента протокол TCP на принимающей стороне имеет два варианта действий для отправки подтверждения:

- **Немедленно.** Как только данные с определенным номером в последовательности приняты, немедленно передается пустой (без данных) сегмент, содержащий соответствующий номер подтверждения.
- **С накоплением.** Когда данные, посланные отправителем, успешно приняты, протокол отмечает необходимость их подтверждения. Флаг ACK может быть выставлен в сегменте с данными, который получатель отправит в ответ отправителю. Для избежания длительных задержек устанавливается таймер окна. Если таймер истекает до момента отсылки очередного сегмента с подтверждением, то посылается пустой сегмент, содержащий соответствующий номер подтверждения.

Первый вариант прост в реализации, позволяет протоколу TCP на отправляющей стороне полностью «контролировать ситуацию» и уменьшает число повторных передач. Однако такой подход приводит к тому, что по сети часто передаются подтверждения (пустые сегменты, используемые только для подтверждения). Следовательно, возрастает загрузка сети.

Рассмотрим, например, ситуацию, при которой протокол TCP на принимающей стороне получает сегмент и немедленно посылает на него подтверждение. А приложение на принимающей стороне вдруг расширяет окно приема, вызывая тем самым посылку еще одного пустого сегмента для предоставления дополнительного кредита отправляющей стороне. Очевидно, что накладные расходы здесь достаточно

велики. Поэтому обычно используется вариант работы с накоплением. Следует отметить, что политика с накоплением приводит к увеличению нагрузки на принимающую сторону и усложняет задачу отправителя по оценке времени обращения.

Таймер повторной передачи. Протокол TCP не формирует явных негативных подтверждений (ACK), то есть подтверждений, явно указывающих на произошедшие нарушения. Вместо этого протокол полагается исключительно на положительные подтверждения и на повторную передачу, которая должна происходить, если подтверждение не поступает в определенный интервал времени. Два события приводят к повторной передаче сегмента:

- Сегмент может быть испорчен при передаче, но, тем не менее, доставлен получателю. Контрольная сумма, включенная в сегмент, позволяет получателю обнаружить ошибку и отбросить такой сегмент.
- Сегмент просто не поступает.

И в том, и в другом случае отправитель не сразу узнает о том, что посылка сегмента была не успешной.

Если сегмент на принимающей стороне принимается с ошибками либо не принимается вовсе, то при этом не формируется и не отсылается ACK. В таком случае потребуется повторная передача этого сегмента. Для принятия решения о том, когда следует осуществлять повторную передачу, вводится таймер, работающий с каждым посланным сегментом. Если время таймера истекает до момента получения ACK для этого сегмента, отправитель должен выполнить повторную передачу. Важной особенностью протокола TCP является то, что время этого таймера можно регулировать. Это очень полезно, так как если время будет слишком малым, то часто будут осуществляться ненужные повторные передачи, снижающие эффективность использования полосы пропускания сети. Если время будет очень большим, протокол не сможет быстро адекватно реагировать на потерю сегмента. Время таймера следует выбрать чуть больше времени обращения RTT (Round Trip Time). Естественно, само времяRTT (задержка в сети) зависит от множества факторов, вносящих свой вклад даже при постоянной загрузке сети.

Существуют два способа задания времени таймера повторной передачи.

- **Фиксированный таймер.** В первом случае используется фиксированное значение таймера, которое определяется по статистическим данным, характерным для «нормального» поведения распределенной сети. Иными словами, определяется среднее значение RTT и таймер выставляется с небольшим превышением. Так как такая политика основывается на устоявшемся режиме работы сети, она не способна адекватно и гибко реагировать на резкие изменения условий в сети. Как уже отмечалось, если время таймера слишком велико, инерция протокола будет велика, а если это значение мало, то может сложиться ситуация, при которой перегрузка в сети приведет к повторным передачам, что еще больше увеличит перегрузку.
- **Адаптивный таймер.** При втором способе используется адаптивная схема задания таймера, которая также имеет достоинства и недостатки. Предположим, что протокол TCP постоянно отслеживает время получения подтверждений на посланные сегменты и устанавливает свой таймер, основываясь на наблюдаемой задержке. Это значение не может быть признано самым оптимальным во всех возможных случаях по следующим причинам:
  - посылка сегмента может не подтверждаться немедленно (напомним, что, как правило, используются совокупные подтверждения сразу нескольких сегментов);
  - если сегмент был послан повторно, отправитель не всегда может узнать, был ли полученный ACK послан на начальный сегмент или на посланный повторно;
  - условия в сети могут внезапно поменяться.

Следует отметить, что эта проблема не имеет удовлетворительного единственно правильного решения. Всегда будет существовать некоторая неуверенность в оптимальности установки таймера повторной передачи. Механизм вычисления таймера описан в документе RFC 793.

13. СОКЕТЫ

13.1 Основы сокетов

Развитие Internet потребовало разрешить проблему: «Как обеспечить сетевое взаимодействие различных компьютеров (Hard) и установленных на них различных ОС (Soft) ?»

Концептуальное решение проблемы:

1) Установить единство протоколов.
2) Установить единство интерфейсов для протоколов прикладного уровня, подключаемых к вычислительной сети.

Предложено ввести *Сокеты* (Sockets – точки коммуникационного доступа внутри области коммуникационной сети.

Сокет (Sockets)- функциональный элемент представительского уровня для двунаправленной связи, который может использоваться для взаимодействия с другим процессом на одной и той же машине или с процессом, запущенным на других машинах. Программы Интернета такие как Telnet, rlogin, FTP, talk , и World Wide Web используют сокеты.

Каждый сокет характеризуется:

- протоколом,
- локальным адресом (инициатора соединения),
- удаленным портом.

Например, можно получить WWW-страницу от сервера Web, используя программу Telnet , так как они обе используют сокеты для сетевого взаимодействия. Для открытия подключения с сервером WWW на www.codesourcery.com, необходимо использовать telnet www.codesourcery.com 80. Константа 80 определяет подключение к Web серверу. Если после того, как подключение будет установлено, передать команду get /, то Web серверу через сокеты будет отправлено сообщение, на которое он ответит, передав исходный текст домашней HTML страницы и затем закроет подключение.

Пример:

```
% telnet www.codesourcery.com 80
Trying 206.168.99.1...
Connected to merlin.codesourcery.com (206.168.99.1).
Escape character is '^]'.
GET /
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
```

Основы создания сокетов

При создании сокета, необходимо определить три параметра: стиль взаимодействия, пространство имен, и пртокол.

Стиль взаимодействия контролирует, как сокет обрабатывает передаваемые данные, и определяет количество партнеров взаимодействия. Через сокеты данные передаются блоками (пакетами). Стиль взаимодействия определяет, как эти пакеты будут обработаны и как они передаются от отправителя к получателю.

Стили соединения гарантируют доставку всех пакетов в том порядке, в каком они были отправлены. Если во время передачи пакеты были потеряны или доставлены в неправильном порядке, получатель автоматически отправляет запрос на их повторную передачу. Стиль соединения напоминает телефонный звонок: адреса отправителя и получателя фиксируются в начале соединения, при установке подключения.

Стили датаграм не гарантирует доставки и правильного порядка прибытия. Пакеты могут быть потеряны или переупорядочены в пути из-за сетевых ошибок. Каждый пакет должен быть помечен его адресатом, и нет гарантии, что он будет доставлен. Система гарантирует только "максимальные усилия", поэтому пакеты могут исчезнуть или прибыть в различном порядке. Сокет стиля датаграмы ведет себя сходно с почтой. Отправитель определяет адрес получателя для каждого индивидуального сообщения.

Пространство имен определяет, как записаны адреса сокета (socket addresses). Адрес сокета идентифицирует один конец подключения сокета. Например, адреса сокета в локальном пространстве имен являются обычными именами файлов. В пространстве имен Интернет адрес сокета состоит из Интернет адреса (IP адрес) главного компьютера, присоединенного к сети и номера порта, который идентифицирует сокет среди множества сокетов на том же главном компьютере.

Протокол определяет, как передаются данные. Существуют следующие виды протоколов: TCP/IP , первичные сетевые протоколы, используемые Интернетом; сетевой протокол AppleTalk ; локальный UNIX протокол взаимодействия. Не все комбинации стилей, пространств имен и протоколов поддерживаются.

Системные вызовы

Виды системных вызовов:

socket - создать сокет

closes - уничтожить сокет

connect - создать соединение между двумя сокетами

bind - привязать сокет к порту сервера

listen - настройка сокета для принятия подключений

accept - принять запрос на соединение и создать сокет для процесса взаимодействия

Сокеты представляются дескрипторами файлов.

Создание и уничтожение сокетов

С помощью функций socket и close создаются и уничтожаются сокеты. При создании сокета, необходимо определить три параметра сокета: пространство имен, стиль взаимодействия и протокол.

Для указания пространства имен используются константы, начинающиеся с PF_ (сокращение "семейство протокола"). Например, PF_LOCAL или PF_UNIX определяют локальное пространство имен, и PF_INET определяет Интернет пространство имен.

Второй параметр, *стиль взаимодействия*, представляет собой константу, начинающуюся с SOCK_ . SOCK_STREAM определяет стиль взаимодействия соединение, SOCK_DGRAM - стиль датаграмы.

Третий параметр, *протокол*, определяет механизм нижнего уровня для передачи и получения данных. Для каждой комбинации пространство имен - стиль взаимодействия существует свой протокол.

Для каждой пары существует лучший протокол, поэтому можно указать 0, что соответствует этому протоколу. Если команда socket выполнена успешно, в качестве результата возвращается дескриптор файла для сокета. С помощью команд read и write , можно читать и записывать данные в сокет.

Вызов connect

Для создания соединение между двумя сокетами, клиент вызывает connect , передавая адрес сокета сервера для подключения. Клиент - процесс, инициализирующий соединение, а сервер - процесс, ожидающий разрешения соединения. Клиент посылает запрос connect , чтобы инициализировать соединение между локальным сокетом и сокетом сервера, переданным в качестве второго параметра. В качестве третьего параметра передается длина, в байтах, адресной структуры, на которую указывает второй параметр.

Отправка данных

Любая техника записи в дескриптор файла, может использоваться при записи в сокет. Функция send , определенная для дескрипторов файлов сокета, аналогична функции write с несколькими дополнительными параметрами.

13.2 Серверы

Цикл жизни сервера состоит из:

- создания сокета,
- привязки сокета к адресу,
- вызова listen , разрешающего соединение с сокетом, вызова accept , принимающего входящие соединения,
- закрытия сокета.

Данные не читаются и не записываются непосредственно через сокет сервера; вместо этого, каждый раз когда программа принимает новое соединение, Linux создает отдельный сокет, используется при передаче данных по этому соединению. В этом разделе рассматриваются вызовы bind, listen и accept .

С помощью команды bind адрес сервера должен быть привязан к сокету. Первый параметр команды - дескриптор файла сокета. Второй параметр - указатель на структуру адреса сервера; формат которого зависит от семейства адреса. Третий параметр - длина структуры адреса, в байтах.

Когда адрес связан с сокетом стиля соединение, необходимо вызвать listen , чтобы указать, что это - сервер. Первый параметр команды - дескриптор файла сокета. Второй параметр определяет, длину очереди ожидающих соединений. Если очередь заполнена, дополнительные соединения будут отвергнуты. Это не ограничивает общее количество соединений, которые сервер может обработать; это ограничивает только число клиентов, пытающихся соединиться и не получивших подтверждение.

С помощью команды accept сервер принимает запрос на соединение от клиента. Первый параметр вызова - дескриптор файла сокета. Второй параметр указывает на структуру адреса сокета, в которой хранится адрес клиентского сокета. Третий параметр - длина, в байтах, структуры адреса сокета. Сервер может использовать адрес клиента, чтобы определить, требуется ли действительно взаимодействовать с клиентом.

Вызов accept создает новый сокет для взаимодействия с клиентом и возвращает соответствующий дескриптор файла. Оригинальный сокет сервера продолжает принимать новые клиентские соединения.

Для чтения данных из сокета, без удаления их из входной очереди, используется команда recv . В качестве параметров передаются теже аргументы, что и в команде read , плюс дополнительный параметр FLAGS . Флаг MSG_PEEK указывает, что данные должны быть прочитаны, но не удалены из входной очереди.

13.3 Локальные сокеты

Сокеты, подключающие процессы на одном компьютере могут использовать локальное пространство имен, представляющий собой синоним для PF_LOCAL и PF_UNIX . Они называются локальными

сокетами или сокетами UNIX-домена. Адреса этих сокетов, определяемые именами файлов, используются только при создании соединения.

Имя сокета указывается в структуре sockaddr_un . Если в AF_LOCAL установлено поле sun_family , это указывает на то, что адрс в докальном пространстве имен. Поле Sun_path указывает, что используется имя файла; максимальная длина поля - 108 байт. Для вычисления длины struct sockaddr_un используется макрокоманда SUN_LEN . Может использоваться любое имя файла, но для процесса должно быть установлено право на запись в каталог. Чтоб соединениться с сокетом, процесса должен иметь право на чтения файла. Хотя различные компьютеры могут совместно использовать одну файловую систему, только процессы, запущенные на этом компьютере, могут взаимодействовать используя сокеты локального пространства имен.

Единственный допустимый протокол для локального пространства имен - 0. Поскольку он находится в файловой системе, локальный сокет представлен как файл.

Например, обратите внимание на начальную s:
% ls -l /tmp/socket
srwxrwx--x 1 user group 0 Nov 13 19:18 /tmp/socket

Вызов unlink удаляет локальный сокет, при завершении работы с ним.
Пример использования локальных сокетов

В листинге 13.1 представлена программа сервера, в которой создается локальный сокет и слушает запросы на соединения с сервером. При получении запроса на соединение, сервер читает текстовые сообщения, передаваемые через соединение и печатает их. Если одно из этих сообщений - "выход", программа сервера удаляет сокет и завершается. Программа socket-server предполагает, что путь к сокету передается через параметр командной строки.

Листинг 13.1 (socket-server.c)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <unistd.h>
/*      Чтение текста из сокета и вывод его на печать. Продолжение цикла до закрытия сокета.
 *      В качестве результата возвратит не ноль, если клиент передал сообщение "quit", иначе 0 */
int server (int client_socket)
{
        while (1) {
                int length;
                char* text;
                   /* Сначала из сокета прочитать длину текстого сообщения. Если в качестве результата возвратиться 0,
                     то клиент закрыл соединение */
                if (read (client_socket, &ength, sizeof (length)) == 0)
                        return 0;
```

```
            /* выделить место в буфере для хранения текста */
            text = (char*) malloc (length);
            /* Чтение текста и распечатка */
            read (client_socket, text, length);
            printf ("%s\n", text);
            /* Освободить буфер */
            free (text);
            /* Если от клиента поступило сообщение "quit", завершить работу */
            if (!strcmp (text, "quit"))
                return 1;
        }
    }
    int main (int argc, char* const argv[])
    {
        const char* const socket_name = argv[1];
        int socket_fd;
        struct sockaddr_un name;
        int client_sent_quit_message;
        /* Создать сокет */
        socket_fd = socket (PF_LOCAL, SOCK_STREAM, 0);
        /* Определить что это сервер */
        name.sun_family = AF_LOCAL;
        strcpy (name.sun_path, socket_name);
        bind (socket_fd, &name, SUN_LEN (&name));
        /* Слушать (ожидать) соединения */
        listen (socket_fd, 5);
        /* Неоднократно принимать соединения, создавая для каждого клиента server().
           Продолжать до получения от клиента сообщения "quit" */
        do {
            struct sockaddr_un client_name;
            socklen_t client_name_len;
            int client_socket_fd;
            /* Принимать соединение */
            client_socket_fd = accept (socket_fd, &client_name, &client_name_len);
            client_sent_quit_message = server (client_socket_fd);
            /* Закрыть соединение с нашей стороны */
            close (client_socket_fd);
        }
        while (!client_sent_quit_message);
        /* Удалить файл сокета */
        close (socket_fd);
        unlink (socket_name);
        return 0;
    }
```

Клиент-программа, представленная в листинге 13.2, соединяется с локальным сокетом и посылает сообщения. Путь к сокету и сообщения передатся через командную строку.

Листинг13.2 (socket-client.c)

```
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <unistd.h>
/* Записать TEXT в сокет, переданный дескриптором файла SOCKET_FD */
void write_text (int socket_fd, const char* text)
{
        /* Записать в строку количество байт, включая NUL-терминал */
        int length = strlen (text) + 1;
        write (socket_fd, &length, sizeof (length));
        /* Записать строку */
        write (socket_fd, text, length);
}
int main (int argc, char* const argv[])
{
        const char* const socket_name = argv[1];
        const char* const message = argv[2];
        int socket_fd;
        struct sockaddr_un name;
        /* Создать сокет */
        socket_fd = socket (PF_LOCAL, SOCK_STREAM, 0);
        /* Сохранить имя сервера в адресе сокета */
        name.sun_family = AF_LOCAL;
        strcpy (name.sun_path, socket_name);
        /* Соединиться с сокетом */
        connect (socket_fd, &name, SUN_LEN (&name));
        /* Записать текст командной строки в сокет */
        write_text (socket_fd, message);
        close (socket_fd);
        return 0;
}
```

Перед передачей сообщения, посылается размер сообщения в байтах в качестве переменной length. Сервер сохраняет размер сообщения, для выделения памяти под сообщение. Чтобы выполнить этот пример, необходимо запустить сервер-программу в одном окне, определить путь к сокету.

Например, /tmp/socket .

```
% ./socket-server /tmp/socket
```

В другом окне запустить клиент-программу несколько раз, опеределяя один и тот же путь сокета и посылая клиенту сообщение:

```
% ./socket-client /tmp/socket "Hello, world."
% ./socket-client /tmp/socket "This is a test."
```

Сервер-программа получает и печатает эти сообщения. Для закрытия соединения, клиент посылает сообщение "quit":

```
% ./socket-client /tmp/socket "quit"
```

Сервер-программа завершена.

13.4 Internet-Domain сокеты

Сокеты UNIX-domain использоваться только для взаимодействия двух процессов только на одном компьютере. Сокеты Internet, используются для соединения нескольких процессов на различных машинах, подключенных к сети.

Для соединения процессов через Интернет сокеты используют пространство имен Интернет указываемое с помощью PF_INET . Большинство протоколов являются TCP/IP . Интернет протокол (IP), протокол нижнего уровня, отправляет пакеты через Интернет, разбивая на меньшие пакеты, в случае необходимости. Он гарантирует только доставку "лучшего усилия", так что пакеты могут быть потеряны или переупорядочены во время транспортировки. Каждый компьютер имеет IP адрес. Протокол управления передачей (TCP), который следует за IP протоколом, обеспечивает надежное подключение. Это позволяет установить между компьютерами соединение, наподобие телефонного и гарантирует доставку данных в правильном порядке.

Интернет адрес сокета состоит из двух частей: номера компьютера и номера порта. Эта информация хранится в переменной структуры sockaddr_in . Для идентификации того, что это адрес Интернет пространства имен, необходимо установить поле sin_family в AF_INET . В поле Sin_addr хранится Интернет адрес компьютера, как 32-разрядное целое число IP . Каждому сокету на одном компьютере присваивается номер порта. Поскольку различные машины сохраняют многобайтовые значения в различном порядке байта, используют htons , чтобы преобразовать число порта к сетевому порядку байтов.

Команда gethostbyname преобразовывает удобочитаемые имена хоста, числа со стандартной точечной нотацией (типа 10.0.0.1) или DNS имена (такие как www.codesourcery.com) в 32-разрядные IP адреса. В качестве результата возвращается указатель на структуру struct hostent ; в поле h_addr хранится IP адрес главного компьютера.

Листинг 13.3 иллюстрирует использование Internet-domain сокетов. Программа получает домашнюю страницу от Web сервера, имя хоста которого определено в командной строке.

Листинг 13.3 (socket-inet.c)

```
#include <stdlib.h>
#include <stdio.h>
#include <netinet/in.h>
#include <netdb.h>
#include <sys/socket.h>
#include <unistd.h>
#include <string.h>
/* Печать содержимого домашней страницы.
 * В качестве результата передать флаг успешного завершения процесса.*/
void get_home_page (int socket_fd)
```

```
{
        char buffer[10000];
        ssize_t number_characters_read;
        /* Передать команду HTTP GET для домашней страницы */
        sprintf (buffer, "GET /\n");
        write (socket_fd, buffer, strlen (buffer));
        /* Читать данные из сокета. Не все данные могут быть возвращены одновременно,
         * продолжать попытку до завершения процесса */
        while (1) {
                number_characters_read = read (socket_fd, buffer, 10000);
                if (number_characters_read == 0)
                        return;
                /* Записать данные в стандартный вывод */
                fwrite (buffer, sizeof (char), number_characters_read, stdout);
        }
}
int main (int argc, char* const argv[])
{
        int socket_fd;
        struct sockaddr_in name;
        struct hostent* hostinfo;
        /* Создать сокет */
        socket_fd = socket (PF_INET, SOCK_STREAM, 0);
        /* Сохранить адрес сервера в адрессе сокета */
        name.sin_family = AF_INET;
        /* Преобразовать строку в число */
        hostinfo = gethostbyname (argv[1]);
        if (hostinfo == NULL)
                return 1;
        else
                name.sin_addr = *((struct in_addr *) hostinfo->h_addr);
        /* Web сервер использует 80 порт */
        name.sin_port = htons (80);
        /* Установить соединение с Web сервером */
        if (connect (socket_fd, &name, sizeof (struct sockaddr_in)) == -1) {
                perror ("connect");
                return 1;
        }
        /* Получить домашнюю страницу */
        get_home_page (socket_fd);
        return 0;
}
```

Имя хоста Web сервера задается в командно строке (без "http: //"). Команда gethostbyname преобразовывает имя хоста в числовой IP адрес и затем подключает поток (TCP) сокета к порту 80 на главном компьютере. Серверы используют Гипертекстовый Транспортный Протокол

(HTTP), поэтому передается команда HTTP GET , сервер в качестве ответа передает текст домашней страницы.

Для отображения страницы www.codesourcery.com, необходимо задать следующую команду

```
% ./socket-inet www.codesourcery.com
<html>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
...
```

13.5 Пары сокетов

Как упоминалось ранее, функция pipe , создает два дескриптора файлов для начала и конца канала. Каналы ограничены, потому что дескрипторы файлов используются только связанными процессами и потому что взаимодействие однонаправлено. Функция socketpair создает два дескриптора файлов для двух сокетов, подключенных на одном компьютере. Эти дескрипторы файлов разрешают двухстороннее взаимодействие двух связанных процессов. Первые три параметра команды - идентичны параметрам команды socket : они определяют домен, стиль подключения и протокол. Последний параметр - массив с двумя целыми числами, в котором хранятся характеристики файлов этих двух сокетов. При использовании команды socketpair , необходимо определить PF_LOCAL как пространство имен.

14 WWW

14.1. Концепция Word Weide Web (Web или WWW) .

К 1989 году гипертекст представлял новую, многообещающую технологию, которая имела относительно большое число реализаций с одной стороны, а с другой стороны делались попытки построить формальные модели гипертекстовых систем, которые носили скорее описательный характер и были навеяны успехом реляционного подхода описания данных.

Идея Т. Бернерс-Ли заключалась в том, чтобы применить гипертекстовую модель к информационным ресурсам, распределенным в сети, и сделать это максимально простым способом. Он заложил три краеугольных камня системы из четырех существующих ныне, разработав:

- язык гипертекстовой разметки документов HTML (HyperText Markup Lan-guage);
- универсальный способ адресации ресурсов в сети URL (Universal Resource Locator);
- протокол обмена гипертекстовой информацией HTTP (HyperText Transfer Protocol).

Позже команда NCSA добавила к этим трем компонентам четвертый:

- универсальный интерфейс шлюзов CGI (Common Gateway Interface).

Идея HTML -- пример чрезвычайно удачного решения проблемы построения гипертекстовой системы при помощи специального средства управления отображением. На разработку языка гипертекстовой разметки существенное влияние оказали два фактора: исследования в области интерфейсов гипертекстовых систем и желание обеспечить простой и быстрый способ создания гипертекстовой базы данных, распределенной на сети.

14.2 Гипертексты

В 1989 году активно обсуждалась проблема интерфейса гипертекстовых систем, т.е. способов отображения гипертекстовой информации и навигации в гипертекстовой сети. Значение гипертекстовой технологии сравнивали со значением книгопечатания. Утверждалось, что лист бумаги и компьютерные средства отображения/воспроизведения серьезно отличаются друг от друга, и поэтому форма представления информации тоже должна отличаться. Наиболее эффективной формой организации гипертекста были признаны контекстные гипертекстовые ссылки, а кроме того было признано деление на ссылки, ассоциированные со всем документом в целом и отдельными его частями.

Самым простым способом создания любого документа является его набивка в текстовом редакторе. Опыт создания хорошо размеченных для последующего отображения документов в CERN€е был - трудно найти физика, который не пользовался бы системой TeX или LaTeX. Кроме того к тому времени существовал стандарт языка разметки -- Standard Generalised Markup Language (SGML).

Следует также принять во внимание, что согласно своим предложениям Т. Бернерс-Ли предполагал объединить в единую систему имеющиеся информационные ресурсы CERN, и первыми демонстрационными системами должны были стать системы для NeXT и VAX/VMS.

Word Weide Web (Web или WWW) предоставляет легкий в управлении графический интерфейс Internet. Эти документы, а также ссылки между ними образуют информационную «паутину» (Web).

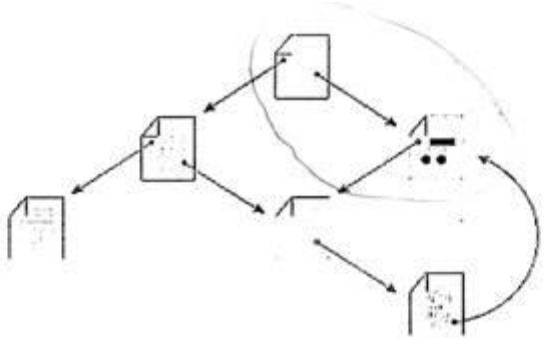


Рис. 14.1 Схема организации Web

Web предоставляет ссылки одной страницы на другие страницы (рис. 14.1). Web можно представить в виде большой библиотеки. Узлы Web подобны книгам, а «страницы» Web подобны страницам этих книг. Страницы могут содержать новости, рисунки, кинофильмы, страницы звукозаписи, объемные миры – все что угодно. Эти страницы могут размещаться на компьютерах в любой части света. При подключении к Web пользователь получает равный доступ к сведениям, распределенным по всему миру.

Обычно гипертекстовые системы имеют специальные программные средства построения гипертекстовых связей. Сами гипертекстовые ссылки хранятся в специальных форматах или даже составляют специальные файлы. Такой подход хорош для локальной системы, но не для распределенной на множестве различных компьютерных платформ. В HTML гипертекстовые ссылки встроены в тело документа и хранятся как его часть. Часто в системах применяют специальные форматы хранения данных для повышения эффективности доступа. В WWW документы -- это обычные ASCII- файлы, которые можно подготовить в любом текстовом редакторе. Таким образом, проблема создания гипертекстовой базы данных была решена чрезвычайно просто.

В качестве базы для разработки языка гипертекстовой разметки был выбран SGML (Standard Generalised Markup Language). Следуя академическим традициям, Бернерс-Ли описал HTML в терминах SGML (как описывают язык программирования в терминах формы Бекуса-Наура). Естественно, что в HTML были реализованы все разметки, связанные с выделением параграфов, шрифтов, стилей и т. п., т.к. реализация для NeXT подразумевала графический интерфейс. Важным компонентом языка стало описание

встроенных и ассоциированных гипертекстовых ссылок, встроенной графики и обеспечение возможности поиска по ключевым словам.

14.3 HTML

С момента разработки первой версии языка (HTML 1.0) произошло довольно серьезное развитие языка. Почти вдвое увеличилось число элементов разметки, оформление документов все больше приближается к оформлению качественных печатных изданий, развиваются средства описания не текстовых информационных ресурсов и способы взаимодействия с прикладным программным обеспечением. Совершенствуется механизм разработки типовых стилей. Фактически, в настоящее время HTML развивается в сторону создания стандартного языка разработки интерфейсов как локальных, так и распределенных систем.

Язык HTML

HyperText Markup Language (HTML) является стандартным языком, предназначенным для создания гипертекстовых документов в среде WEB. Такие документы могут просматриваться различными типами WEB-броузеров. Использование HTML позволяет форматировать документы для их представления с использованием шрифтов, линий и других графических элементов на любой системе, их просматривающей.

Большинство документов имеют стандартные элементы, такие, как заголовок, параграфы или списки. Используя тэги HTML вы можете обозначать данные элементы, обеспечивая WEB-браузеры минимальной информацией для отображения данных элементов, сохраняя в целом общую структуру и информационную полноту документов.

Основное преимущество HTML заключается в том, что ваш документ может быть просмотрен на WEB-браузерах различных типов и на различных платформах.

HTML-документы могут быть созданы при помощи любого текстового редактора или специализированных HTML-редакторов и конвертеров. Выбор редактора зависит исключительно от понятия удобства и личных пристрастий каждого автора. Например, HTML редакторы, такие, как "Netscape Navigator Gold" компании Netscape позволяют создавать документы графически с использованием технологии WYSIWYG (What You See Is What You Get). С другой стороны, большинство традиционных средств для создания документов имеют конвертеры, позволяющие преобразовывать документы к формату HTML.

Основные положения

Все тэги HTML начинаются с "<" (левой угловой скобки) и заканчиваются символом ">" (правой угловой скобки). Как правило, существует стартовый тэг и завершающий тэг. Завершающий тэг выглядит так же, как стартовый, и отличается от него прямым слешем перед текстом внутри угловых скобок.

Пример тэга заголовка, определяющий текст, находящийся внутри стартового и завершающего тэга и описывающий заголовок документа:

<TITLE> Заголовок документа </TITLE>

Некоторые тэги, такие, как <P> (тэг, определяющий абзац), не требуют завершающего тэга, но его использование придает исходному тексту документа улучшенную читаемость и структурируемость. HTML не реагирует на регистр символов, описывающих тэг.

Дополнительные пробелы, символы табуляции и возврата каретки, добавленные в исходный текст HTML-документа для его лучшей читаемости, будут проигнорированы WEB-броузером при интерпретации документа, если они не помещены внутрь тэгов <PRE> и </PRE>.

Структура документа

HTML-документ представляет собой тэговую модель, то есть совокупность элементов, каждый из которых окружён тэгами.

По своему значению тэги близки к понятию скобок «begin/end» в универсальных языках программирования,

Тэги определяют области действия правил интерпретации текстовых элементов документа.

В своём наиболее общем виде структура документа HTML выглядит следующим образом:

```
< HTML>
Содержание документа
</HTML>
```

Команда <HTML> должна быть первой в документе. Она всегда используется в паре с </HTML>, завершающей документ.

Между этими двумя командами располагается текст страницы и другие команды.

Сам элемент HTML состоит из двух частей: *заголовка (HEAD) и тела документа (BODY):*

```
<HTML>
<HEAD>
Содержание заголовка
</HEAD>
<BODY>
Содержание тела документа
</BODY>
</HTML>
```

Тэг заголовочной части документа должен быть использован сразу после тэга <HTML> и более нигде в теле документа Стартовый тэг <HEAD> помещается непосредственно перед тэгом <TITLE> и другими тэгами, описывающими документ, а завершающий тэг </HEAD> размещается сразу после окончания описания документа.

Большинство WEB-броузеров отображают содержимое тэга <TITLE> в заголовке окна, содержащего документ и в файле закладок, если он поддерживается WEB-броузером. Заголовок, ограниченный тэгами <TITLE> и </TITLE>, размещается внутри <HEAD>-тэгов.

HTML позволяет вставлять в тело документа комментарии, которые сохраняются при передаче документа по сети, но не отображаются броузером. **<!--** Это комментарий **-->**

Комментарии могут встречаться в документе где угодно и в любом количестве.

Тэги тела документа идентифицируют отображаемые в окне компоненты HTML-документа. Тело документа может содержать ссылки на другие документы, текст и другую форматированную информацию. Тело документа должно находиться между тэгами <BODY> и </BODY>. Это та часть документа, которая отображается как текстовая и графическая (смысловая) информация вашего документа.

Например:

```
<HTML>
    <HEAD>
        <TITLE> Список сотрудников </TITLE>
    </HEAD>
    <!-- Это комментарий -->
        <BODY> тело документа </BODY>
</HTML>
```

<Hx>, <P>, <CENTER>, <PRE>,

Когда пишется HTML-документ, текст структурно делится на просто текст, заголовки частей текста, заголовки более высокого уровня и т.д. Первый уровень заголовков (самый большой) обозначается цифрой 1, следующий - 2, и т.д. Большинство броузеров поддерживает интерпретацию шести уровней заголовков, определяя каждому из них собственный стиль. Стиль верхнего уровня имеет признак "1".

<H1> Заголовок первого уровня </H1>

В отличие от большинства текстовых процессоров, в HTML-документе обычно игнорируются символы возврата каретки. Физический разрыв абзаца может находиться в любом месте исходного текста документа. Однако броузер разделяет абзацы только при наличии тэга **<P>**. Дополнительные параметры тэга <P>:

<P ALIGN=left|center|right> позволяют выравнивать абзац по левому краю, центру и правому краю.

Можно центрировать все элементы документа в окне браузера. Для этого можно использовать тэг **<CENTER>.** Все элементы между тэгами <CENTER> и </CENTER> будут находиться в центре окна.

Тэг преформатирования, **<PRE>**, позволяет представлять текст со специфическим форматированием на экране. Предварительно сформатированный текст заканчивается завершающим тэгом </PRE>. Внутри предварительно сформатированного текста разрешается использовать: перевод строки, символы табуляции (сдвиг на 8 символов вправо) и непропорциональный шрифт, устанавливаемый броузером

Использование тэгов, определяющих формат абзаца, таких как <Hx> или <ADDRESS>, будет игнорироваться броузером при помещении их между тэгами <PRE> и </PRE>.

Тэг **
** извещает броузер о разрыве строки.

Наилучший пример использования данного тэга - форматированный адрес или любая другая последовательность строк, где броузер должен отображать их одну под другой. Дополнительный параметр позволяет расширить возможности тэга
. <**BR CLEAR=left|right|all>**

Он позволяет разместить следующую строку, начиная с чистой левой (left), правой (right) или обоих (all) границ окна броузера. Например, если рядом с текстом слева встречается рисунок, то можно использовать тэг
 для смещения текста ниже рисунка:

Если вы не хотите, чтобы броузер автоматически переносил строку, то вы можете обозначить ее тэгами <**NOBR> и </NOBR>**. В этом случае броузер не будет переносить строку даже если она выходит за границы экрана; вместо этого броузер позволит горизонтально прокручивать окно.

<BLOCKQUOTE>, данный тэг предназначен для обозначения в документе цитаты из другого источника. Текст, обозначенный тэгом <BLOCKQUOTE>, отступает от левого края документа на 8 пробелов

<HR>, стили шрифтов, размер и цвет шрифта.

Используя тэг **<HR>** можно разделить текст горизонтальной чертой. Формат тэга: **<HR SIZE=number WIDTH=number|percent ALIGN=left|right|center NOSHADE>**

Параметры тэга:

SIZE Толщина линии в пикселях.

WIDTH Ширина линии в пикселях или процентах от ширины окна броузера.

ALIGN Расположение на экране (слева | по центру | справа).

NOSHADE По умолчанию линия представлена в 3D виде с тенью. NOSHADE позволяет представить линию просто однотонной темной полоской.

HTML позволяет использовать различные стили шрифтов для выделения текстовой информации в ваших документах. Вы можете комбинировать различные виды стилей.

Стиль	Элемент или тэг	Результат
Bold	<B> Этот текст жирный </B>	**Этот текст жирный**
Italic	<I>Этот текст наклонный </I>	*Этот текст наклонный*
Mono spaced	<TT> Текст с непроп. шрифтом <TT>	Текст с непроп. шрифтом

Комбинирование стилей позволяет вам отображать в одной строке несколько элементов различными стилями. Дополнительные стили:

big (большой) <BIG> большой </BIG>

small (маленький) <SMALL> маленький </SMALL>

sub (подстрочник) _{подстрочник}

sup (надстрочник) ^{надстрочник}

Можно изменять размер шрифта при помощи тэга: **<FONT SIZE=+|- n>**

Шрифт может иметь размер от 1 до 7. Вы можете прямо указать размер шрифта цифрой, или указать смещение относительно базового значения (по умолчанию - 3) в положительную или отрицательную сторону. Базовое значение можно изменить при помощи тэга: <**BASEFONT SIZE=n>** .

Можно изменить цвет шрифта при помощи тэга: **<FONT COLOR="#xxxxxx>.**

Цвет указывается в RGB-формате (Red-Green-Blue) посредством указания размерности каждой компоненты цвета в шестнадцатеричном формате. Например, белый цвет обозначается "000000", черный - "FFFFFF", синий - "0000FF" и т.п.

<FONT COLOR="#FF0000"> Красный </FONT>

<FONT COLOR="#00FF00"> Зеленый </FONT>

<FONT COLOR="#0000FF"> Синий </FONT>

Специальные тэги HTML

Тэг <ADDRESS> используется для выделения автора документа и его адреса (например, e-mail). Синтаксис: **<ADDRESS> Адрес-автора </ADDRESS>**

Escape-последовательности.

Некоторые символы являются управляющими символами в HTML и не могут напрямую использоваться в документе: левая угловая скобка "<" правая угловая скобка ">" амперсант "&" двойные кавычки """

Чтобы использовать данные символы в документе, необходимо заменить их escape-последовательностями: < < > > & & " "

Список базовых тэгов HTML

Стартовый	Завершающий	Описание
<HTML>	</HTML>	Обозначение HTML-документа
<HEAD>	</HEAD>	Заголовочная часть документа
<TITLE>	</TITLE>	Заголовок документа
<BODY>	</BODY>	Тело документа
<H1>	</H1>	Заголовок абзаца первого уровня
<H2>	</H2>	Заголовок абзаца второго уровня
<H3>	</H3>	Заголовок абзаца третьего уровня
<H4>	</H4>	Заголовок абзаца четвертого уровня
<H5>	</H5>	Заголовок абзаца пятого уровня
<H6>	</H6>	Заголовок абзаца шестого уровня
<P>	</P>	Абзац
<PRE>	</PRE>	Форматированный текст
 		Перевод строки без конца абзаца
<BLOCKQUOTE>	</BLOCKQUOTE>	Цитата
<HR>		Горизонтальная черта
<B>	</B>	Жирный шрифт
<I>	</I>	Наклонный шрифт

<TT>	<TT>	Непроп. шрифт
<FONT SIZE=+|- n>		Размер шрифта
<FONT COLOR="#xxxxxx>.		Цвет шрифта
<! >		Комментарий

14.4 WWW-архитектура.

Архитектура WWW-технологии

От описания основных компонентов перейдем к архитектуре взаимодействия программного обеспечения в системе World Wide Web. WWW построена по хорошо известной схеме "клиент-сервер". На рисунке 14.2 показано, как разделены функции в этой схеме.

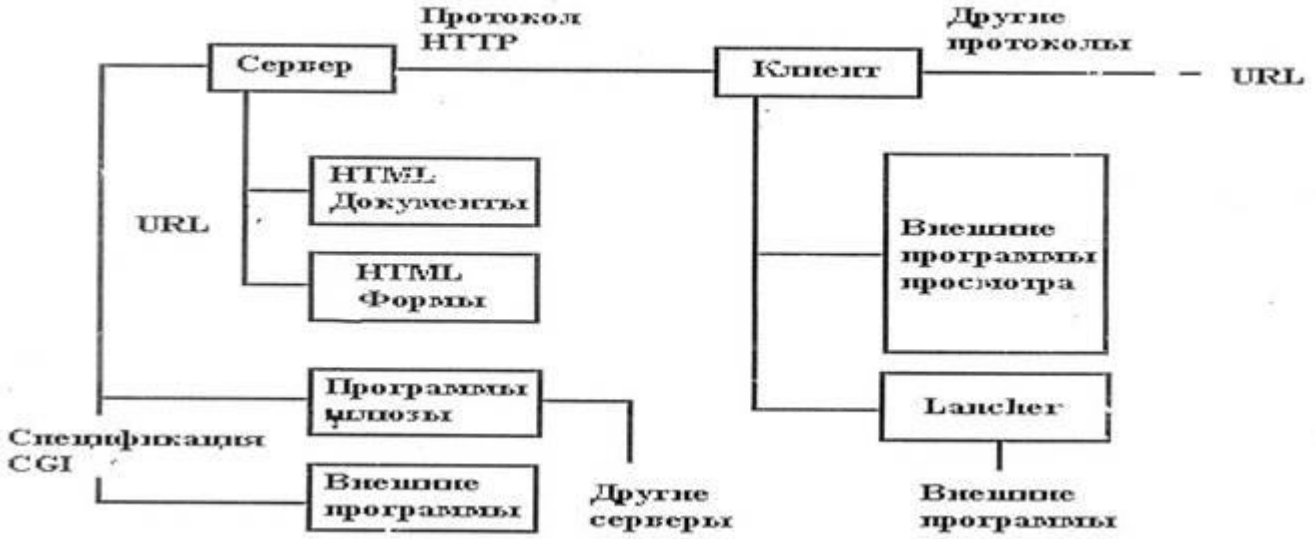


Рис. 14.2 Архитектура WWW- технологии

Программа-клиент выполняет функции интерфейса пользователя и обеспечивает доступ практически ко всем информационным ресурсам Internet. В этом смысле она выходит за обычные рамки работы клиента только с сервером определенного протокола, как это происходит в telnet, например. Отчасти, довольно широко распространенное мнение, что Mosaic или Netscape, которые безусловно являются WWW-клиентами, это просто графический интерфейс в Internet, является отчасти верным.

Однако, как уже было отмечено, базовые компоненты WWW-технологии (HTML и URL) играют при доступе к другим ресурсам Mosaic не последнюю роль, и поэтому мультипротокольные клиенты должны быть отнесены именно к World Wide Web, а не к другим информационным технологиям Internet.

Фактически, клиент -- это интерпретатор HTML. И как типичный интерпретатор, клиент в зависимости от команд (разметки) выполняет различные функции. В круг этих функций входит не только размещение текста на экране, но обмен информацией с сервером по мере анализа полученного HTML-текста, что наиболее наглядно происходит при отображении встроенных в текст графических образов.

При анализе URL-спецификации или по командам сервера клиент запускает дополнительные внешние программы для работы с документами в форматах, отличных от HTML, например GIF, JPEG, MPEG, Postscript и т. п.

Для запуска клиентом программ независимо от типа документа была разработана программа запуска (Luncher), но в последнее время гораздо большее распространение получил механизм согласования запускаемых программ через MIME-типы. Другую часть программного комплекса WWW составляет

сервер протокола HTTP, базы данных документов в формате HTML, управляемые сервером, и программное обеспечение, разработанное в стандарте спецификации CGI.

В настоящее время число базовых HTTP серверов расширилось. Появился очень неплохой сервер для MS-Windows и Apachie-сервер для Unix-платформ.

Сервер для Windows -- это shareware, но без встроенного самоликвидатора, как в Netscape. Учитывая распространенность персоналок в нашей стране, такое программное обеспечение дает возможность попробовать, что такое WWW.

Разработанный Apachie - свободно распространяемый (freeware) и реализует новые дополнения к протоколу HTTP, связанные с защитой от несанкционированного доступа, которые предложены группой по разработке этого протокола и реализуются практически во всех коммерческих серверах.

База данных HTML-документов -- это часть файловой системы, которая содержит текстовые файлы в формате HTML и связанные с ними графику и другие ресурсы. Особое внимание хотелось бы обратить на документы, содержащие элементы экранных форм. Эти документы реально обеспечивают доступ к внешнему программному обеспечению.

Прикладное программное обеспечение, работающее с сервером, можно разделить на программы-шлюзы и прочие. Шлюзы -- это программы, обеспечивающие взаимодействие сервера с серверами других протоколов, например ftp, или с распределенными на сети серверами Oracle. Прочие программы -- это программы, принимающие данные от сервера и выполняющие какие-либо действия: получение текущей даты, реализацию графических ссылок, доступ к локальным базам данных или просто расчеты.

Спецификация Common Gateway Interface (CGI) была специально разработана группой NCSA для расширения возможностей WWW за счет подключения всевозможного внешнего программного обеспечения.

Такой подход логично продолжал принцип публичности и простоты разработки и наращивания возможностей WWW. Если команда CERN предложила простой и быстрый способ разработки баз данных, то NCSA развила этот принцип на разработку программных средств.

Следует заметить, что в общедоступной библиотеке CERN были модули, позволяющие программистам подключать свои программы к серверу HTTP, но это требовало использования этой библиотеки. Предложенный и описанный в CGI способ подключения не требовал дополнительных библиотек и буквально ошеломлял своей простотой. Сервер взаимодействовал с программами через стандартные потоки ввода/вывода, что упрощает программирование до предела.

При реализации CGI чрезвычайно важное место заняли методы доступа, описанные в HTTP. И хотя реально используются только два из них (GET и POST), опыт развития HTML показывает, что сообщество WWW ждет развития и CGI по мере усложнения задач, в которых будет использоваться WWW-технология.

Завершая обсуждение архитектуры World Wide Web хотелось бы еще раз подчеркнуть, что ее компоненты существуют практически для всех типов компьютерных платформ и свободно доступны в сети. Любой, кто имеет доступ в Internet, может создать свой WWW-сервер, или, по крайней мере, посмотреть информацию с других серверов.

14.5 URL (Universal Resource Locator).

Одним из основный понятий концепции WWW стала универсальная форма адресации информационных ресурсов. Universal Resource Identification (URI) представляет собой довольно стройную систему, учитывающую опыт адресации и идентификации e-mail, Gopher, WAIS, telnet, ftp и т. п.

Однако, реально из всего, что описано в URI, для организации баз данных в WWW требуется только Universal Resource Locator (URL). Без наличия этой спецификации вся мощь HTML оказалась бы бесполезной. URL используется в гипертекстовых ссылках и обеспечивает доступ к распределенным ресурсам сети.

В URL можно адресовать как другие гипертекстовые документы формата HTML, так и ресурсы e-mail, telnet, ftp, Gopher, WAIS, например. Различные интерфейсные программы по разному осуществляют доступ к этим ресурсам. Одни, как например Netscape, сами способны поддерживать взаимодействие по протоколам, отличным от протокола HTTP, базового для WWW, другие, вызывают для этой цели внешние программы.

Однако, даже в первом случае, базовой формой представления отображаемой информации является HTML, а ссылки на другие ресурсы имеют форму URL. Следует отметить, что программы обработки электронной почты в формате MIME также имеют возможность отображать документы, представленные в формате HTML. Для этой цели в MIME (Multiperposal Internet Mail Exchange) зарезервирован тип "text/html".

Необходимость в URI была понятна разработчикам WWW с момента зарождения системы, т.к. предполагалось объединение в единую информационную среду средств, использующих различные способы идентификации информационных ресурсов. Первоначально это были FTP архивы, информационно-поисковая система Alise и справочная система ЦЕРН. Однако Бернерс Ли подошел к делу основательно и разработал спецификацию, которая включала в себя обращения к FTP, Gopher, WAIS, Usenet, E-mail, Prospero, Telnet, Whois, X500 и конечно HTTP(WWW). В итоге была разработана универсальная спецификация, которая позволяет расширять список адресуемых ресурсов за счет появления новых.

Место применения URL -- гипертестовые ссылки, которые записываются в тагах < A HREF=URI > и < LINK HREF=URI >. Встраиваемые графические объекты также адресуются по спецификации URI в тагах < IMG SRC=URL > и <FIG SRC=URI >. Реализация URI для WWW называется URL(Uniform Resource Locator). Точнее, URI -- это реализация схемы URL, отображенная на алгоритм доступа к ресурсам по сетевым протоколам. Существует еще и URN (Uniform Resource Name), которое отображает URL в пространство имен на сети. Вообще говоря, на мой взгляд это уже перебор. Собственно, появление URN связано с желанием адресовать части почтового сообщения MIME. Но здесь есть момент, который находится в стадии дебатов. Сообщение "живет" не более 5 дней. Если оно сохранено, то его можно превратить в другой информационный ресурс, например WWW страницу. Поэтому судьба URN еще не решена.

При разработке URI преследовались следующие принципы:

- Расширяемость -- новые адресные схемы должны были легко вписываться в существующий синтаксис URI.
- Полнота -- по возможности, любая из существовавших схем должна была описываться посредством URI.
- Читаемость -- адрес должен был быть легко читаем человеком, что вообще характерно для технологии WWW -- документы вместе с ссылками могут разрабатываться в обычном текстовом редакторе.

Полнота и Читаемость порождали коллизию, связанную с тем, что в некоторых схемах используется двоичная информация. Эта проблема была решена за счет формы представления такой информации. Символы, которые несут служебные функции и двоичные данные отображаются в URI в шестнадцатеричном коде и предваряются символом "%".

Прежде, чем рассмотреть различные схемы представления адресов, приведем пример простого адреса URI:

http://polyn.net.kiae.su/polyn/index.html

Перед двоеточием стоит имя схемы адреса -- "http". Это имя отделено двоеточием от остатка URI, который называется путь. В данном случае путь состоит из доменного адреса машины, на которой установлен сервер HTTP и пути от корня дерева сервера к файлу "index.html".

Кроме представленной выше полной записи URI, существует упрощённая. Она предполагает, что к моменту ее использования многие параметры адреса ресурса уже определены (протокол, адрес машины в сети, некоторые элементы пути). При таких предположениях автор гипертекстовых страниц может указывать только относительный адрес ресурса, т.е. адрес относительный базовых определенных ресурсов.

Ссылки

Гипертекстовые ссылки являются ключевым компонентом, делающим WEB привлекательным для пользователей. Ссылки могут указывать на другой документ, специальное место данного документа или выполнять другие функции, например запрашивать файл по FTP-протоколу для отображения его браузером. URL может указывать на специальное место по абсолютному пути доступа, или указывать на документ в текущем пути доступа, что часто используется при организации больших структурированных WEB-сайтов. Вы можете использовать ссылки как для перемещения по документу, так и для перемещения от одного документа к другому.

Однако, HTML не поддерживает возврат на предыдущую ссылку, если перемещение происходило внутри документа.

HTML использует **URL (**Uniform Resource Locator) для представления гипертекстовых ссылок и ссылок на сетевые сервисы внутри HTML-документа. Первая часть URL (до двоеточия) описывает метод доступа или сетевой сервис. Другая часть URL (после двоеточия) интерпретируется в зависимости от метода доступа. Обычно, два прямых слэша после двоеточия обозначают имя машины**: method://machine-name/path/foo.html**

URL имеет следующий формат: method://servername:port/pathname#anchor

METHOD Имя операции, которая будет выполняться при интерпретации данного URL. Наиболее часто используемые методы:

file: чтение файла с локального диска. Данный метод используется для отображения какого-либо файла, находящегося на машине пользователя.

http: доступ к WEB-странице в сети с использованием HTTP-протокола.

ftp: запрос файла с анонимного FTP-сервера.

mailto: активизирует почтовую сессию с указанным пользователем и хостом.

telnet: обращение к службе telnet

news: вызов службы новостей, если броузер ее поддерживает.

SERVERNAME Необязательный параметр, описывающий полное сетевое имя машины.

PORT Номер порта TCP на котором функционирует WEB-сервер.

PATHNAME Частичный или полный путь к документу, который должен вызваться в результате интерпретации. Если после сетевого имени машины сразу идет имя документа, то он должен находиться в корневом каталоге на удаленной машине или (что чаще) в каталоге, выделенном WEB-сервером в качестве корневого. Если же URL заканчивается сетевым именем машины, то в качестве документа запрашивается документ из корневого каталога удаленной машины с именем, установленным в настройках WEB-сервера (как правило, это index.html).

#ANCHOR Данный элемент является ссылкой на строку (точку) внутри HTML-документа. Большинство броузеров, встречая после имени документа данный элемент, размещают документ на экране таким образом, что указанная строка документа помещается в верхнюю строку рабочего окна броузера. Точки, на которые ссылается #anchor, указываются в документе при помощи тэга NAME, как это будет описано далее. Для того, чтобы броузер отобразил ссылку на URL, необходимо отметить URL специальными тэгами в HTML-документе.

<A HREF="URL"> текст-который-будет-подсвечен-как-ссылка </A>

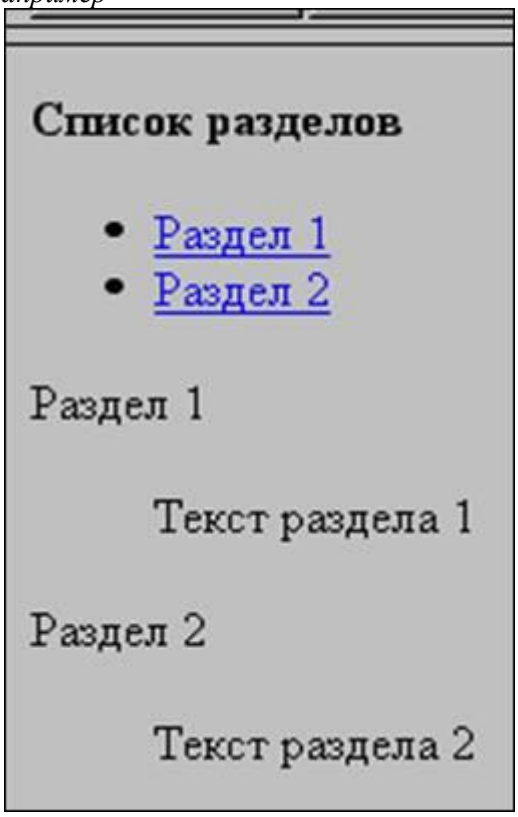
Тэг <A HREF="URL">открывает описание ссылки, а тэг </A> - закрывает его. Любой текст, находящийся между данными двумя тэгами подсвечивается специальным образом Web-браузером. Текст, обозначающий

URL, не отображается браузером, а используется только для выполнения предписанных им действий при активизации ссылки
Вот пример сегмента HTML-документа:

Для получения дополнительной информации смотри
<A HREF="http://nenwork-journal.mpei.ac.ru> страницу компании СофтСервис </A>
Данная строка будет выглядеть на экране следующим образом:
http://nenwork-journal.mpei.ac.ru

Ссылки на точки внутри документа.
Можно делать ссылки на различные участки или разделы одного и того же документа, используя специальных скрытый маркер для этих разделов. Для создания такой ссылки необходимо выполнить следующие шаги:
1. Создайте маркер раздела.
<A NAME="named_anchor"> Текст в первой строке броузера </A>
2. Создайте ссылку на данный маркер:
<A HREF="#named_anchor"> Текст </A>

Например




```
<HTML>
<HEAD>
  <TITLE>пример 2</TITLE>
</HEAD>
<BODY>
<p><b>Список разделов</b></p>
<ul> <li><a href="#ex1">Раздел 1</a></li>
<li><a href="#ex2">Раздел 2</a></li> </ul>
```

```
<p><a name="ex1"></a>Раздел 1</p>
<ul> <p>Текст раздела 1</p> </ul>
<p><a name="ex2"></a>Раздел 2</p>
<ul> <p>Текст раздела 2 <br></p>
</BODY>
</HTML>
```

а) **б)**

Рис. 14.3 Текст (а) и вид (б)отображаемой страницы примера

Маркер раздела может быть поставлен как в том же документе, который просматривается в текущий момент, так и в другом документе. Во втором случае браузер осуществит загрузку другого документа и перейдет к указанному для него разделу.

14.6 Протокол HTTP

HTTP (англ. HyperText Transfer Protocol — «протокол передачи гипертекста») — протокол прикладного уровня передачи данных (изначально — в виде гипертекстовых документов). Основой HTTP является технология «клиент-сервер», то есть предполагается существование потребителей (клиентов), которые инициируют соединение и посылают запрос, и поставщиков (серверов), которые ожидают соединения для получения запроса, производят необходимые действия и возвращают обратно сообщение с результатом.

HTTP в настоящее время повсеместно используется во Всемирной паутине для получения информации с веб-сайтов. В 2006 году в Северной Америке доля HTTP-трафика превысила долю P2P-сетей и составила 46 %, из которых почти половина — это передача потокового видео и звука[1].

К концептуальным понятиям относится также протокол обмена данными в World Wide Web -- HyperText Transfer Protocol (HTTP), который предназначен для обмена гипертекстовыми документами и учитывает специфику такого обмена. Так в процессе взаимодействия, клиент может получить новый адрес ресурса на сети (relocation), запросить встроенную графику, принять и передать параметры и т. п. Управление в HTTP реализовано в виде ASCII-команд. Реально разработчик гипертекстовой базы данных сталкивается с элементами протокола только при использовании внешних расчетных программ или при доступе к внешним относительно WWW информационным ресурсам, например базам данных.

HTTP используется также в качестве «транспорта» для других протоколов прикладного уровня, таких как SOAP, XML-RPC, WebDAV.

Основным объектом манипуляции в HTTP является ресурс, на который указывает URI (англ. Uniform Resource Identifier) в запросе клиента. Обычно такими ресурсами являются хранящиеся на сервере файлы, но ими могут быть логические объекты или что-то абстрактное. Особенностью протокола HTTP является возможность указать в запросе и ответе способ представления одного и того же ресурса по различным параметрам: формату, кодировке, языку и т. д. (В частности для этого используется HTTP-заголовок.) Именно благодаря возможности указания способа кодирования сообщения клиент и сервер могут обмениваться двоичными данными, хотя данный протокол является текстовым.

HTTP — протокол прикладного уровня, аналогичными ему являются FTP и SMTP. Обмен сообщениями идёт по обыкновенной схеме «запрос-ответ». Для идентификации ресурсов HTTP использует глобальные URI. В отличие от многих других протоколов, HTTP не сохраняет своего состояния. Это означает отсутствие сохранения промежуточного состояния между парами «запрос-ответ». Компоненты, использующие HTTP, могут самостоятельно осуществлять сохранение информации о состоянии, связанной с последними запросами и ответами (например, «куки» на стороне клиента, «сессии» на стороне сервера). Браузер, посылающий запросы, может отслеживать задержки ответов. Сервер может хранить IP-адреса и заголовки запросов последних клиентов. Однако сам протокол не осведомлён о предыдущих запросах и

ответах, в нём не предусмотрена внутренняя поддержка состояния, к нему не предъявляются такие требования.

Структура протокола

Каждое HTTP-сообщение состоит из трёх частей, которые передаются в указанном порядке:

- Стартовая строка (англ. Starting line) — определяет тип сообщения;
- Заголовки (англ. Headers) — характеризуют тело сообщения, параметры передачи и прочие сведения;
- Тело сообщения (англ. Message Body) — непосредственно данные сообщения. Обязательно должно отделяться от заголовков пустой строкой.

Заголовки и тело сообщения могут отсутствовать, но стартовая строка является обязательным элементом, так как указывает на тип запроса/ответа. Исключением является версия 0.9 протокола, у которой сообщение запроса содержит только стартовую строку, а сообщения ответа только тело сообщения.

Стартовая строка

Стартовые строки различаются для запроса и ответа. Строка запроса выглядит так:

GET URI — для версии протокола 0.9.

Метод URI HTTP/Версия — для остальных версий.

Здесь:

Метод (англ. Method) — название запроса, одно слово заглавными буквами. В версии HTTP 0.9 использовался только метод GET, список запросов для версии 1.1 представлен ниже.

URI определяет путь к запрашиваемому документу.

Версия (англ. Version) — пара разделённых точкой цифр. Например: 1.0

Стартовая строка ответа сервера имеет следующий формат: HTTP/Версия КодСостояния Пояснение, где:

Версия — пара разделённых точкой цифр как в запросе.

Код состояния (англ. Status Code) — три цифры. По коду состояния определяется дальнейшее содержимое сообщения и поведение клиента.

Пояснение (англ. Reason Phrase) — текстовое короткое пояснение к коду ответа для пользователя. Никак не влияет на сообщение и является необязательным.

Методы

Метод HTTP (англ. HTTP Method) — последовательность из любых символов, кроме управляющих и разделителей, указывающая на основную операцию над ресурсом. Обычно метод представляет собой короткое английское слово, записанное заглавными буквами. Обратите внимание, что название метода чувствительно к регистру.

Каждый сервер обязан поддерживать как минимум методы GET и HEAD. Если сервер не распознал указанный клиентом метод, то он должен вернуть статус 501 (Not Implemented). Если серверу метод известен, но он неприменим к конкретному ресурсу, то возвращается сообщение с кодом 405 (Method Not Allowed). В обоих случаях серверу следует включить в сообщение ответа заголовок Allow со списком поддерживаемых методов.

Кроме методов GET и HEAD, часто применяется метод POST.

OPTIONS

Используется для определения возможностей веб-сервера или параметров соединения для конкретного ресурса. В ответ серверу следует включить заголовок Allow со списком поддерживаемых методов. Также в заголовке ответа может включаться информация о поддерживаемых расширениях.

Предполагается, что запрос клиента может содержать тело сообщения для указания интересующих его сведений. Формат тела и порядок работы с ним в настоящий момент не определён. Сервер пока должен его игнорировать. Аналогичная ситуация и с телом в ответе сервера.

Для того, чтобы узнать возможности всего сервера, клиент должен указать в URI звёздочку — «*». Запросы «OPTIONS * HTTP/1.1» могут также применяться для проверки работоспособности сервера (аналогично «пингованию») и тестирования на предмет поддержки сервером протокола HTTP версии 1.1. Результат выполнения этого метода не кэшируется.

GET

Используется для запроса содержимого указанного ресурса. С помощью метода GET можно также начать какой-либо процесс. В этом случае в тело ответного сообщения следует включить информацию о ходе выполнения процесса.

Клиент может передавать параметры выполнения запроса в URI целевого ресурса после символа «?»:

GET /path/resource?param1=value1&param2=value2 HTTP/1.1

Согласно стандарту HTTP, запросы типа GET считаются идемпотентными.

Кроме обычного метода GET, различают ещё условный GET и частичный GET. Условные запросы GET содержат заголовки If-Modified-Since, If-Match, If-Range и подобные. Частичные GET содержат в запросе Range. Порядок выполнения подобных запросов определён стандартами отдельно.

HEAD

Аналогичен методу GET, за исключением того, что в ответе сервера отсутствует тело. Запрос HEAD обычно применяется для извлечения метаданных, проверки наличия ресурса (валидация URL) и чтобы узнать, не изменился ли он с момента последнего обращения.

Заголовки ответа могут кэшироваться. При несовпадении метаданных ресурса с соответствующей информацией в кэше копия ресурса помечается как устаревшая.

POST

Применяется для передачи пользовательских данных заданному ресурсу. Например, в блогах посетители обычно могут вводить свои комментарии к записям в HTML-форму, после чего они передаются серверу методом POST и он помещает их на страницу. При этом передаваемые данные (в примере с блогами — текст комментария) включаются в тело запроса. Аналогично с помощью метода POST обычно загружаются файлы на сервер.

В отличие от метода GET, метод POST не считается идемпотентным[4], то есть многократное повторение одних и тех же запросов POST может возвращать разные результаты (например, после каждой отправки комментария будет появляться одна копия этого комментария).

При результате выполнения 200 (Ok) в тело ответа следует включить сообщение об итоге выполнения запроса. Если был создан ресурс, то серверу следует вернуть ответ 201 (Created) с указанием URI нового ресурса в заголовке Location.

Сообщение ответа сервера на выполнение метода POST не кэшируется.

PUT

Применяется для загрузки содержимого запроса на указанный в запросе URI. Если по заданному URI не существовало ресурса, то сервер создаёт его и возвращает статус 201 (Created). Если же был изменён ресурс, то сервер возвращает 200 (Ok) или 204 (No Content). Сервер не должен игнорировать некорректные заголовки Content-*, передаваемые клиентом вместе с сообщением. Если какой-то из этих заголовков не может быть распознан или не допустим при текущих условиях, то необходимо вернуть код ошибки 501 (Not Implemented).

Фундаментальное различие методов POST и PUT заключается в понимании предназначений URI ресурсов. Метод POST предполагает, что по указанному URI будет производиться обработка передаваемого клиентом содержимого. Используя PUT, клиент предполагает, что загружаемое содержимое соответствует находящемуся по данному URI ресурсу.

Сообщения ответов сервера на метод PUT не кэшируются.

PATCH

Аналогично PUT, но применяется только к фрагменту ресурса.

DELETE - Удаляет указанный ресурс.

Литература:

а) основная литература:

1. Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципы, технологии, протоколы. – СПб.: Питер 2003.-627 с.

б) дополнительная литература:

1.Таненбаум Э. Компьютерные сети. СПб.:Питер 2007.-902с.