

РАЗДЕЛ 3. ОСНОВЫ ПЕРЕДАЧИ ДАННЫХ

Тема 10. Передача данных по сети.

Уровни модели OSI

Ниже перечислены (в направлении сверху вниз) уровни модели OSI и указаны их общие функции.

Уровень приложения (Application) - интерфейс с прикладными процессами.

Уровень представления (Presentation) - согласование представления (форматов, кодировок) данных прикладных процессов.

Сеансовый уровень (Session) - установление, поддержка и закрытие логического сеанса связи между удаленными процессами.

Транспортный уровень (Transport) - обеспечение безошибочного сквозного обмена потоками данных между процессами во время сеанса.

Сетевой уровень (Network) - фрагментация и сборка передаваемых транспортным уровнем данных, маршрутизация и продвижение их по сети от компьютера-отправителя к компьютеру-получателю.

Канальный уровень (Data Link) - управление каналом передачи данных, управление доступом к среде передачи, передача данных по каналу, обнаружение ошибок в канале и их коррекция.

Физический уровень (Physical) - физический интерфейс с каналом передачи данных, представление данных в виде физических сигналов и их кодирование (модуляция).

Информация в локальных сетях, как правило, передается отдельными порциями, кусками, называемыми в различных источниках пакетами, кадрами или блоками. Использование пакетов связано с тем, что в сети, как правило, одновременно может происходить несколько сеансов связи, то есть в течение одного и того же интервала времени могут идти два или больше процессов передачи данных между различными парами абонентов. Пакеты как раз и позволяют разделить во времени сеть между передающими информацию абонентами.

Если бы вся требуемая информация передавалась сразу, непрерывно, без деления на пакеты, то это привело бы к монопольному захвату сети одним из абонентов на довольно продолжительное время. Все остальные абоненты вынуждены были бы ждать окончания передачи всей информации, что в ряде случаев могло бы потребовать десятков секунд и даже минут (например, при копировании содержимого целого жесткого диска). Чтобы уравнивать в правах всех абонентов, а также примерно уравнивать время доступа к сети и интегральную скорость передачи информации для всех абонентов, как раз и используются пакеты (кадры). Длина пакета зависит от типа сети, но обычно она составляет от нескольких десятков байт до нескольких килобайт.

Структура пакета определяется прежде всего аппаратными особенностями данной сети, выбранной топологией и типом среды передачи информации, а также существенно зависит от используемого протокола (порядка обмена информацией). Строго говоря, в каждой сети структура пакета индивидуальна. Но существуют некоторые общие принципы формирования пакета, определяемые характерными особенностями обмена информацией по любым локальным сетям.

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

Следует также иметь в виду, что приложение может взять на себя функции некоторых верхних уровней модели OSI. Например, некоторые СУБД имеют встроенные средства удаленного доступа к файлам. В этом случае приложение, выполняя доступ к удаленным ресурсам, не использует системную файловую службу; оно обходит верхние уровни модели OSI и обращается напрямую к системным средствам, ответственным за транспортировку сообщений по сети, которые располагаются на нижних уровнях модели OSI.

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

После формирования сообщения прикладной уровень направляет его вниз по стеку представительному уровню. Протокол представительного уровня на основании информации, полученной из заголовка прикладного уровня, выполняет требуемые действия и добавляет к сообщению собственную служебную информацию - заголовок представительного уровня, в котором содержатся указания для протокола представительного уровня машины-адресата. Полученное в результате сообщение передается вниз сеансовому уровню, который в свою очередь добавляет свой заголовок, и т. д. (Некоторые реализации протоколов помещают служебную информацию не только в начале сообщения в виде заголовка, но и в

конце, в виде так называемого «концевика».) Наконец, сообщение достигает нижнего, физического уровня, который собственно и передает его по линиям связи машине-адресату. К этому моменту сообщение «обрастает» заголовками всех уровней (рис. 13).

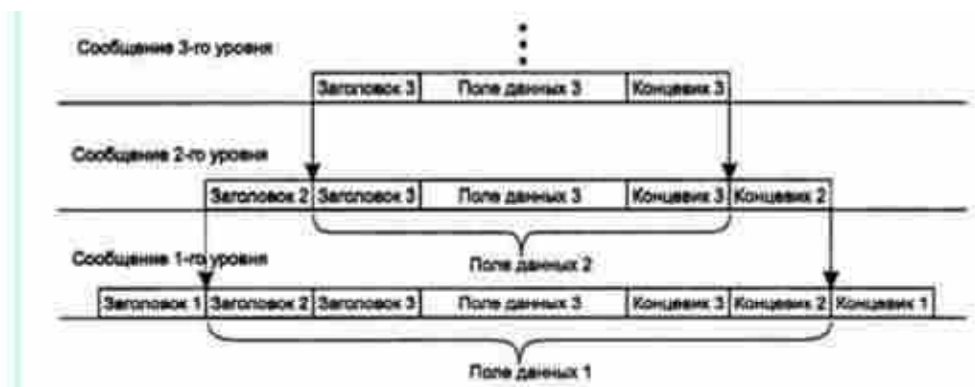


Рисунок 13 – Вложенность сообщений различных уровней

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

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

В модели OSI различаются два основных типа протоколов. В протоколах с *установлением соединения (connection-oriented)* перед обменом данными отправитель и получатель должны сначала установить соединение и, возможно, выбрать некоторые параметры протокола, которые они будут использовать при обмене данными. После завершения диалога они должны разорвать это соединение. Телефон - это пример взаимодействия, основанного на установлении соединения.

Вторая группа протоколов - протоколы *без предварительного установления соединения (connectionless)*. Такие протоколы называются также *дейтаграммными* протоколами. Отправитель просто передает сообщение, когда оно готово. Опускание письма в почтовый ящик - это пример связи без предварительного установления соединения. При взаимодействии компьютеров используются протоколы обоих типов.

Инкапсуляция в компьютерных сетях — это метод построения модульных сетевых протоколов, при котором логически независимые функции сети

абстрагируются от нижележащих механизмов путём включения или инкапсулирования этих механизмов в более высокоуровневые объекты. Например, когда приложению требуется послать сообщение с помощью UDP, то производится последовательность действий:

- в первую очередь приложение заполняет специальную структуру данных, в которой указывает информацию о получателе (сетевой протокол, IP-адрес, порт UDP);

- передаёт сообщение, его длину и структуру с информацией о получателе обработчику протокола UDP (транспортный уровень);

- обработчик UDP формирует датаграмму, в которой в качестве данных выступает сообщение, а в заголовках находится UDP-порт получателя (а также другие данные);

- обработчик UDP передаёт сформированную датаграмму обработчику IP (сетевой уровень);

- обработчик IP рассматривает переданную UDP датаграмму как данные и предваряет их своим заголовком (в котором, в частности, находится IP-адрес получателя, взятый из той же структуры данных приложения, и номер верхнего протокола);

- полученный пакет обработчик IP передаёт на канальный уровень, который опять-таки рассматривает данный пакет как «сырые» данные;

- обработчик канального уровня, аналогично предыдущим обработчикам, добавляет в начало свой заголовок (в котором так же указывается номер протокола верхнего уровня, в нашем случае это 0x0800(IP)) и, в большинстве случаев, добавляет конечную контрольную сумму, тем самым формируя кадр;

- далее полученный кадр передаётся на физический уровень, который осуществляет преобразование битов в электрические или оптические сигналы и посылает их в среду передачи.

То есть, говоря более простым языком, инкапсуляция — упаковка пакетов (возможно, разного протокола) в пакеты одного протокола, включая адрес.

Типичная структура пакета:

- *стартовая комбинация*, или *преамбула*, которая обеспечивает настройку аппаратуры адаптера или другого сетевого устройства на прием и обработку пакета. Это поле может отсутствовать или сводиться к одному-единственному стартовому биту.

- *сетевой адрес (идентификатор) принимающего абонента*, то есть индивидуальный или групповой номер, присвоенный каждому принимающему абоненту в сети. Этот адрес позволяет приемнику распознать пакет, адресованный ему лично, группе, в которую он входит, или всем абонентам сети одновременно.

- *сетевой адрес (идентификатор) передающего абонента*, то есть индивидуальный или групповой номер, присвоенный каждому передающему

абоненту. Этот адрес информирует принимающего абонента, откуда пришел данный пакет. Включение в пакет адреса передатчика необходимо в том случае, когда одному приемнику могут попеременно приходить пакеты от разных передатчиков.

- *служебная информация*, которая указывает на тип пакета, его номер, размер, формат, маршрут его доставки, на то, что с ним надо делать приемнику и т.д.

- *данные* - та информация, ради передачи которой используется данный пакет. Правда, существуют специальные управляющие пакеты, которые не имеют поля данных. Их можно рассматривать как сетевые команды. Пакеты, включающие поле данных, называются информационными пакетами. Управляющие пакеты могут выполнять функцию начала сеанса связи, конца сеанса связи, подтверждения приема информационного пакета, запроса информационного пакета и т.д.

- *контрольная сумма пакета* - это числовой код, формируемый передатчиком по определенным правилам и содержащий в свернутом виде информацию обо всем пакете. Приемник, повторяя вычисления, сделанные передатчиком, с принятым пакетом, сравнивает их результат с контрольной суммой и делает вывод о правильности или ошибочности передачи пакета. Если пакет ошибочен, то приемник запрашивает его повторную передачу.

- *стоповая комбинация* служит для информирования аппаратуры принимающего абонента об окончании пакета, обеспечивает выход аппаратуры приемника из состояния приема. Это поле может отсутствовать, если используется самосинхронизирующийся код, позволяющий детектировать факт передачи пакета.

- В сетях с коммутацией пакетов сегодня применяется два класса механизмов передачи пакетов:

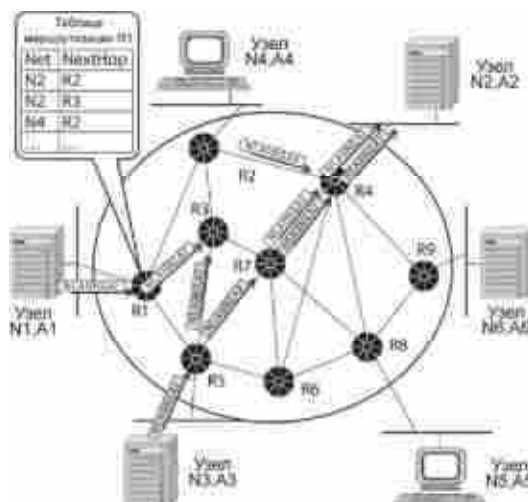
- дейтаграммная передача;
- виртуальные каналы.

- Примерами сетей, реализующих дейтаграммный механизм передачи, являются сети Ethernet, IP и IPX. С помощью виртуальных каналов передают данные сети X.25, frame relay и ATM. Сначала мы рассмотрим базовые принципы дейтаграммного подхода.

- Дейтаграммный способ передачи данных основан на том, что все передаваемые пакеты обрабатываются независимо друг от друга, пакет за пакетом. Принадлежность пакета к определенному потоку между двумя конечными узлами и двумя приложениями, работающими на этих узлах, никак не учитывается.

- Выбор следующего узла — например, коммутатора Ethernet или маршрутизатора IP/IPX — происходит только на основании адреса узла назначения, содержащегося в заголовке пакета. Решение о том, какому узлу передать пришедший пакет, принимается на основе таблицы, содержащей набор адресов назначения и адресную информацию, однозначно определяющую следующий (транзитный или конечный) узел. Такие таблицы имеют разные названия — например, для сетей Ethernet они обычно называются таблицей продвижения

(forwarding table), а для сетевых протоколов, таких как IP и IPX, — таблицами маршрутизации (routing table). Далее для простоты будем пользоваться термином "таблица маршрутизации" в качестве обобщенного названия такого рода таблиц, используемых для дейтаграммной передачи на основании только адреса назначения конечного узла.



– Рисунок 14 – Дейтаграммный принцип передачи пакетов

– В таблице маршрутизации для одного и того же адреса назначения может содержаться несколько записей, указывающих, соответственно, на различные адреса следующего маршрутизатора. Такой подход используется для повышения производительности и надежности сети. В примере на рис. 14 пакеты, поступающие в маршрутизатор R1 для узла назначения с адресом N2, A2, в целях баланса нагрузки распределяются между двумя следующими маршрутизаторами — R2 и R3, что снижает нагрузку на каждый из них, а значит, уменьшает очереди и ускоряет доставку. Некоторая "размытость" путей следования пакетов с одним и тем же адресом назначения через сеть является прямым следствием принципа независимой обработки каждого пакета, присущего дейтаграммным протоколам. Пакеты, следующие по одному и тому же адресу назначения, могут добираться до него разными путями и вследствие изменения состояния сети, например отказа промежуточных маршрутизаторов.

– Такая особенность дейтаграммного механизма как размытость путей следования трафика через сеть также в некоторых случаях является недостатком. Например, если пакетам определенной сессии между двумя конечными узлами сети необходимо обеспечить заданное качество обслуживания. Современные методы поддержки QoS работают эффективней, когда трафик, которому нужно обеспечить гарантии обслуживания, всегда проходит через одни и те же промежуточные узлы.

– Виртуальные каналы в сетях с коммутацией пакетов

– Механизм виртуальных каналов (virtual circuit или virtual channel) создает в сети устойчивые пути следования трафика через сеть с коммутацией пакетов. Этот механизм учитывает существование в сети потоков данных.

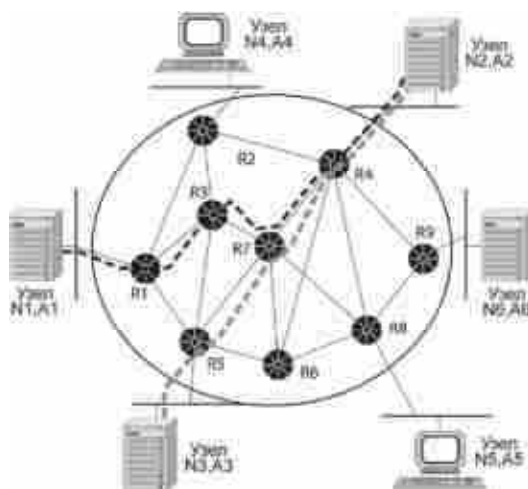


Рисунок 15 – Принцип работы виртуального канала.

Если целью является прокладка для всех пакетов потока единого пути через сеть, то необходимым (но не всегда единственным) признаком такого потока должно быть наличие для всех его пакетов общих точек входа и выхода из сети. Именно для передачи таких потоков в сети создаются виртуальные каналы. На рисунке 15 показан фрагмент сети, в которой проложены два виртуальных канала. Первый проходит от конечного узла с адресом N1, A1 до конечного узла с адресом N2, A2 через промежуточные коммутаторы сети R1, R3, R7 и R4. Второй обеспечивает продвижение данных по пути N3, A3 — R5 — R7 — R4 — N2, A2. Между двумя конечными узлами может быть проложено несколько виртуальных каналов, как полностью совпадающих в отношении пути следования через транзитные узлы, так и отличающихся.

Сеть только обеспечивает возможность передачи трафика вдоль виртуального канала, а какие именно потоки будут передаваться по этим каналам, решают сами конечные узлы. Узел может использовать один и тот же виртуальный канал для передачи всех потоков, которые имеют общие с данным виртуальным каналом конечные точки, или же только части из них. Например, для потока реального времени можно использовать один виртуальный канал, а для трафика электронной почты — другой. В последнем случае разные виртуальные каналы будут предъявлять разные требования к качеству обслуживания, и удовлетворить их будет проще, чем в том случае, когда по одному виртуальному каналу передается трафик с разными требованиями к параметрам QoS.

Важной особенностью сетей с виртуальными каналами является использование локальных адресов пакетов при принятии решения о передаче. Вместо достаточно длинного адреса узла назначения (его длина должна позволять уникально идентифицировать все узлы и подсети в сети, например технология ATM оперирует адресами длиной в 20 байт) применяется локальная, то есть меняющаяся от узла к узлу, метка, которой помечаются все пакеты, перемещаемые по определенному виртуальному каналу. Эта метка в различных технологиях называется по-разному: в технологии X.25 — номер логического канала (Logical Channel number, LCN), в технологии frame relay — идентификатор соединения

уровня канала данных (Data Link Connection Identifier, DLCI), в технологии ATM — идентификатор виртуального канала (Virtual Channel Identifier, VCI). Однако назначение ее везде одинаково — промежуточный узел, называемый в этих технологиях коммутатором, читает значение метки из заголовка пришедшего пакета и просматривает свою таблицу коммутации, в которой указывается, на какой выходной порт нужно передать пакет. Таблица коммутации содержит записи только о проходящих через данный коммутатор виртуальных каналах, а не обо всех имеющихся в сети узлах (или подсетях, если применяется иерархический способ адресации). Обычно в крупной сети количество проложенных через узел виртуальных каналов существенно меньше количества узлов и подсетей, поэтому по размерам таблица коммутации намного меньше таблицы маршрутизации, а, следовательно, просмотр занимает гораздо меньше времени и не требует от коммутатора большой вычислительной мощности.

Идентификатор виртуального канала (именно такое название метки будет использоваться далее) также намного короче адреса конечного узла (по той же причине), поэтому и избыточность заголовка пакета, который теперь не содержит длинного адреса, а переносит по сети только идентификатор, существенно меньше.

Виртуальный путь – это соединение между двумя коммутаторами сети ATM, описанные в таблицах коммутации соответствующих коммутаторов. Виртуальные пути применяются для наиболее часто используемых направлений. По одному виртуальному пути могут передаваться несколько виртуальных каналов. Виртуальный путь существует независимо от того, идет по нему передача данных или нет. Всего виртуальных путей в рамках сети может быть 256. В каждом виртуальном пути м.б. до 65 000 соединений.

Виртуальный канал – это соединение между двумя конечными станциями сети ATM. Виртуальный канал является двунаправленным.

Имеются три вида виртуальных каналов:

1) постоянные виртуальные каналы (PVC). PVC устанавливается вручную в процессе конфигурирования сети.

2) коммутируемые виртуальные каналы (SVC). SVC устанавливается по мере необходимости всякий раз, когда конечная станция пытается передать данные другой станции. Это наиболее часто используемый тип каналов.

3) интеллектуальные постоянные виртуальные каналы (SPVC). SPVC представляет собой гибрид двух предыдущих типов каналов. Данное соединение устанавливается вручную на этапе конфигурирования сети, однако провайдер ATM знает только конечные станции.

Преимущества PVC:

1) не тратится время на установление соединения, поэтому обеспечивается более высокая производительность сети.

2) обеспечивается лучший контроль над сетью.

Недостаток: они должны формироваться вручную

Преимущества SVC:

- 1) данный вид соединения лучше установить или устранить, нежели PVC.
- 2) с помощью SVC могут эмулироваться каналы без установления соединения.
- 3) SVC требует меньше затрат на обслуживание, т.к. данное соединение проводится автоматически, а не вручную.
- 4) данный вид соединения имеет более высокую отказоустойчивость.

Преимущества SPVC:

- 1) Позволяет заранее задать конечные станции, поэтому не приходится тратить время на установление соединения.
- 2) Имеет более высокую отказоустойчивость подобно SVC.

Тема 11. Коммутация каналов, коммутация пакетов, коммутация сообщений.

Разные подходы к выполнению коммутации

В общем случае решение каждой из частных задач коммутации — определение потоков и соответствующих маршрутов, фиксация маршрутов в конфигурационных параметрах и таблицах сетевых устройств, распознавание потоков и передача данных между интерфейсами одного устройства, мультиплексирование/демультиплексирование потоков и разделение среды передачи — тесно связано с решением всех остальных. Комплекс технических решений обобщенной задачи коммутации в совокупности составляет базис любой сетевой технологии. От того, какой механизм прокладки маршрутов, продвижения данных и совместного использования каналов связи заложен в той или иной сетевой технологии, зависят ее фундаментальные свойства.

Среди множества возможных подходов к решению задачи коммутации абонентов в сетях выделяют два основополагающих:

коммутация каналов (circuit switching);

коммутация пакетов (packet switching).

Внешне обе эти схемы соответствуют приведенной на рис. 16 структуре сети, однако возможности и свойства их различны.

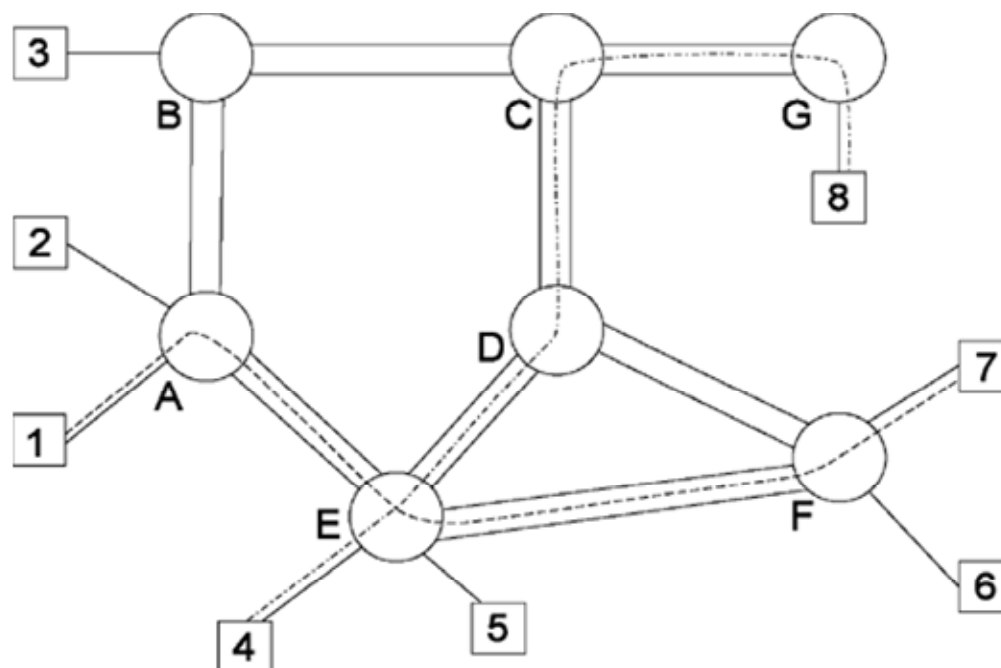


Рисунок 16 – Общая структура сети с коммутацией абонентов

Сети с коммутацией каналов имеют более богатую историю, они произошли от первых телефонных сетей. Сети с коммутацией пакетов сравнительно молоды, они появились в конце 60-х годов как результат экспериментов с первыми глобальными компьютерными сетями. Каждая из этих схем имеет свои достоинства и недостатки, но по долгосрочным прогнозам многих специалистов, будущее принадлежит технологии коммутации пакетов, как более гибкой и универсальной.

Коммутация каналов

При коммутации каналов коммутационная сеть образует между конечными узлами непрерывный составной физический канал из последовательно соединенных коммутаторов промежуточных канальных участков. Условием того, что несколько физических каналов при последовательном соединении образуют единый физический канал, является равенство скоростей передачи данных в каждом из составляющих физических каналов. Равенство скоростей означает, что коммутаторы такой сети не должны буферизовать передаваемые данные.

В сети с коммутацией каналов перед передачей данных всегда необходимо выполнить процедуру установления соединения, в процессе которой и создается составной канал. И только после этого можно начинать передавать данные.

Например, если сеть, изображенная на рис. 16, работает по технологии коммутации каналов, то узел 1, чтобы передать данные узлу 7, сначала должен передать специальный запрос на установление соединения коммутатору А, указав адрес назначения 7. Коммутатор А должен выбрать маршрут образования составного канала, а затем передать запрос следующему коммутатору, в данном случае Е. Затем коммутатор Е передает запрос коммутатору F, а тот, в свою очередь, передает запрос узлу 7. Если узел 7 принимает запрос на установление соединения, он направляет по уже установленному каналу ответ исходному узлу, после чего

составной канал считается скоммутированным, и узлы 1 и 7 могут обмениваться по нему данными.

Техника коммутации каналов имеет свои достоинства и недостатки.

Достоинства коммутации каналов

Постоянная и известная скорость передачи данных по установленному между конечными узлами каналу. Это дает пользователю сети возможности на основе заранее произведенной оценки необходимой для качественной передачи данных пропускной способности установить в сети канал нужной скорости.

Низкий и постоянный уровень задержки передачи данных через сеть. Это позволяет качественно передавать данные, чувствительные к задержкам (называемые также трафиком реального времени) — голос, видео, различную технологическую информацию.

Недостатки коммутации каналов

Отказ сети в обслуживании запроса на установление соединения. Такая ситуация может сложиться из-за того, что на некотором участке сети соединение нужно установить вдоль канала, через который уже проходит максимально возможное количество информационных потоков. Отказ может случиться и на конечном участке составного канала — например, если абонент способен поддерживать только одно соединение, что характерно для многих телефонных сетей. При поступлении второго вызова к уже разговаривающему абоненту сеть передает вызывающему абоненту короткие гудки — сигнал "занято".

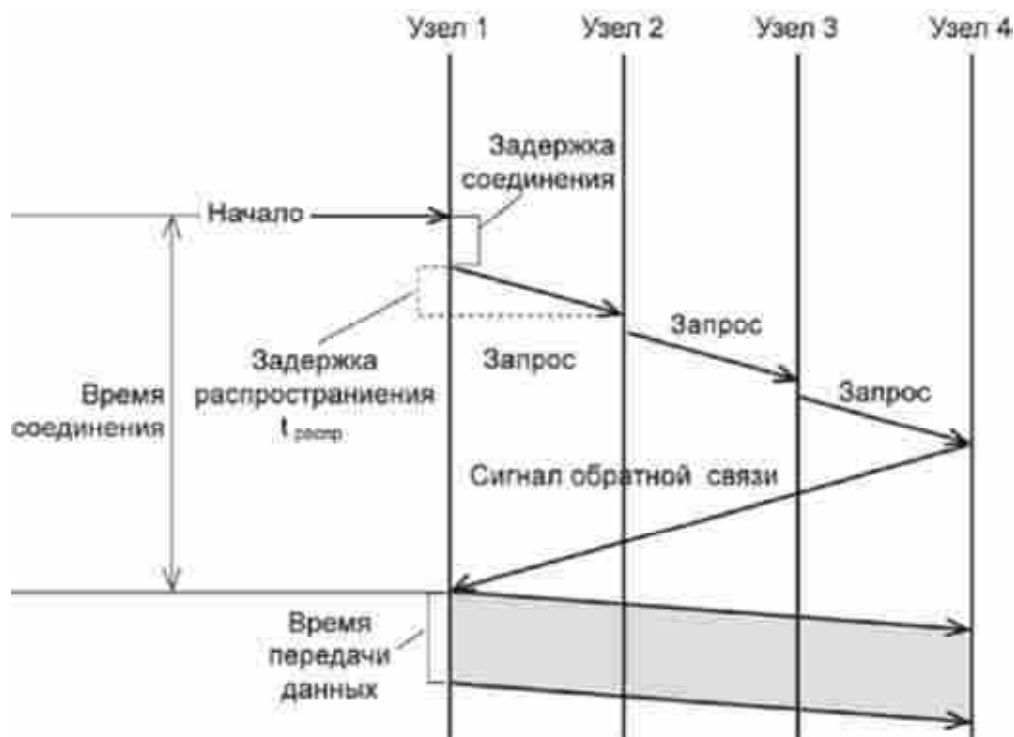


Рисунок 17 – Установление составного канала

Нерациональное использование пропускной способности физических каналов. Та часть пропускной способности, которая отводится составному каналу после установления соединения, предоставляется ему на все время, т.е. до тех пор, пока

соединение не будет разорвано. Однако абонентам не всегда нужна пропускная способность канала во время соединения, например в телефонном разговоре могут быть паузы, еще более неравномерным во времени является взаимодействие компьютеров. Невозможность динамического перераспределения пропускной способности представляет собой принципиальное ограничение сети с коммутацией каналов, так как единицей коммутации здесь является информационный поток в целом.

Обязательная задержка перед передачей данных из-за фазы установления соединения.

Достоинства и недостатки любой сетевой технологии относительны. В определенных ситуациях на первый план выходят достоинства, а недостатки становятся несущественными. Так, техника коммутации каналов хорошо работает в тех случаях, когда нужно передавать только трафик телефонных разговоров. Здесь с невозможностью "вырезать" паузы из разговора и более рационально использовать магистральные физические каналы между коммутаторами можно мириться. А вот при передаче очень неравномерного компьютерного трафика эта нерациональность уже выходит на первый план.

Коммутация пакетов

Эта техника коммутации была специально разработана для эффективной передачи компьютерного трафика. Первые шаги на пути создания компьютерных сетей на основе техники коммутации каналов показали, что этот вид коммутации не позволяет достичь высокой общей пропускной способности сети. Типичные сетевые приложения генерируют трафик очень неравномерно, с высоким уровнем пульсации скорости передачи данных. Например, при обращении к удаленному файловому серверу пользователь сначала просматривает содержимое каталога этого сервера, что порождает передачу небольшого объема данных. Затем он открывает требуемый файл в текстовом редакторе, и эта операция может создать достаточно интенсивный обмен данными, особенно если файл содержит объемные графические включения. После отображения нескольких страниц файла пользователь некоторое время работает с ними локально, что вообще не требует передачи данных по сети, а затем возвращает модифицированные копии страниц на сервер — и это снова порождает интенсивную передачу данных по сети.

Коэффициент пульсации трафика отдельного пользователя сети, равный отношению средней интенсивности обмена данными к максимально возможной, может достигать 1:50 или даже 1:100. Если для описанной сессии организовать коммутацию канала между компьютером пользователя и сервером, то большую часть времени канал будет простаивать. В то же время коммутационные возможности сети будут закреплены за данной парой абонентов и будут недоступны другим пользователям сети.

При коммутации пакетов все передаваемые пользователем сообщения разбиваются в исходном узле на сравнительно небольшие части, называемые

пакетами. Напомним, что сообщением называется логически завершенная порция данных — запрос на передачу файла, ответ на этот запрос, содержащий весь файл и т.д. Сообщения могут иметь произвольную длину, от нескольких байт до многих мегабайт. Напротив, пакеты обычно тоже могут иметь переменную длину, но в узких пределах, например от 46 до 1500 байт. Каждый пакет снабжается заголовком, в котором указывается адресная информация, необходимая для доставки пакета на узел назначения, а также номер пакета, который будет использоваться узлом назначения для сборки сообщения (рис. 18). Пакеты транспортируются по сети как независимые информационные блоки. Коммутаторы сети принимают пакеты от конечных узлов и на основании адресной информации передают их друг другу, а в конечном итоге — узлу назначения.

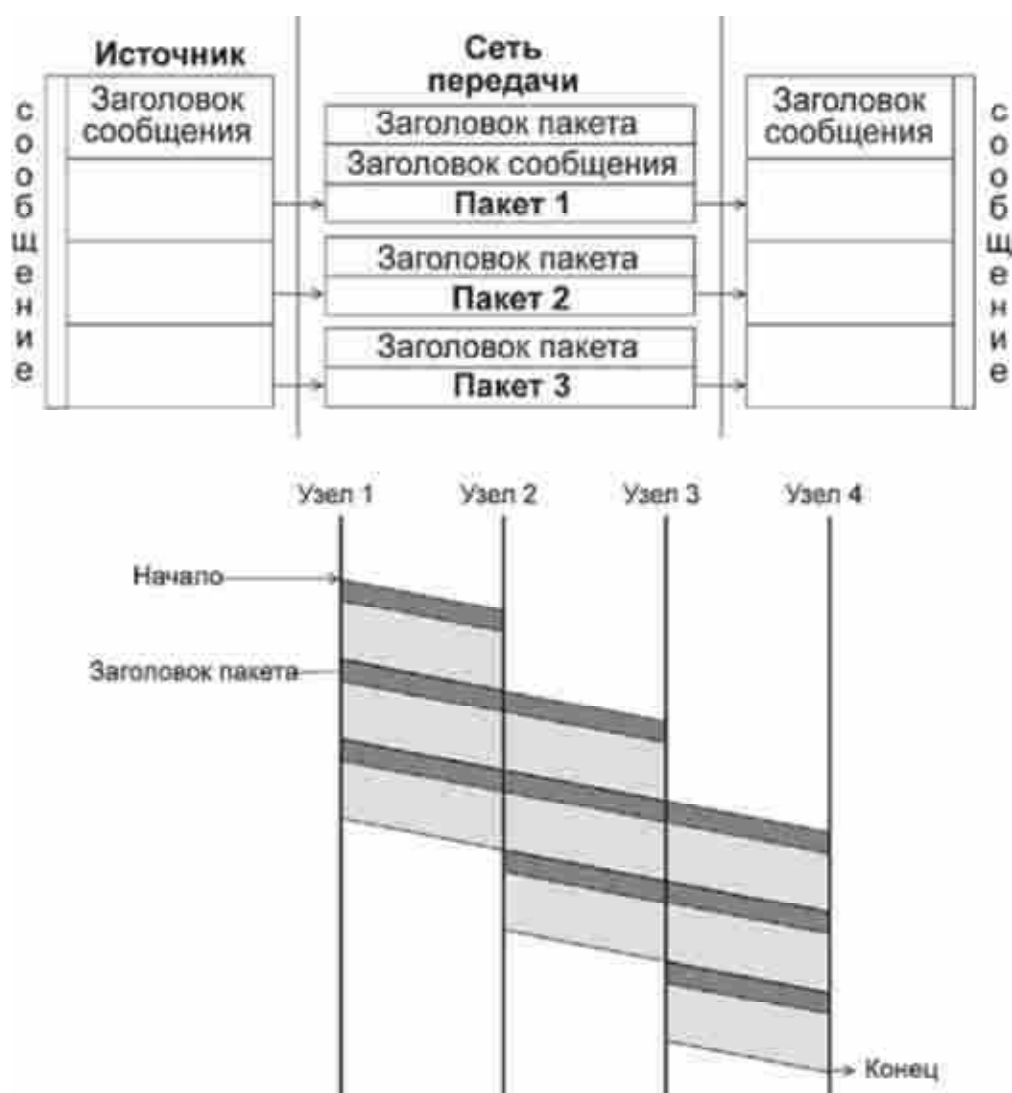


Рисунок 18 – Разбиение сообщения на пакеты

Коммутаторы пакетной сети отличаются от коммутаторов каналов тем, что они имеют внутреннюю буферную память для временного хранения пакетов, если выходной порт коммутатора в момент принятия пакета занят передачей другого пакета (рис. 18). В этом случае пакет находится некоторое время в очереди пакетов в буферной памяти выходного порта, а когда до него дойдет очередь, он передается следующему коммутатору. Такая схема передачи данных позволяет сглаживать

пульсацию трафика на магистральных связях между коммутаторами и тем самым наиболее эффективно использовать их для повышения пропускной способности сети в целом.

Действительно, для пары абонентов наиболее эффективным было бы предоставление им в единоличное пользование скомутированного канала связи, как это делается в сетях с коммутацией каналов. В таком случае время взаимодействия этой пары абонентов было бы минимальным, так как данные без задержек передавались бы от одного абонента другому. Простой канала во время пауз передачи абонентов не интересуют, для них важно быстрее решить свою задачу. Сеть с коммутацией пакетов замедляет процесс взаимодействия конкретной пары абонентов, так как их пакеты могут ожидать в коммутаторах, пока по магистральным связям передаются другие пакеты, пришедшие в коммутатор ранее.

Тем не менее, общий объем передаваемых сетью компьютерных данных в единицу времени при технике коммутации пакетов будет выше, чем при технике коммутации каналов. Это происходит потому, что пульсации отдельных абонентов в соответствии с законом больших чисел распределяются во времени так, что их пики не совпадают. Поэтому коммутаторы постоянно и достаточно равномерно загружены работой, если число обслуживаемых ими абонентов действительно велико. На рис. 19 показано, что трафик, поступающий от конечных узлов на коммутаторы, распределен во времени очень неравномерно. Однако коммутаторы более высокого уровня иерархии, которые обслуживают соединения между коммутаторами нижнего уровня, загружены более равномерно, и поток пакетов в магистральных каналах, соединяющих коммутаторы верхнего уровня, имеет почти максимальный коэффициент использования. Буферизация сглаживает пульсации, поэтому коэффициент пульсации на магистральных каналах гораздо ниже, чем на каналах абонентского доступа — он может быть равным 1:10 или даже 1:2.

Более высокая эффективность сетей с коммутацией пакетов по сравнению с сетями с коммутацией каналов (при равной пропускной способности каналов связи) была доказана в 60-е годы как экспериментально, так и с помощью имитационного моделирования. Здесь уместна аналогия с мультипрограммными операционными системами. Каждая отдельная программа в такой системе выполняется дольше, чем в однопрограммной системе, когда программе выделяется все процессорное время, пока ее выполнение не завершится. Однако общее число программ, выполняемых за единицу времени, в мультипрограммной системе больше, чем в однопрограммной.

Сеть с коммутацией пакетов замедляет процесс взаимодействия конкретной пары абонентов, но повышает пропускную способность сети в целом.

Задержки в источнике передачи:

- время на передачу заголовков;
- задержки, вызванные интервалами между передачей каждого следующего пакета.

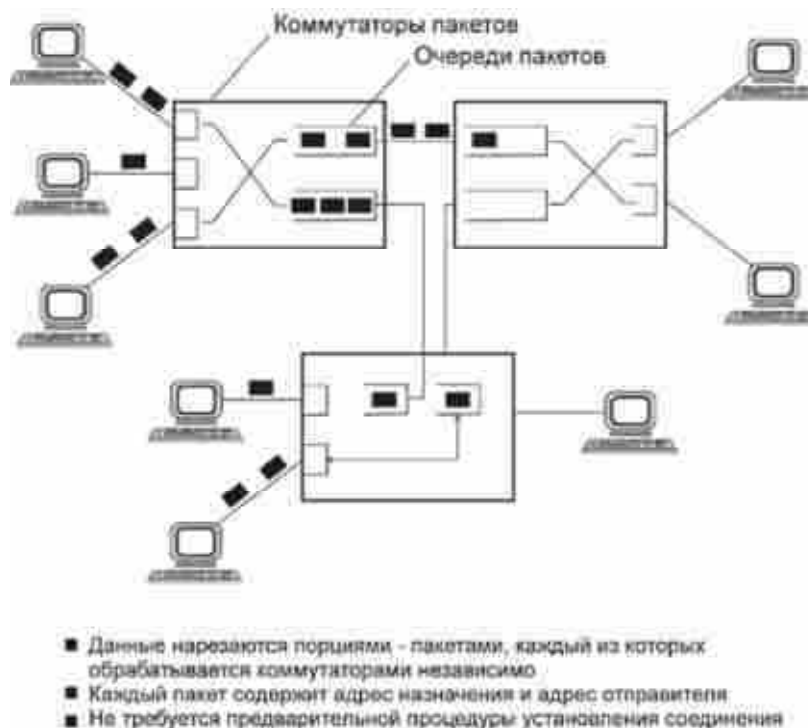


Рисунок 19 – Сглаживание пульсаций трафика в сети с коммутацией пакетов
Задержки в каждом коммутаторе:

- время буферизации пакета;
- время коммутации, которое складывается из:
- времени ожидания пакета в очереди (переменная величина);
- времени перемещения пакета в выходной порт.

Достоинства коммутации пакетов

1. Высокая общая пропускная способность сети при передаче пульсирующего трафика.
2. Возможность динамически перераспределять пропускную способность физических каналов связи между абонентами в соответствии с реальными потребностями их трафика.
3. Недостатки коммутации пакетов
4. Неопределенность скорости передачи данных между абонентами сети, обусловленная тем, что задержки в очередях буферов коммутаторов сети зависят от общей загрузки сети.
5. Переменная величина задержки пакетов данных, которая может быть достаточно продолжительной в моменты мгновенных перегрузок сети.
6. Возможные потери данных из-за переполнения буферов.

В настоящее время активно разрабатываются и внедряются методы, позволяющие преодолеть указанные недостатки, которые особенно остро проявляются для чувствительного к задержкам трафика, требующего при этом постоянной скорости передачи. Такие методы называются методами обеспечения качества обслуживания (Quality of Service, QoS).

Сети с коммутацией пакетов, в которых реализованы методы обеспечения качества обслуживания, позволяют одновременно передавать различные виды

трафика, в том числе такие важные как телефонный и компьютерный. Поэтому методы коммутации пакетов сегодня считаются наиболее перспективными для построения конвергентной сети, которая обеспечит комплексные качественные услуги для абонентов любого типа. Тем не менее, нельзя сбрасывать со счетов и методы коммутации каналов. Сегодня они не только с успехом работают в традиционных телефонных сетях, но и широко применяются для образования высокоскоростных постоянных соединений в так называемых первичных (опорных) сетях технологий SDH и DWDM, которые используются для создания магистральных физических каналов между коммутаторами телефонных или компьютерных сетей. В будущем вполне возможно появление новых технологий коммутации, в том или ином виде комбинирующих принципы коммутации пакетов и каналов.

Коммутация сообщений

Коммутация сообщений по своим принципам близка к коммутации пакетов. Под коммутацией сообщений понимается передача единого блока данных между транзитными компьютерами сети с временной буферизацией этого блока на диске каждого компьютера. Сообщение в отличие от пакета имеет произвольную длину, которая определяется не технологическими соображениями, а содержанием информации, составляющей сообщение.

Транзитные компьютеры могут соединяться между собой как сетью с коммутацией пакетов, так и сетью с коммутацией каналов. Сообщение (это может быть, например, текстовый документ, файл с кодом программы, электронное письмо) хранится в транзитном компьютере на диске, причем довольно продолжительное время, если компьютер занят другой работой или сеть временно перегружена.

По такой схеме обычно передаются сообщения, не требующие немедленного ответа, чаще всего сообщения электронной почты. Режим передачи с промежуточным хранением на диске называется режимом "хранения-и-передачи" (store-and-forward).

Режим коммутации сообщений разгружает сеть для передачи трафика, требующего быстрого ответа, например трафика службы WWW или файловой службы.

Количество транзитных компьютеров обычно стараются уменьшить. Если компьютеры подключены к сети с коммутацией пакетов, то число промежуточных компьютеров уменьшается до двух. Например, пользователь передает почтовое сообщение своему серверу исходящей почты, а тот сразу старается передать его серверу входящей почты адресата. Но если компьютеры связаны между собой телефонной сетью, то часто используется несколько промежуточных серверов, так как прямой доступ к конечному серверу может быть в данный момент невозможен из-за перегрузки телефонной сети (абонент занят) или экономически невыгоден из-за высоких тарифов на дальнюю телефонную связь.

Техника коммутации сообщений появилась в компьютерных сетях раньше техники коммутации пакетов, но потом была вытеснена последней, как более эффективной по критерию пропускной способности сети. Запись сообщения на диск занимает достаточно много времени, и кроме того, наличие дисков предполагает использование в качестве коммутаторов специализированных компьютеров, что влечет за собой существенные затраты на организацию сети.

Сегодня коммутация сообщений работает только для некоторых не оперативных служб, причем чаще всего поверх сети с коммутацией пакетов, как служба прикладного уровня.

Сравнение способов коммутации

Сравнение коммутации каналов и коммутации пакетов

Сравнение коммутации каналов и коммутации пакетов	
Коммутация каналов	Коммутация пакетов
Гарантированная пропускная способность (полоса) для взаимодействующих абонентов	Пропускная способность сети для абонентов неизвестна, задержки передачи носят случайный характер
Сеть может отказать абоненту в установлении соединения	Сеть всегда готова принять данные от абонента
Трафик реального времени передается без задержек	Ресурсы сети используются эффективно при передаче пульсирующего трафика
Адрес используется только на этапе установления соединения	Адрес передается с каждым пакетом

Постоянная и динамическая коммутация

Как сети с коммутацией пакетов, так и сети с коммутацией каналов можно разделить на два класса:

- сети с динамической коммутацией;
- сети с постоянной коммутацией.

В сетях с динамической коммутацией:

- разрешается устанавливать соединение по инициативе пользователя сети;
- коммутация выполняется только на время сеанса связи, а затем (по инициативе одного из пользователей) разрывается;
- в общем случае пользователь сети может соединиться с любым другим пользователем сети;
- время соединения между парой пользователей при динамической коммутации составляет от нескольких секунд до нескольких часов и завершается после выполнения определенной работы — передачи файла, просмотра страницы текста или изображения и т.п.

Примерами сетей, поддерживающих режим динамической коммутации, являются телефонные сети общего пользования, локальные сети, сети TCP/IP.

Сеть, работающая в режиме постоянной коммутации:

- разрешает паре пользователей заказать соединение на длительный период времени;
- соединение устанавливается не пользователями, а персоналом, обслуживающим сеть;
- период, на который устанавливается постоянная коммутация, составляет обычно несколько месяцев;
- режим постоянной (permanent) коммутации в сетях с коммутацией каналов часто называется сервисом выделенных (dedicated) или арендуемых (leased) каналов;
- в том случае, когда постоянное соединение через сеть коммутаторов устанавливается с помощью автоматических процедур, инициированных обслуживающим персоналом, его часто называют полупостоянным (semi-permanent) соединением, в отличие от режима ручного конфигурирования каждого коммутатора.

Наиболее популярными сетями, работающими в режиме постоянной коммутации, сегодня являются сети технологии SDH, на основе которых строятся выделенные каналы связи с пропускной способностью в несколько гигабит в секунду.

Некоторые типы сетей поддерживают оба режима работы. Например, сети X.25 и АТМ могут предоставлять пользователю возможность динамически связаться с любым другим пользователем сети и в то же время отправлять данные по постоянному соединению определенному абоненту.

Пропускная способность сетей с коммутацией пакетов

Одним из отличий метода коммутации пакетов от метода коммутации каналов является неопределенность пропускной способности соединения между двумя абонентами. В случае коммутации каналов после образования составного канала пропускная способность сети при передаче данных между конечными узлами известна — это пропускная способность канала. Данные после задержки, связанной с установлением канала, начинают передаваться на максимальной для канала скорости (рис. 20).

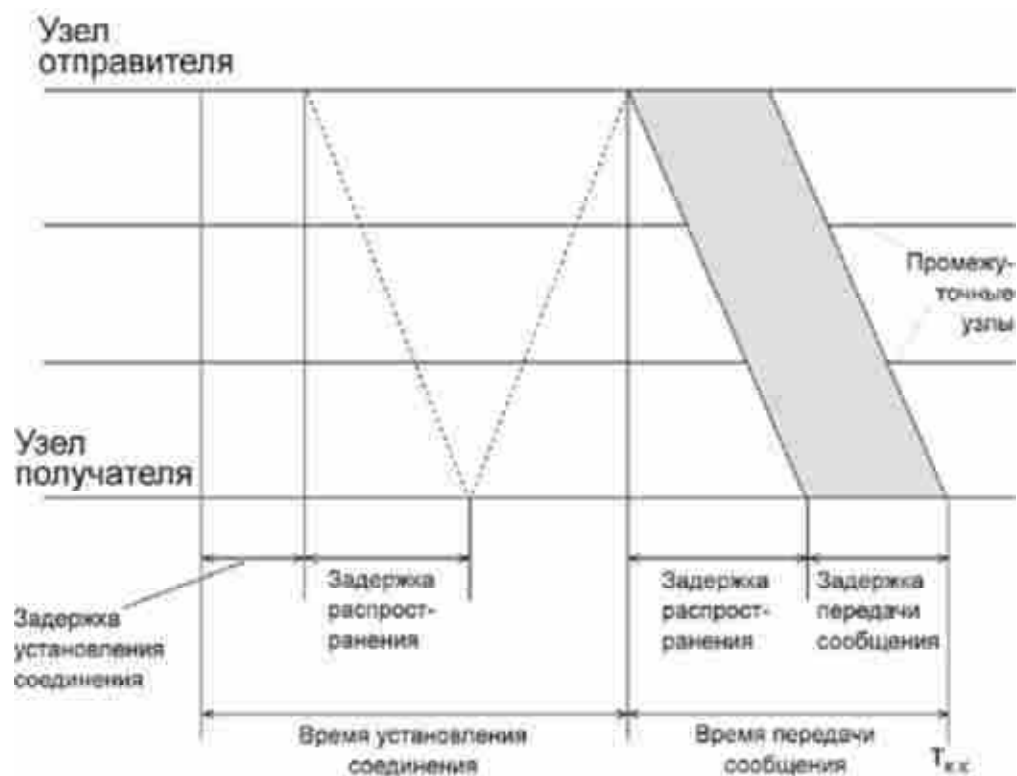


Рисунок 20 – Задержки передачи данных в сетях с коммутацией каналов.

Время передачи сообщения в сети с коммутацией каналов $T_{к.к.}$ равно сумме задержки распространения сигнала по линии связи и задержки передачи сообщения. Задержка распространения сигнала зависит от скорости распространения электромагнитных волн в конкретной физической среде, которая колеблется от 0,6 до 0,9 скорости света в вакууме. Время передачи сообщения равно V/C , где V — объем сообщения в битах, а C — пропускная способность канала в битах в секунду.

В сети с коммутацией пакетов картина совсем иная.

Процедура установления соединения в этих сетях, если она используется, занимает примерно такое же время, как и в сетях с коммутацией каналов, поэтому будем сравнивать только время передачи данных.

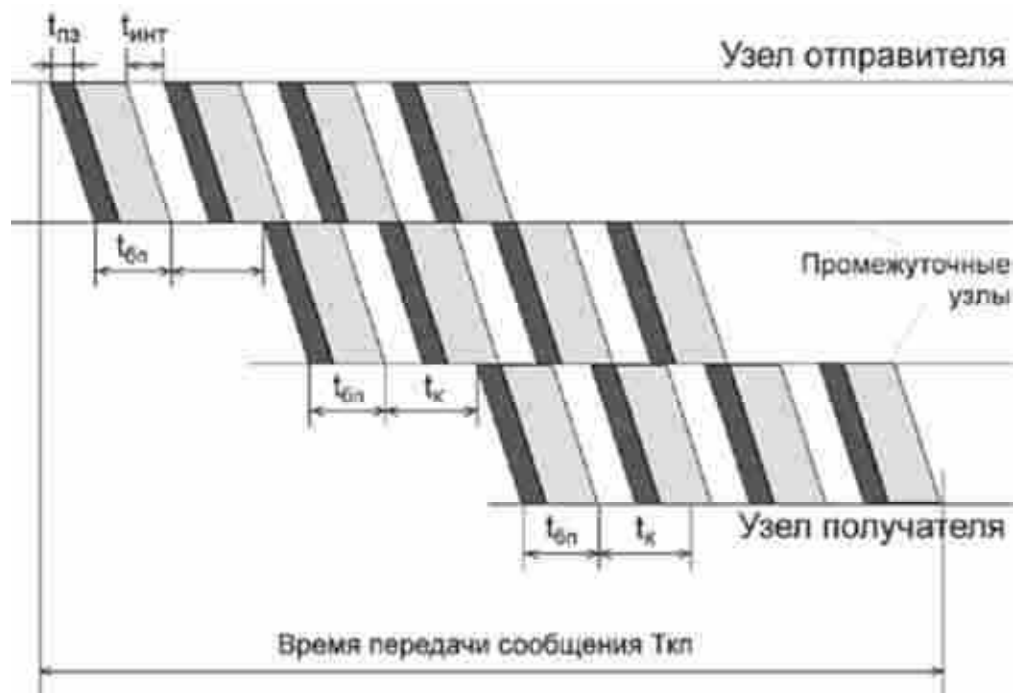


Рисунок 21 – Задержки при передаче данных в сетях с коммутацией пакетов.

На рис. 21 показан пример передачи данных в сети с коммутацией пакетов. Предполагается, что по сети передается сообщение того же объема, что и сообщение, передаваемое на рис. 20 однако оно разделено на пакеты, каждый из которых снабжен заголовком. Время передачи сообщения в сети с коммутацией пакетов обозначено на рисунке $T_{к.п.}$. При передаче этого разбитого на пакеты сообщения по сети с коммутацией пакетов возникают дополнительные задержки. Во-первых, это задержки в источнике передачи, который, помимо передачи собственно сообщения, тратит дополнительное время на передачу заголовков $t_{п.з.}$, к тому же добавляются задержки $t_{инт}$, вызванные интервалами между передачей каждого следующего пакета (это время уходит на формирование очередного пакета стеком протоколов).

Во-вторых, дополнительное время тратится в каждом коммутаторе. Здесь задержки складываются из времени буферизации пакета $t_{б.п.}$ (коммутатор не может начать передачу пакета, не приняв его полностью в свой буфер) и времени коммутации $t_к$. Время буферизации равно времени приема пакета с битовой скоростью протокола. Время коммутации складывается из времени ожидания пакета в очереди и времени перемещения пакета в выходной порт. Если время перемещения пакета фиксировано и, как правило, невелико (от нескольких микросекунд до нескольких десятков микросекунд), то время ожидания пакета в очереди колеблется в очень широких пределах и заранее неизвестно, так как зависит от текущей загрузки сети.

Проведем грубую оценку задержки при передаче данных в сетях с коммутацией пакетов по сравнению с сетями с коммутацией каналов на простейшем примере. Пусть тестовое сообщение, которое нужно передать в обоих видах сетей,

имеет объем 200 Кбайт. Отправитель находится от получателя на расстоянии 5000 км. Пропускная способность линий связи составляет 2 Мбит/с.

Время передачи данных по сети с коммутацией каналов складывается из времени распространения сигнала, которое для расстояния 5000 км можно оценить примерно в 25 мс (принимая скорость распространения сигнала равной $2/3$ скорости света), и времени передачи сообщения, которое при пропускной способности 2 Мбит/с и длине сообщения 200 Кбайт равно примерно 800 мс. При расчете корректное значение K (210), равное 1024, округлялось до 1000, аналогично значение M (220), равное 1048576, округлялось до 1000000. Таким образом, передача данных оценивается в 825 мс.

Ясно, что при передаче этого сообщения по сети с коммутацией пакетов, обладающей такой же суммарной длиной и пропускной способностью каналов, пролегающих от отправителя к получателю, время распространения сигнала и время передачи данных будут такими же — 825 мс. Однако из-за задержек в промежуточных узлах общее время передачи данных увеличится. Давайте оценим, на сколько возрастет это время. Будем считать, что путь от отправителя до получателя пролегает через 10 коммутаторов. Пусть исходное сообщение разбивается на пакеты в 1 Кбайт, всего 200 пакетов. Вначале оценим задержку, которая возникает в исходном узле. Предположим, что доля служебной информации, размещенной в заголовках пакетов, по отношению к общему объему сообщения составляет 10%. Следовательно, дополнительная задержка, связанная с передачей заголовков пакетов, составляет 10% от времени передачи целого сообщения, то есть 80 мс. Если принять интервал между отправкой пакетов равным 1 мс, то дополнительные потери за счет интервалов составят 200 мс. Таким образом, в исходном узле из-за пакетирования сообщения при передаче возникла дополнительная задержка в 280 мс.

Каждый из 10 коммутаторов вносит задержку коммутации, которая может составлять от долей до тысяч миллисекунд. В данном примере будем считать, что на коммутацию в среднем тратится 20 мс. Кроме того, при прохождении сообщений через коммутатор возникает задержка буферизации пакета. Эта задержка при величине пакета 1 Кбайт и пропускной способности линии 2 Мбит/с равна 4 мс. Общая задержка, вносимая 10 коммутаторами, составляет примерно 240 мс. В результате дополнительная задержка, созданная сетью с коммутацией пакетов, составила 520 мс. Учитывая, что вся передача данных в сети с коммутацией каналов заняла 825 мс, эту дополнительную задержку можно считать существенной.

Хотя приведенный расчет носит очень приблизительный характер, он объясняет, почему процесс передачи для определенной пары абонентов в сети с коммутацией пакетов является более медленным, чем в сети с коммутацией каналов.

Неопределенная пропускная способность сети с коммутацией пакетов — это плата за ее общую эффективность при некотором ущемлении интересов отдельных абонентов. Аналогично, в мультипрограммной операционной системе время

выполнения приложения предсказать невозможно, так как оно зависит от количества других приложений, с которыми данное приложение делит процессор.

На эффективность работы сети влияют размеры пакетов, которые передает сеть. Слишком большие размеры пакетов приближают сеть с коммутацией пакетов к сети с коммутацией каналов, поэтому эффективность сети падает. Кроме того, при большом размере пакетов увеличивается время буферизации на каждом коммутаторе. Слишком маленькие пакеты заметно увеличивают долю служебной информации, так как каждый пакет содержит заголовок фиксированной длины, а количество пакетов, на которые разбиваются сообщения, при уменьшении размера пакета будет резко расти. Существует некоторая "золотая середина", когда обеспечивается максимальная эффективность работы сети, однако это соотношение трудно определить точно, так как оно зависит от многих факторов, в том числе изменяющихся в процессе работы сети. Поэтому разработчики протоколов для сетей с коммутацией пакетов выбирают пределы, в которых может находиться размер пакета, а точнее его поле данных, так как заголовок, как правило, имеет фиксированную длину. Обычно нижний предел поля данных выбирается равным нулю, что дает возможность передавать служебные пакеты без пользовательских данных, а верхний предел не превышает 4 Кбайт. Приложения при передаче данных пытаются занять максимальный размер поля данных, чтобы быстрее выполнить обмен, а небольшие пакеты обычно используются для коротких служебных сообщений, содержащих, к примеру, подтверждение доставки пакета.

При выборе размера пакета необходимо также учитывать интенсивность битовых ошибок канала. На ненадежных каналах необходимо уменьшать размеры пакетов, так как это сокращает объем повторно передаваемых данных при искажениях пакетов.

Ethernet — пример стандартной технологии коммутации пакетов

Рассмотрим, каким образом описанные выше общие подходы к решению проблем построения сетей воплощены в наиболее популярной сетевой технологии — Ethernet. (Заметим, что мы не будем сейчас подробно рассматривать саму технологию — отложим этот важный вопрос до следующего курса, а сегодня остановимся лишь на некоторых принципиальных моментах, иллюстрирующих ряд уже рассмотренных базовых концепций.)

Сетевая технология — это согласованный набор стандартных протоколов и программно-аппаратных средств (например, сетевых адаптеров, драйверов, кабелей и разъемов), достаточный для построения вычислительной сети.

Эпитет "достаточный" подчеркивает то обстоятельство, что речь идет о минимальном наборе средств, с помощью которых можно построить работоспособную сеть. Эту сеть можно усовершенствовать, например, за счет выделения в ней подсетей, что сразу потребует кроме протоколов стандарта Ethernet применения протокола IP, а также специальных коммуникационных устройств — маршрутизаторов. Усовершенствованная сеть будет, скорее всего, более надежной и

быстродействующей, но за счет надстроек над средствами технологии Ethernet, которая составила базис сети.

Термин "сетевая технология" чаще всего используется в описанном выше узком смысле, но иногда применяется и его расширенное толкование как любого набора средств и правил для построения сети, например "технология сквозной маршрутизации", "технология создания защищенного канала", "технология IP-сетей".

Протоколы, на основе которых строится сеть определенной технологии (в узком смысле), создавались специально для совместной работы, поэтому от разработчика сети не требуется дополнительных усилий по организации их взаимодействия. Иногда сетевые технологии называют базовыми технологиями, имея в виду, что на их основе строится базис любой сети. Примерами базовых сетевых технологий могут служить наряду с Ethernet такие известные технологии локальных сетей как Token Ring и FDDI, или же технологии территориальных сетей X.25 и frame relay. Для получения работоспособной сети в этом случае достаточно приобрести программные и аппаратные средства, относящиеся к одной базовой технологии — сетевые адаптеры с драйверами, концентраторы, коммутаторы, кабельную систему и т. п., — и соединить их в соответствии с требованиями стандарта на данную технологию.

Итак, для сетевой технологии Ethernet характерны:

- коммутация пакетов;
- типовая топология "общая шина";
- плоская числовая адресация;
- разделяемая передающая среда.

Основной принцип, положенный в основу Ethernet, — случайный метод доступа к разделяемой среде передачи данных. В качестве такой среды может использоваться толстый или тонкий коаксиальный кабель, витая пара, оптоволокно или радиоволны (кстати, первой сетью, построенной на принципе случайного доступа к разделяемой среде, была радиосеть Aloha Гавайского университета).

В стандарте Ethernet строго зафиксирована топология электрических связей. Компьютеры подключаются к разделяемой среде в соответствии с типовой структурой "общая шина" (рис. 22). С помощью разделяемой во времени шины любые два компьютера могут обмениваться данными. Управление доступом к линии связи осуществляется специальными контроллерами — сетевыми адаптерами Ethernet. Каждый компьютер, а точнее, каждый сетевой адаптер, имеет уникальный адрес. Передача данных происходит со скоростью 10 Мбит/с. Эта величина является пропускной способностью сети Ethernet.



Рисунок 22 – Сеть Ethernet.

Суть случайного метода доступа состоит в следующем. Компьютер в сети Ethernet может передавать данные по сети, только если сеть свободна, то есть если никакой другой компьютер в данный момент не занимается обменом. Поэтому важной частью технологии Ethernet является процедура определения доступности среды.

После того как компьютер убедился, что сеть свободна, он начинает передачу и при этом "захватывает" среду. Время монопольного использования разделяемой среды одним узлом ограничивается временем передачи одного кадра. Кадр — это единица данных, которыми обмениваются компьютеры в сети Ethernet. Кадр имеет фиксированный формат и наряду с полем данных содержит различную служебную информацию, например адрес получателя и адрес отправителя.

Сеть Ethernet устроена так, что при попадании кадра в разделяемую среду передачи данных все сетевые адаптеры начинают одновременно принимать этот кадр. Все они анализируют адрес назначения, располагающийся в одном из начальных полей кадра, и, если этот адрес совпадает с их собственным, кадр помещается во внутренний буфер сетевого адаптера. Таким образом компьютер-адресат получает предназначенные ему данные.

Может возникнуть ситуация, когда несколько компьютеров одновременно решают, что сеть свободна, и начинают передавать информацию. Такая ситуация, называемая коллизией, препятствует правильной передаче данных по сети. В стандарте Ethernet предусмотрен алгоритм обнаружения и корректной обработки коллизий. Вероятность возникновения коллизии зависит от интенсивности сетевого трафика.

После обнаружения коллизии сетевые адаптеры, которые пытались передать свои кадры, прекращают передачу и после паузы случайной длительности пытаются снова получить доступ к среде и передать тот кадр, который вызвал коллизию.

Основные достоинства технологии Ethernet

Главным достоинством сетей Ethernet, благодаря которому они стали такими популярными, является их экономичность. Для построения сети достаточно иметь по одному сетевому адаптеру для каждого компьютера плюс один физический сегмент коаксиального кабеля нужной длины.

Кроме того, в сетях Ethernet реализованы достаточно простые алгоритмы доступа к среде, адресации и передачи данных. Простота логики работы сети ведет к упрощению и, соответственно, снижению стоимости сетевых адаптеров и их драйверов. По той же причине адаптеры сети Ethernet обладают высокой надежностью.

И, наконец, еще одним замечательным свойством сетей Ethernet является их хорошая расширяемость, то есть возможность подключения новых узлов.

Другие базовые сетевые технологии, такие как Token Ring и FDDI, хотя и обладают индивидуальными чертами, в то же время имеют много общего с Ethernet. В первую очередь, это применение регулярных фиксированных топологий ("иерархическая звезда" и "кольцо"), а также разделяемых сред передачи данных. Существенные отличия одной технологии от другой связаны с особенностями используемого метода доступа к разделяемой среде. Так, отличия технологии Ethernet от технологии Token Ring во многом определяются спецификой заложенных в них методов разделения среды — случайного алгоритма доступа в Ethernet и метода доступа путем передачи маркера в Token Ring.

РАЗДЕЛ 4. СТЕК ПРОТОКОЛОВ TCP/IP

Тема 12. Протокол IP

Ранние эксперименты по передаче и приему информации с помощью компьютеров начались еще в 50-х годах и имели лабораторный характер. Лишь в конце 60-х годов на средства Агентства Перспективных Разработок министерства обороны США была создана **сеть национального масштаба**. Она получила название **ARPANET**. Эта сеть связывала несколько крупных научных, исследовательских и образовательных центров. Ее основной задачей была координация групп коллективов, работающих над едиными научно-техническими проектами, а основным назначением стал обмен электронной почтой файлами с научной и проектно-конструкторской документацией.

Сеть ARPANET заработала в 1969 году. Немногочисленные узлы, входившие в нее в то время, были связаны выделенными линиями. Прием и передача информации обеспечивались программами, работающими на узловых компьютерах. Сеть постепенно расширялась за счет подключения новых узлов, а к началу 80-х годов на базе наиболее крупных узлов были созданы свои региональные сети, воссоздающие общую архитектуру ARPANET на более низком уровне (в региональном или локальном масштабе).

По-настоящему **рождением Интернета** принято считать 1983 год. В этом году произошли революционные изменения в программном обеспечении компьютерной связи. Днем рождения Интернета в современном понимании этого слова стала дата стандартизации протокола связи TCP/IP, лежащего в основе Всемирной сети по нынешний день.

TCP/IP — это не один сетевой протокол, а несколько протоколов, лежащих на разных уровнях сетевой модели OSI (это так называемый стек протоколов). Из них протокол TCP — протокол транспортного уровня. Он управляет тем, как происходит передача информации. Протокол IP — адресный. Он принадлежит сетевому уровню и определяет, куда происходит передача.

Протокол TCP.

Согласно протоколу TCP, отправляемые данные «нарезаются» на небольшие пакеты, после чего каждый пакет маркируется таким образом, чтобы в нем были данные, необходимые для правильной сборки документа на компьютере получателя.

Для понимания сути протокола TCP можно представить игру в шахматы по переписке, когда двое участников разыгрывают одновременно десяток партий. Каждый ход записывается на отдельной открытке с указанием номера партии и номера хода. В этом случае между двумя партнерами через один и тот же почтовый канал работает как бы десяток соединений (по одному на партию). Два компьютера, связанные между собой одним физическим соединением, могут точно так же поддерживать одновременно несколько TCP-соединений. Так, например, два

промежуточных сетевых сервера могут одновременно по одной линии связи передавать друг другу в обе стороны множество TCP-пакетов от многочисленных клиентов.

Протокол IP.

Суть адресного протокола - IP (Internet Protocol) - состоит в том, что у каждого участника Всемирной сети должен быть свой уникальный адрес (IP-адрес). Без этого нельзя говорить о точной доставке TCP-пакетов на нужное рабочее место. Этот адрес выражается очень просто — четырьмя байтами, например: 195.38.46.11.

Поскольку один байт содержит до 256 различных значений, то теоретически с помощью четырех байтов можно выразить более четырех миллиардов уникальных IP-адресов (256^4 за вычетом некоторого количества адресов, используемых в качестве служебных). На практике же из-за особенностей адресации к некоторым типам локальных сетей количество возможных адресов составляет порядка двух миллиардов, но и это по современным меркам достаточно большая величина.

Протокол TCP/ IP и его основные свойства

Основой сети Интернет является стек протоколов *TCP/ IP (Transmission Control Protocol/ Internet Protocol)*.

Основными преимуществами протокола TCP/ IP являются:

- *Независимость от сетевой технологии отдельной сети.* TCP/ IP не зависит от оборудования, так как он определяет только элемент передачи, который называется *дейтаграммой*, и описывает способ ее движения по сети.
- *Всеобщая связанность сетей.* Протокол позволяет любой паре компьютеров взаимодействовать друг с другом. Каждому компьютеру назначается логический адрес, а каждая передаваемая дейтаграмма содержит адреса отправителей и получателей. Промежуточные маршрутизаторы используют адрес получателя для принятия решения о маршрутизации.
- *Подтверждение.* Протокол TCP/IP обеспечивает подтверждение правильно прохождения информации при обмене между отправителем и получателем.
- *Стандартные прикладные протоколы.* Протокол TCP/IP включает в свой состав поддержку основных приложений, таких как электронная почта, передача файлов, удаленный доступ и т.д.

В стеке TCP/ IP определены 4 уровня взаимодействия, каждый из которых берет на себя определенную функцию по организации надежной работы глобальной сети:

Уровень I	Прикладной уровень
Уровень II	Основной (транспортный) уровень
Уровень III	Уровень межсетевого взаимодействия
Уровень IV	Уровень сетевых интерфейсов

Уровень межсетевого взаимодействия.

Уровень межсетевого взаимодействия является стержнем всей архитектуры протокола, который реализует концепцию передачи пакетов в режиме без установления соединений, то есть дейтаграммным способом. Именно этот уровень обеспечивает возможность перемещения пакетов по сети, используя тот маршрут, который в данный момент является оптимальным. Этот уровень также называют *уровнем Интернет*, подчеркивая его основную функцию- передачу данных через составную сеть. Основным протоколом уровня межсетевого взаимодействия является протокол IP (Internet Protocol). IP – протокол проектировался для передачи пакетов в составных сетях, состоящих из большого количества локальных сетей, поэтому он хорошо работает в сетях со сложной топологией. Так как IP- протокол является дейтаграммным протоколом, то он не гарантирует доставку пакетов до узла назначения.

Основной (транспортный) уровень.

Так как на сетевом уровне не происходит установление соединения, то нет никаких гарантий, что межсетевым уровнем пакеты будут доставлены в место назначения неповрежденными. Обеспечения надежной связи между двумя конечными компьютерами осуществляет основной уровень стека TCP/IP, называемый также транспортным. На этом уровне работает протокол управления передачей TCP (Transmission Control Protocol) и протокол дейтаграмм пользователя UDP (User Datagram Protocol). Основной задачей TCP является доставка всей информации компьютеру получателя, контроль последовательности передаваемой информации, повторная отправка не доставленных пакетов в случае сбоев работы сети. Надежность доставки информации достигается следующим образом.

На передающем компьютере TCP разбивает блок данных, поступающих с прикладного уровня, на отдельные *сегменты*, присваивает номера сегментам, добавляет заголовок и передает сегменты на уровень межсетевого взаимодействия. При этом размер сегмента должен быть таким, чтобы он полностью помещался в IP - пакет. Для каждого отправленного сегмента передающий компьютер ожидает прихода от принимающего компьютера специального сообщения – квитанции, подтверждающей тот факт, что компьютер нужный сегмент принял. Время ожидания прихода соответствующей квитанции называется *временем тайм- аута*. Переданный сегмент хранится в буфере на все время ожидания квитанции. В случае получения квитанции о правильности приема, TCP передает следующий сегмент, удаляя переданный из буфера, а в случае отсутствия квитанции о подтверждении приема, TCP повторяет передачу сегмента. Для ускорения передачи сегментов в протоколе TCP организован принцип их передачи, который называется принцип «скользящего окна». Этот принцип основывается на возможности передачи нескольких сегментов в пределах одного «окна», не дожидаясь прихода квитанции на первый отправленный сегмент. На принимающем компьютере TCP, получая от уровня межсетевого взаимодействия сегменты, собирает их в блок по номерам и

передает этот блок на верхний уровень приложений, отправляя обратно в сети квитанции о правильности принятого сегмента. Для производительности сети является очень важным установления времени тайм-аута и размера «скользящего окна». В общем случае для их выбора необходимо учитывать пропускную способность физических линий связи, отметим, однако, что в протоколе TCP предусмотрен специальный автоматический алгоритм определения этих величин.

В задачи протокола TCP входит также важнейшая задача определения к какому типу прикладных программ относятся данные, поступившие из сети. Прикладные программы с точки зрения TCP различаются специальными идентификаторами, которые называются *портами*. Назначение номеров портов осуществляется либо централизованно, если прикладные программы являются популярными и общедоступными (например, служба удаленного доступа к файлам FTP имеет порт 21, а служба WWW – порт 80), или локально – если разработчик своего приложения просто связывает с этим приложением любой доступный, произвольно выбранный номер. В дальнейшем все запросы к данному приложению от других приложений должны адресоваться с указанием назначенного ему номера порта. *Номер порта в совокупности с номером сети и номером конечного хоста однозначно определяют процессы в сети Интернет.* Этот набор идентифицирующих параметров процесса носит название *сокета*. Отметим также, что протокол TCP управляет двумя очередями: очередь пакетов, поступающих из сети и очередь пакетов, поступающих из прикладного уровня по соответствующему порту.

Протокол UDP был разработан для пользователей, не нуждающихся в услугах протокола TCP. Этот протокол, в отличие от TCP, не обеспечивает достоверность доставки пакетов и надежность от сбоев в передаче информации. К IP-пакету он добавляет только номера портов верхнего уровня. Преимущество этого протокола состоит в том, что он требует минимум установок и параметров для передачи информации и используется для наиболее простых протоколов верхнего уровня (например, для Простого протокола управления сетью - Simple Network Management Protocol, SNMP).

Прикладной уровень.

Прикладной уровень объединяет все службы пользователей сети. Прикладной уровень реализуется различными программными системами и постоянно расширяется. Наиболее известными прикладными службами являются электронная почта (E-mail), система новостей UseNet, всемирная паутина World Wide Web (WWW), передача файлов (FTP), удаленный терминал и терминальные серверы (TELNET) и д.р. Указанные службы рассмотрим в следующей лекции.

Уровень сетевых интерфейсов.

В отличие от физического и канального уровня модели OSI в архитектуре стека TCP/IP существует несколько другая интерпретация уровня сетевых

интерфейсов. Протоколы этого уровня должны обеспечить интеграцию в составную сеть локальных сетей, использующих различные технологии. Поэтому разработчики той или другой технологии должны предусмотреть возможность инкапсуляции (включения) в свои кадры IP – пакетов. Уровень сетевых интерфейсов в протоколах TCP/ IP не регламентируется, но он поддерживает все популярные стандарты физического и канального уровня: Ethernet, Token Ring, FDDI, Gigabit Ethernet, Fast Ethernet и д.р. Для глобальных сетей имеется возможность работы с протоколами SLIP и PPP. Разработаны спецификации для соединения с сетями X.25, frame relay, ATM.

Отметим, что в настоящее время каждый из разработчиков сетевых технологий канального и физического уровня стремиться обеспечить их совместимость с протоколом TCP/ IP.

Соответствие уровней стека TCP/ IP семиуровневой модели OSI

Соответствие стека TCP/IP модели OSI показано на рис 23.

Как видно из рисунка 23 протокол TCP занимает транспортный и сеансовый уровень, а на сетевом уровне используется протокол IP. Отметим, что в модели TCP/IP программные модули, соответствующие транспортному и сеансовому уровню, устанавливаются только на конечных компьютерах.

Программный модуль протокола TCP/ IP реализуется в операционной системе компьютера в виде отдельного системного модуля (драйвера). Интерфейс между прикладным уровнем и TCP представляет собой библиотеку вызовов, такую же, как, например, библиотека системных вызовов для работы с файлами. Пользователь может самостоятельно настраивать протокол TCP/ IP для каждого конкретного случая (количество пользователей сети, пропускная способность физических линий связи и т.д.).

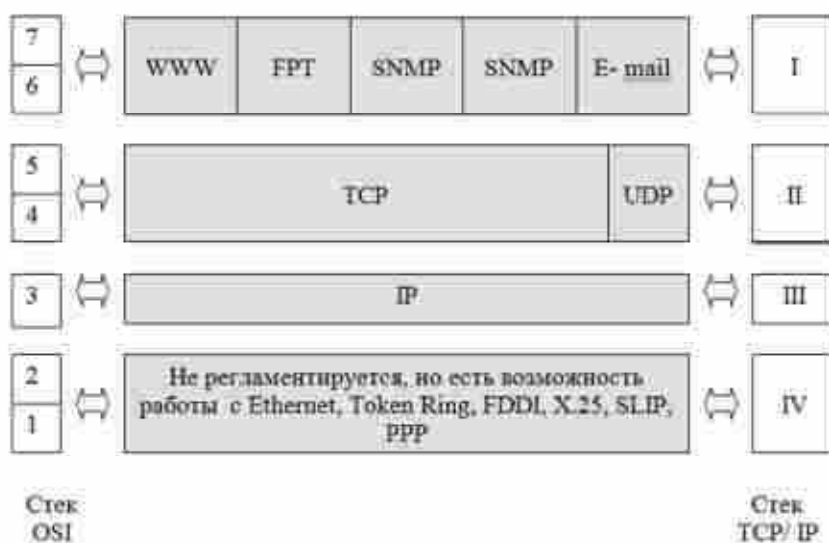


Рисунок 23 – Соответствие стека TCP/IP модели

Адресация в IP - сетях

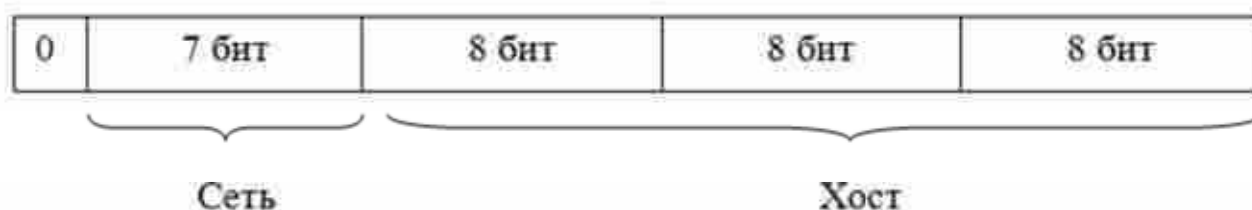
IP- адресация компьютеров в сети Интернет построена на концепции сети, состоящей из хостов. *Хост* представляет собой объект сети, который может передавать и принимать IP- пакеты, например, компьютер, рабочая станция или маршрутизатор. Хосты соединяются между собой через одну или несколько сетей. IP – адрес любого из хостов состоит *из адреса (номера) сети и адреса хоста в этой сети.*

В соответствии принятым в момент разработки IP – протокола соглашением, адрес представляется четырьмя десятичными числами, разделенными точками. Каждое из этих чисел не может превышать 255 и представляет один байт 4-байтного IP- адреса. Выделение всего лишь четырех байт для адресации всей сети Интернет связано с тем, что в то время массового распространения локальных сетей пока не предвиделось. О персональных компьютерах и рабочих станциях вообще не было речи. В результате под IP-адрес было отведено 32 бита, из которых первые 8 бит обозначали сеть, а оставшиеся 24 бита — компьютер в сети. IP – адрес назначается администратором сети во время конфигурирования компьютеров и маршрутизаторов. Номер сети может быть выбран администратором произвольным образом, или назначен по рекомендации специального подразделения Интернет – InterNIC. Обычно поставщики услуг Интернет получают диапазоны адресов у подразделений InterNIC, а затем распределяют их среди своих абонентов. Отметим, что маршрутизатор может входить сразу в несколько сетей, поэтому каждый порт маршрутизатора имеет свой IP – адрес. Таким же образом и конечный компьютер так же может входить в несколько сетей, а значит иметь несколько IP- адресов. Таким образом IP- адрес характеризует не отдельный компьютер или маршрутизатор, а одно сетевое соединение.

Как уже отмечалось выше, адрес состоит из двух частей - номера сети и номера узла в сети. Для того, чтобы определить, какая часть адреса относится к номеру сети, а какая к номеру узла, в начале адреса несколько бит отводится для определения класса сети.

IP-адресация определяет пять классов сетей.

Класс А.



Сети класса А предназначены главным образом для использования крупными организациями, их адрес начинается с 0 в двоичной записи, или с 1 в десятичной записи, они имеют номера от 1 до 126 (если все семь бит равны «1» = 1111111 = 127,

номер сети 0 не используется, а номер 127 используется для специальных целей). В сетях класса А предусмотрено большое количество узлов – $2^{24} = 16\,777\,216$ узлов.

Пример.

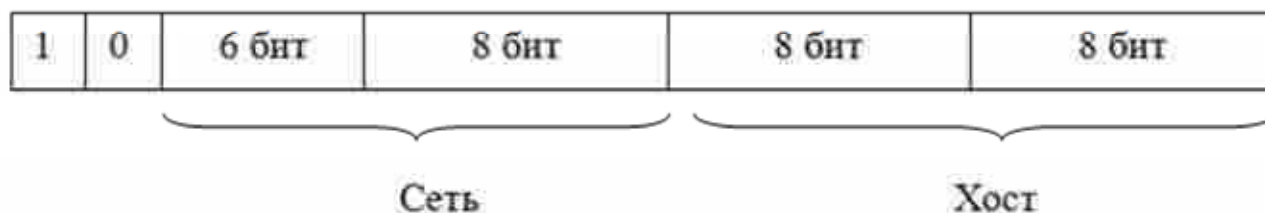
Узел имеет минимально возможный номер в сети класса А с минимально возможным номером сети

$$\underbrace{00000001}_1. \underbrace{00000000}_0. \underbrace{00000000}_0. \underbrace{00000001}_1 = 1.0.0.1$$

Узел имеет максимально возможный номер в сети класса А с максимально возможным номером сети

$$\underbrace{01111110}_{126}. \underbrace{11111111}_{255}. \underbrace{11111111}_{255}. \underbrace{11111110}_{254} = 126.255.255.254$$

Класс В.



В сетях класса В выделяют 14 бит для номера сети и 16 бит для номеров хостов, их адрес начинается с 10 в двоичной записи, или со 128 в десятичной записи, они имеют номера от 128.0 до 191.255 ($10000000.00000000 = 128.0$, $10111111.11111111 = 191.255$). Сети В представляют хороший компромисс между адресным пространством номера сети и номерами хостов. Сеть класса В является сетью среднего размера с максимальным числом узлов $2^{16} = 65\,536$.

Пример.

Узел имеет минимально возможный номер в сети класса В с минимально возможным номером сети

$$\underbrace{10000000}_{128}. \underbrace{00000000}_0. \underbrace{00000000}_0. \underbrace{00000001}_1 = 128.0.0.1$$

Узел имеет максимально возможный номер в сети класса В с максимально возможным номером сети

$$\underbrace{10111111}_{191}. \underbrace{11111111}_{255}. \underbrace{11111111}_{255}. \underbrace{11111110}_{254} = 191.255.255.254$$

Класс C.



Сети класса C выделяют 22 бита для номера сети и 8 бит для номеров хостов, их адрес начинается с 110 в двоичной записи, или со 192 в десятичной записи, они имеют номера от 192.0.0 до 223.255.255 ($11000000.00000000.00000000 = 192.0.0$, $11011111.11111111.11111111 = 223.255.255$). Сети класса C являются наиболее распространенными сетями, число узлов в одной сети равно $2^8 = 256$.

Пример.

Узел имеет минимально возможный номер в сети класса C с минимально возможным номером сети

$11000000.00000000.00000000.00000001 = 192.0.0.1$

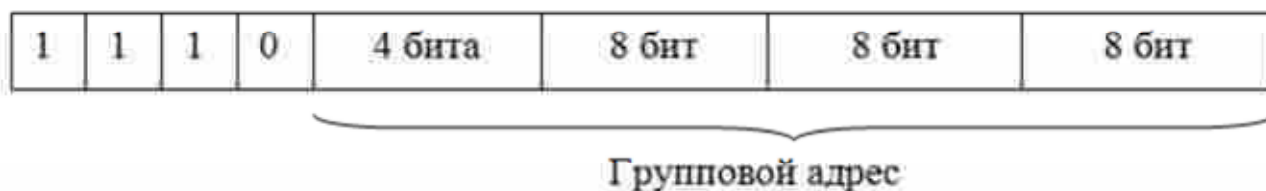
192 0 0 1

Узел имеет максимально возможный номер в сети класса C с максимально возможным номером сети

$11011111.11111111.11111111.11111110 = 223.255.255.254$

223 255 255 254

Класс D



Адреса сетей класса D начинаются с 1110 в двоичной записи, или с 224 в десятичной записи, они имеют номера от 224.0.0.0 до 239.255.255.255 ($11100000.00000000.00000000.00000000 = 224.0.0.0$, $11101111.11111111.11111111.11111111 = 239.255.255.255$)

Если в пакете указан адрес сети класса D, то его получают все узлы этой сети. Поэтому сети класса D называются сетями multicast – широкоэмитательными сетями

и используются для обращения к группам узлов. Основное назначение multicast - распространение информации по схеме «один- ко- многим». Групповая адресация предназначена для экономичного распространения в Интернет или большой корпоративной сети аудио- или видеопрограмм, предназначенных сразу большой аудитории слушателей или зрителей.

Класс E



Адреса сетей класса E начинаются с 11110 в двоичной записи, или с 240 в десятичной записи, они имеют номера от 240.0.0.0 до 247.255.255.255 (11110000.00000000.00000000.00000000.=240.0.0.0, 111101111.11111111.11111111.11111111= 247.255.255.255). Сети класса E зарезервированы для будущих использований.

Некоторые IP – адреса являются выделенными и трактуются по- особому:

- Все нули – 0.0.0.0 – обозначает адрес данного узла
- Номер сети. Все нули (194.28.0.0) – данная IP- сеть
- Все нули. Номер узла (0.0.0.15) – узел в данной IP- сети
- Все единицы (255.255.255.255) – все узлы в данной IP- сети
- Номер сети. Все единицы (194.28.255.255) – все узлы в указанной IP- сети
- Число 127. единица (127.0.0.1) – «петля». Петля используется при тестировании компьютера, и данные не пересылаются по сети, а направляются на модули верхнего уровня, как будто принятые из сети. Поэтому в сетях запрещается использовать IP- адреса, начинающиеся с 127.

Использование масок в IP- адресации

Основной недостаток использования классов IP- адресов напрямую состоит в том, что если организация имеет несколько сетевых номеров, то все компьютеры вне сети имеют доступ к этим адресам и сеть организации становится прозрачной.

Для устранения указанного недостатка адресное пространство сети разбивается на более мелкие непересекающиеся пространства – подсети (subnet). С каждой из подсетей можно работать как с обычной TCP/IP – сетью.

Разбивка адресного пространства на подсети осуществляется с помощью *масок*.

Маска- это число, которое используется в паре с IP- адресом; двоичная запись маски содержит единицы в тех разрядах, которые должны в IP- адресах

11111111.11111111.10000000.00000000 = 255.255.128.0

В этом случае наложение маски на IP-адрес дает новое число, интерпретируемое как номер сети:

10000010. 00100000. 10000000. 00000000 = 130.32.128.0

Номер узла в этой сети становится 0.0.5.1

Как видно из примера, снабжая IP-адреса маской, можно отказаться от понятий классов адресов и сделать более гибкой систему адресации сетей.

Пример

Пусть в сети работают два компьютера, имеющие два соответствующие IP-адреса: 210.20.30.193 и 210.20.30.70. Для разделения указанных компьютеров в две разные подсети используем маску 255.255.255.192

В двоичной форме маска имеет вид:

11111111. 11111111. 11111111. 11000000
└───┘ └───┘ └───┘ └───┘
255 255 255 192

Двоичный адрес первого компьютера:

11010010. 00010100. 00011110. 11000001
└───┘ └───┘ └───┘ └───┘
210 20 30 193

Двоичный адрес второго компьютера:

11010010. 00010100. 00011110. 01000110
└───┘ └───┘ └───┘ └───┘
210 20 30 70

Накладывая маску на адрес первого компьютера, получим его новый адрес:

11010010. 00010100. 00011110. 11 000001
└───┘ └───┘ └───┘ └───┘
210 20 30 / 6

Подсеть №
3

Накладывая маску на адрес второго компьютера, получим его новый адрес:

11010010. 00010100. 00011110. 01 000110
└───┘ └───┘ └───┘ └───┘
210 20 30 / 6

Подсеть №
1

Таким образом, сеть с помощью маски разбилась на две подсети, номер второго компьютера в подсети стал равным шести.

Следует отметить, что в настоящее время наблюдается дефицит IP- адресов, выделяемых организацией InterNIC. Очень трудно получить адрес класса В и практически невозможно стать обладателем адреса класса А. Если же IP- сеть создана для работы в автономном режиме, без связи с Интернет, то администратор сети сам произвольно назначает номер. Но даже в этой ситуации в стандартах Интернет определены несколько диапазонов адресов, не рекомендуемых для использования в локальных сетях. Эти адреса не обрабатываются маршрутизаторами Интернет ни при каких условиях. Для сетей класса А – это сеть 10.0.0.0, в классе В- это диапазон из 16 номеров сетей 172.16.0.0 – 172.31.0.0, в классе С – это диапазон из 255 сетей – 192.168.0.0 – 192.168.255.0.

Для разрешения проблемы дефицита адресов осуществляется переход на новую версию IP- протокола- протокол IPv6, в котором резко расширяется адресное пространство за счет 16- байтных адресов.

Протокол IPv6, как развитие транспортных средств IP- протокола

Указанный протокол решает принципиальную проблему нехватки IP-адресов посредством использования 128- разрядных адресов вместо 32 – разрядных адресов, благодаря чему адресное пространство расширяется в 296 раз. Результатом этого будет то, что любой житель Земли может получить в свое распоряжение несколько IP- адресов, новое количество адресов позволит подключить к сети свыше 1 квадрильона компьютеров в 1 триллионе сетей.

Адреса в IPv6 – протоколе разделяются на три типа: *обычные, групповые и нечеткие.*

Пакет с обычным адресом передается конкретному адресату, в то время как пакет с групповым адресом доставляется всем членам группы. Пакет с нечетким адресом доставляется только ближайшему члену данной группы.

В IPv6 128 разрядные адреса записываются в виде восьми 16- разрядных целых чисел, разделенных двоеточием. Каждое число представлено шестнадцатеричными цифрами, разделенными двоеточиями. Другими словами, необходимо вводить 32 шестнадцатеричные цифры для задания IP- адреса. IPv6 – адрес может выглядеть так: 501A:0000:0000:00FC:ABCD:3F1F:3D5A.

Переход от традиционных IP- адресов к IPv6 – адресам займет ни один год и старая адресация будет постепенно замещаться новыми программными продуктами и оборудованием, использующим IPv6- протокол.

Среди других новых свойств IPv6 – протокола можно отметить также более рациональную структуру формата заголовка пакета, увеличение производительности маршрутизаторов, работающих с этим протоколом, возможность маркировки потока данных, если их необходимо обрабатывать особым образом, аутентификацию дейтаграмм и д.р.

Протокол ICMP

Межсетевой протокол управляющих сообщений (ICMP - Internet Control Message Protocol) разработан для того, чтобы маршрутизаторы в Интернете сообщали об ошибках или предоставляли информацию о нестандартных условиях работы сети. Он является необходимой частью протокола IP. И обеспечивает обратную связь, оповещение отправителя данных о проблемах, возникающих в коммуникационном оборудовании.

Протокол ICMP - это механизм сообщения об ошибках. Он обеспечивает маршрутизаторам, обнаруживающим ошибки, способ сообщения об ошибке первоначальному источнику. Хотя спецификация протокола определяет допустимые способы использования ICMP и предлагает варианты возможных действий в ответ на ошибки, ICMP не специфицирует полностью действия, которые необходимо предпринять в ответ на все возможные ошибки. Таким образом, ICMP только сообщает о возникших ошибках первоначальному источнику; источник сам должен связать ошибки с конкретными прикладными программами и предпринять действия по исправлению ошибок.

Протокол ICMP выполняет следующие основные функции:

- обмен тестовыми сообщениями для выяснения наличия и активности узлов сети;
- анализ достижимости узлов и сброс пакетов, направленных к недостижимым узлам;
- изменение маршрутов (Redirect);
- уничтожение пакетов с истекшим временем жизни (Time-To-Live);
- синхронизация времени в узлах сети;
- управление трафиком (регулирование частоты отправки пакетов).

С точки зрения уровней протоколов, ICMP является частью сетевого уровня. Но по отношению к IP ICMP протокол более высокого уровня, так как он пользуется услугами IP для доставки своих сообщений. Другими словами, каждое сообщение ICMP передается по сети внутри IP-дейтаграммы.

ICMP-сообщения бывают двух видов: сообщение-запрос и сообщение об ошибке. Сообщение об ошибке тесно связано с породившей его дейтаграммой и всегда содержит заголовок этой IP-дейтаграммы и первые 64 бит ее данных. Это необходимо для того, чтобы узел-источник смог более точно проанализировать причину ошибки, так как все прикладные протоколы стека TCP/IP содержат наиболее важную информацию для анализа в первых 8 байт своего сообщения. Сообщения-запросы передают информацию об определенной сети и об определенном компьютере или их используют для диагностических целей.

IP-пакеты с сообщениями ICMP передаются по сети «на общих основаниях», без приоритетов, поэтому они тоже могут теряться. В загруженной сети они могут

вызвать дополнительную загрузку маршрутизаторов, когда потеря сообщения об ошибке приводит к порождению нового сообщения и т. д., пока канал связи не исчерпает своей пропускной способности. Для того чтобы предотвратить подобные ситуации, в спецификации четко определены правила, руководствуясь которыми компьютер может решить, передавать его ICMP-сообщение или нет.

Правило 1: потеря пакета с ICMP-сообщением никогда не генерирует нового ICMP-сообщения.

Правило 2: сообщения об ошибке никогда не генерируются в ответ на IP-дейтаграммы с широковещательными или групповыми адресами, чтобы не вызвать полную перегрузку сети - широковещательный шторм (broadcast storm).

Правило 3: при повреждении фрагментированной дейтаграммы, ICMP-сообщение отправляют только после первого поврежденного фрагмента (так как источник все равно повторит передачу всей дейтаграммы целиком).

Доставка ICMP-пакетов требует двух уровней инкапсуляции. ICMP-пакеты инкапсулируются внутри IP-дейтаграммы, которая сама передается по каждой физической сети в поле данных кадра (рис. 24).



Рисунок 24 – Два уровня инкапсуляции сообщения ICMP

Несмотря на то, что сообщения ICMP инкапсулируются и посылаются, используя IP, ICMP не считают протоколом более высокого уровня - он является необходимой частью IP. Протокол IP необходим для транспортировки сообщений ICMP, так как им, чтобы достичь своего конечного назначения, надо пересечь несколько физических сетей. Поэтому, они не могут быть доставлены только с помощью физической передачи.

Формат ICMP-пакетов. Хотя каждое сообщение ICMP имеет свой собственный формат, все они начинаются с трех одинаковых полей: 8-битового целочисленного поля «Тип», идентифицирующего сообщение, 8-битового поля «Код», обеспечивающего более точную информацию о типе сообщения, и 16-битового поля «Контрольная сумма» (рис. 25). Помимо того, сообщения ICMP, сообщающие об ошибках, всегда включают заголовок и первые 64 бит данных дейтаграммы, вызвавшей ошибку. Это необходимо для того, чтобы узел-от-правитель смог более точно проанализировать причину ошибки, так как все протоколы прикладного уровня стека TCP/IP содержат наиболее важную информацию для анализа в первых 64 бит своих сообщений.

0	8	16	31
Тип (8 или 0)	Код (0)	Контрольная сумма	
Идентификатор		Последовательный номер	
Необязательные данные			
...			

Рисунок 25 – Формат пакета ICMP

Сетевые программы распознают ICMP-сообщения по двум признакам: значению поля «Тип» и значению поля «Код». Контрольная сумма вычисляется не только для ICMP-заголовка, но и для всего сообщения.

Таблица 4 – Типы сообщений ICMP

Тип сообщения ICMP	Описание
0	Ответ на эхо (Echo Reply)
3	Узел назначения недостижим (Destination Unreachable)
4	Подавление источника (Source Quench)
5	Перенаправление маршрута (Redirect)
8	Запрос эха (Echo Request)
9	Информация о маршрутизаторах (Router Advertisement)
10	Регистрация маршрутизатора (Router Solicitation)
11	Лимит времени для дейтаграммы превышен (Time Exceeded for a Datagram)
12	Проблема с параметром пакета (Parameter Problem on a Datagram)
13	Запрос метки времени (Timestamp Request)
14	Ответ для метки времени (Timestamp Reply)
15	Запрос информации (не действует)
16	Ответ на запрос информации (не действует)
17	Запрос маски адреса (Address Mask Request)
18	Ответ на запрос маски адреса (Address Mask Reply)

Типы сообщений ICMP представлены в табл. 4. Рассмотрим каждое из этих сообщений и его формат подробнее.

Тестирование достижимости места назначения. Протоколы TCP/IP обеспечивают средства, помогающие сетевым администраторам или пользователям идентифицировать проблемы в сети. Пользователь в качестве одного из широко используемых средств отладки применяют команду, которая вызывает сообщения ICMP запроса эха и ответа эха. Компьютер или маршрутизатор посылает сообщение запроса эха указанному месту назначения. Любая машина, получившая запрос эха, генерирует ответ на эхо и возвращает его первоначальному отправителю. Этот запрос содержит необязательную область данных; ответ содержит копию данных, посланных в запросе. Запрос эха и связанный с ним ответ можно использовать для проверки достижимости назначения и его способности отвечать на запросы. Так как и запрос эха, и ответ на него передаются в IP-дейтаграммах, успешный прием ответа свидетельствует о работоспособности основных частей транспортной системы. В-первых, программное обеспечение IP на машине источника выполнило

маршрутизацию дейтаграммы. Во-вторых, промежуточные маршрутизаторы между источником и получателем работоспособны и корректно маршрутизируют дейтаграммы. В-третьих, машина получателя работает (по крайней мере, она обрабатывает прерывания) и программное обеспечение, как IP, так и ICMP, выполняет свои функции. И, наконец, таблицы маршрутов в маршрутизаторах на всем обратном пути корректны.

Во многих системах команда, которую пользователи вызывают для отправки запроса эха ICMP, называется ping. Усложненные версии этой программы посылают серии запросов эха ICMP, принимают ответы и выдают статистику о потерях дейтаграмм. Они позволяют пользователю указывать длину посылаемых данных и интервалы времени между запросами. Менее сложные версии просто посылают запрос эха ICMP и ждут ответа.

Формат сообщения запроса эха и ответа эха. Средства для тестирования достижимости узлов сети представляют собой очень простой эхо-протокол, включающий обмен двумя типами сообщений: эхо-запрос и эхо-ответ. Компьютер или маршрутизатор посылают по интересети эхо-запрос, в котором указывают IP-адрес узла, достижимость которого нужно проверить. Узел, получающий эхо-запрос, формирует и отправляет эхо-ответ и возвращает сообщение узлу - отправителю запроса. В запросе могут содержаться некоторые данные, которые должны быть возвращены в ответе.

Рис. 25 иллюстрирует формат сообщений запроса эха и ответа на запрос эха. Поле «Необязательные данные» имеет переменную длину и содержит данные, которые надо вернуть отправителю. Ответ на эхо всегда возвращает те же самые данные, что были получены им в запросе. Поля «Идентификатор» и «Последовательный номер» отправитель использует для проверки соответствия ответов запросам. Значение поля «Тип» определяет, является ли сообщение запросом (8) или ответом (0).

0	8	16	31
Тип (3)	Код (0-5)	Контрольная сумма	
Не используется (должно быть нулевым)			
Префикс дейтаграммы (Заголовок плюс первые 8 байт дейтаграммы)			
...			

Рисунок 26 – Формат сообщения о недостижимости назначения

Сообщения о недостижимости назначения. Когда маршрутизатор не может доставить IP-дейтаграмму, он посылает сообщение «назначение недостижимо» первоначальному отправителю, используя формат, приведенный на рис. 26. Поле «Код» в сообщении о недостижимости назначения содержит целое число, которое описывает причину. Возможные значения представлены в табл. 5.

Таблица 5 – Коды сообщений о недостижимости

Код сообщения	Пояснения
0	Сеть недостижима
1	Компьютер недостижим
2	Протокол недостижим
3	Порт недостижим
4	Необходима фрагментация
5	Ошибка при маршрутизации источника
6	Сеть назначения неизвестна
7	Компьютер назначения неизвестен
8	Компьютер источника изолирован
9	Взаимодействие с сетью назначения административно запрещено
10	То же с компьютером назначения
И	Сеть недостижима из-за класса обслуживания
12	Компьютер недостижим из-за класса обслуживания

Хотя протокол ЕР является механизмом ненадежной доставки, дейтаграммы не уничтожаются просто так. Всякий раз, когда ошибка мешает маршрутизатору произвести маршрутизацию или доставку дейтаграммы, маршрутизатор посылает сообщение о недостижимости назначения его источнику, а затем уничтожает дейтаграмму. Ошибки недостижимости сети обычно являются следствием ошибок маршрутизации; ошибки недостижимости компьютера - следствие ошибок при доставке.

Назначения могут быть недостижимыми из-за того, что оборудование было временно неработоспособно, отправитель указал несуществующий адрес назначения или (в редких случаях) у маршрутизатора не указано маршрута к сети назначения. Необходимо отметить, что не все подобные ошибки можно обнаружить.

Если дейтаграмма содержит опцию маршрутизации источника с некорректным маршрутом, то это может привести к появлению сообщения об ошибке маршрутизации источника. Если шлюзу нужно фрагментировать дейтаграмму, но установлен бит «не фрагментировать», то шлюз посылает сообщение «требуется фрагментация» обратно источнику.

Управление потоком дейтаграмм и переполнение сети. Так как IP- протокол не устанавливает соединения, то маршрутизаторы не могут резервировать память или коммуникационные ресурсы до получения дейтаграмм. В результате, трафик может вызвать перегрузку маршрутизаторов, ситуацию, называемую переполнением сети (congestion). Переполнение сети происходит по двум совершенно разным причинам. Во-первых, высокоскоростной компьютер может генерировать трафик быстрее, чем сеть может передавать его. Например, представим суперкомпьютер, генерирующий межсетевой трафик. Дейтаграммам, посылаемым им, может потребоваться передача, в конечном счете, по медленной глобальной сети (WAN), хотя сам суперкомпьютер может быть присоединен к высокоскоростной LAN. Переполнение будет возникать в маршрутизаторе, присоединенном к глобальной сети, так как дейтаграммы будут

прибывать быстрее, чем их можно послать. Во-вторых, если большому числу компьютеров одновременно нужно посылать дейтаграммы через один маршрутизатор, этот маршрутизатор может оказаться переполненным, хотя ни один источник в отдельности не вызывает эту проблему.

Когда дейтаграммы прибывают на шлюз или маршрутизатор быстрее, чем он успевает их обрабатывать, он временно ставит их в очередь в своей памяти. Если эти дейтаграммы создают небольшую пиковую нагрузку при передаче дейтаграмм, то такая буферизация решает проблему. Если же трафик продолжает поступать, то, в конечном счете, маршрутизатор или шлюз займет всю память под очередь и вынужден будет удалять новые прибывающие дейтаграммы. Тогда машина для выхода из состояния переполнения использует сообщения о подавлении источника.

Сообщение о подавлении источника требует от источника уменьшить скорость передачи дейтаграмм. Обычно переполненные маршрутизаторы посылают по одному сообщению о подавлении источника на каждую удаляемую дейтаграмму или используют более сложные технологии выхода из переполнения. Формат подавления источника представлен на рис. 27. Помимо обычных полей ICMP «Тип», «Код» и «Контрольная сумма» и неиспользуемого 32-битового поля, сообщения о подавлении источника имеют поле, содержащее префикс дейтаграммы. Как и в других сообщениях об ошибках ICMP поле префикса дейтаграммы содержит префикс дейтаграммы, вызвавшей этот запрос подавления источника.

0	8	16	31
Тип (4)	Код (0)	Контрольная сумма	
Не используется (должно быть нулевым)			
Префикс дейтаграммы (Заголовок плюс первые 8 байт дейтаграммы)			
...			

Рисунок 26 – Формат сообщения о подавлении источника ICMP

Сообщения ICMP, вызывающего эффект, обратный подавлению источника, не существует. Вместо этого, компьютер, принявший сообщения о подавлении источника от некоторой машины, снижает скорость, с которой он посылает ей дейтаграммы. Это происходит до тех пор, пока к нему не перестанут приходить сообщения о подавлении источника. Затем он постепенно увеличивает скорость пока снова не получит сообщения о подавлении источника.

Перенаправление маршрута. Маршрутные таблицы у компьютеров обычно статические, так как их конфигурирует администратор сети, а у маршрутизаторов - динамические, формируемые автоматически с помощью протоколов обмена маршрутной информацией. Поэтому с течением времени при изменении топологии сети маршрутные таблицы компьютеров могут устаревать.

При изменении топологии сети таблицы маршрутизации в маршрутизаторе или компьютере могут стать некорректными. Изменение может быть временным (например, нужно заменить неисправное оборудование) или постоянным (например,

когда в межсетевое взаимодействие включается новая сеть). Маршрутизаторы периодически обмениваются информацией о маршрутизации, чтобы отслеживать изменения в сети и своевременно менять маршруты. Для корректировки поведения компьютеров маршрутизатор может использовать сообщение протокола ICMP, называемое «перенаправлением» (Redirect), запрашивающее изменение маршрута в таблице маршрутизации компьютера.

Механизм перенаправления протокола ICMP позволяет компьютерам содержать в конфигурационном файле только IP-адреса его локальных маршрутизаторов. С помощью сообщений о перенаправлении маршрутизаторы будут сообщать компьютеру всю необходимую ему информацию о том, какому маршрутизатору следует отправлять пакеты для той или иной сети назначения, т. е. маршрутизаторы передадут компьютеру нужную ему часть их таблиц маршрутизации.

Преимуществом схемы перенаправления ICMP является ее простота: она позволяет компьютеру знать вначале адрес только одного маршрутизатора в локальной сети. Этот начальный маршрутизатор возвращает сообщение ICMP о перенаправлении всякий раз, когда компьютер посылает дейтаграмму, для которой существует лучший маршрут. Таблица маршрутизации компьютера останется маленькой, но содержит оптимальные маршруты для всех используемых назначений.

Сообщения о перенаправлении, тем не менее, не решают проблему распространения информации о маршрутах полностью, так как они предназначены только для взаимодействия между маршрутизатором и компьютером в одной физической сети. Каждое сообщение о перенаправлении содержит 32-битовое поле «IP-адрес маршрутизатора» и поле «Префикс дейтаграммы», как это показано на рис. 27.

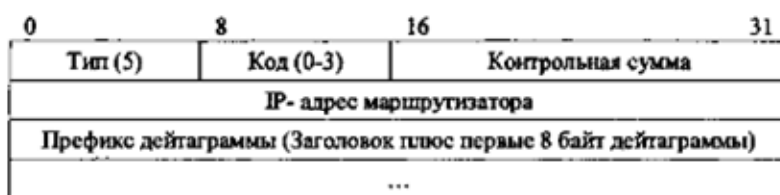


Рисунок 27 – Формат сообщения о перенаправлении ICMP

Поле «Межсетевой адрес маршрутизатора» содержит IP-адрес маршрутизатора, который должен использовать компьютер при отправлении дейтаграммы к назначению, указанному в заголовке дейтаграммы. Поле «Префикс дейтаграммы» содержит заголовок IP и следующие 8 байт дейтаграммы, которая привела к появлению этого сообщения. Поэтому компьютер, принимающий сообщение о перенаправлении ICMP, должен выделить адрес назначения дейтаграммы из префикса дейтаграммы. Поле «Код» в сообщении о перенаправлении ICMP более конкретно указывает, как интерпретировать адрес

назначения, при этом значения имеют следующий смысл: 0 - перенаправление дейтаграмм для этой сети (устарело), 1 - перенаправление дейтаграмм для этого компьютера, 2 - перенаправление дейтаграмм для этого типа сервиса и сети, 3 - перенаправление дейтаграмм для этого типа сервиса и компьютера. Напомним, что каждый заголовок IP указывает тип сервиса, используемого при маршрутизации. Как правило, маршрутизаторы посылают запросы переназначения ICMP только на компьютеры, а не на другие маршрутизаторы.

Изменение маршрута является одной из наиболее интересных функций протокола ICMP - по существу, это один из механизмов автоматической оптимизации доставки пакетов и адаптации сетей TCP/IP к изменениям топологии.

Запросы «Информация о маршрутизаторах» (типы 9 и 10).

Информация о маршрутизации находится в местных конфигурационных файлах и загружается оттуда при запуске компьютера. Чтобы таблица маршрутизации не содержала устаревших данных она обновляется динамически. ICMP- протокол реализует один из способов ее обновления.

Существует 2 типа сообщений маршрутизаторов:

- 9 - информация о маршрутизации;
- 10-регистрация маршрутизатора.

Всякий раз, когда компьютер запускают, он генерирует сообщения о регистрации. В ответ маршрутизаторы, находящиеся в той же локальной сети, посылают сообщения с информацией о маршрутизации, позволяющие правильно сконфигурировать маршрутную таблицу.

Формат сообщения «Информация о маршрутизации» (тип 9) описан в RFC 1256 (рис. 28).

0	8	16	31
Тип (9)	Код (0)	Контрольная сумма	
Количество адресов	Длина поля адреса	Время существования	
IP-адрес маршрутизатора 1			
Приоритет 1			
IP-адрес маршрутизатора 2			
Приоритет 2			
...			

Рисунок 28 – Формат сообщения «Информация о маршрутизации»

В одном ICMP-сообщении может содержаться описание нескольких адресов, количество которых указано в поле «Количество адресов». Поле «Размер адреса» задает длину адреса в 32-битовых словах. В настоящее время «Длина поля адреса» всегда равна 2.

Поле «Время существования» задает интервал времени, в течение которого информация еще не устарела. Как правило, это 1800 с.

Поле «Приоритет» указывает, какой из адресов следует использовать первым и более интенсивно. Как правило, чем больше значение поля, тем выше приоритет. Маршрутизаторы передают информационные сообщения широковещательно через случайные интервалы времени. Обычно через 450...600 с. Поле «Время существования» можно использовать для уведомления, что данный маршрутизатор выключается. При этом содержимое данного поля устанавливается равным 0.

Формат сообщения «Регистрация» (тип 10) представлен на рис. 29.

0	8	16	31
Тип (10)	Код (0)	Контрольная сумма	
Указатель	Заполняется нулями		

Рисунок 29 – Формат сообщения «Регистрация»

Запрос «Регистрация» передается 3 раза с интервалом 3 с при запуске маршрутизатора и продолжает (при необходимости) передаваться, пока маршрутизатор не получит информационного сообщения с нужной маршрутной информацией.

Обнаружение циклических или слишком длинных путей. Как было отмечено выше для защиты Интернета от перегрузок каждая дейтаграмма имеет счетчик времени жизни TTL (Time-To-Live). Маршрутизатор декрементирует счетчик времени жизни всякий раз, когда он обрабатывает дейтаграмму, и удаляет ее, когда счетчик становится нулевым.

Независимо от того, удалил ли маршрутизатор дейтаграмму из-за обнуления счетчика времени жизни или из-за превышения времени ожидания фрагментов дейтаграммы, он посылает сообщение ICMP «Лимит времени для дейтаграммы превышен» источнику дейтаграммы определенного формата (рис. 30).

0	8	16	31
Тип (10)	Код (0)	Контрольная сумма	
Указатель	Заполняется нулями		

Рисунок 30 – Формат сообщения «Лимит времени для дейтаграмм превышен»

Поле «Код» объясняет причину сообщения: 0 - превышено значение счетчика времени жизни; 1 - превышено время ожидания фрагмента при сборке.

Сообщения о других ситуациях. Когда маршрутизатор или компьютер сталкивается с проблемой, не укладывающейся в рамки описанных сообщений об ошибках ICMP (например, некорректный заголовок дейтаграммы), связанной с дейтаграммой, он посылает сообщение «Проблема с параметром пакета» первоначальному отправителю. Такую ситуацию может вызвать некорректность аргументов опции. Сообщение, формат которого показан на рис. 31, посылается только в том случае, если дейтаграмма должна быть удалена из-за этой ошибки. Для уточнения места ошибки в дейтаграмме отправитель использует поле

«Указатель» в заголовке сообщения для идентификации октета в дейтаграмме, содержащего ошибку.

0	8	16	31
Тип (12)	Код (0-1)	Контрольная сумма	
Указатель	Не используется (должно быть нулевым)		
Префикс дейтаграммы (Заголовок плюс первые 8 байт дейтаграммы)			
...			

Рисунок 31 – Формат сообщения «Проблемы с параметром пакета»

0	8	16	31
Тип (13 или 14)	Код (0)	Контрольная сумма	
Идентификатор		Последовательный номер	
Префикс дейтаграммы (Заголовок плюс первые 8 байт дейтаграммы)			
Временная метка отправителя			
Временная метка приема			
Временная метка передачи			

Рисунок 32 – Формат сообщения «Запрос метки времени»
и «Ответ для метки времени»

Синхронизация часов и оценка времени передачи. Стек протоколов TCP/IP включает несколько протоколов, которые могут использоваться для синхронизации часов. В сети для этого используется несколько технологий. Одна из простейших технологий реализуется сообщениями ICMP для получения значения времени от другой машины. Запрашивающая машина посылает сообщение ICMP «Запрос метки времени» другой машине, ожидая, что вторая машина вернет ей текущее значение времени. Принимающая машина возвращает «Ответ для метки времени» машине, выдавшей запрос. Рис. 32 иллюстрирует формат сообщений запроса и ответа временной метки. Поле «Тип» идентифицирует сообщение как запрос (13) или ответ (14); поля «Идентификатор» и «Последовательный номер» используют источник для связи между ответами и запросами. Оставшиеся поля специфицируют времена, указанные в миллисекундах после полуночи, по Гринвичу. Поле «Временная метка отправителя» заполняет первоначальный отправитель перед передачей пакета, поле «Временная метка приема» заполняется сразу после приема запроса, а поле «Временная метка передачи» - непосредственно перед отправкой ответа.

Компьютеры используют эти три поля временных меток для определения ожидаемого времени передачи между ними и синхронизации своих часов. Так как ответ включает поле «Временная метка отправителя», компьютер может вычислить общее время, требуемое для передачи запроса к назначению, формирования ответа на него и возвращения ответа. Так как ответ содержит как время прихода запроса на удаленную машину, так и время выдачи ответа, компьютер может вычислить время передачи по сети, а на его основе - разницу между своими и удаленными часами. На практике бывает трудно точно оценить время передачи по сети, так как ГР является технологией с негарантированной доставкой, дейтаграммы могут быть потеряны,

задержаны или доставлены не по порядку, что ограничивает полезность сообщений ICMP о временных метках.

Сообщения запроса и ответа информации. Сообщения ICMP запроса информации и ответа информации (тип 15 и 16) в настоящее время устарели и их использовать не рекомендуется. Они предназначались для обнаружения компьютерами своих IP-адресов при загрузке. Сейчас для определения адреса используют протоколы RARP и BOOTP.

DHCP

DHCP (англ. Dynamic Host Configuration Protocol — протокол динамической конфигурации узла) — это сетевой протокол, позволяющий компьютерам автоматически получать IP-адрес и другие параметры, необходимые для работы в сети TCP/IP. Данный протокол работает по модели «клиент-сервер». Для автоматической конфигурации компьютер-клиент на этапе конфигурации сетевого устройства обращается к так называемому серверу DHCP, и получает от него нужные параметры. Сетевой администратор может задать диапазон адресов, распределяемых сервером среди компьютеров. Это позволяет избежать ручной настройки компьютеров сети и уменьшает количество ошибок. Протокол DHCP используется в большинстве сетей TCP/IP.

DHCP является расширением протокола BOOTP, использовавшегося ранее для обеспечения бездисковых рабочих станций IP-адресами при их загрузке. DHCP сохраняет обратную совместимость с BOOTP.

Протокол DHCP предоставляет три способа распределения IP-адресов:

Ручное распределение. При этом способе сетевой администратор сопоставляет аппаратному адресу (для Ethernet сетей это MAC-адрес) каждого клиентского компьютера определённый IP-адрес. Фактически, данный способ распределения адресов отличается от ручной настройки каждого компьютера лишь тем, что сведения об адресах хранятся централизованно (на сервере DHCP), и потому их проще изменять при необходимости.

Автоматическое распределение. При данном способе каждому компьютеру на постоянное использование выделяется произвольный свободный IP-адрес из определённого администратором диапазона.

Динамическое распределение. Этот способ аналогичен автоматическому распределению, за исключением того, что адрес выдаётся компьютеру не на постоянное пользование, а на определённый срок. Это называется арендой адреса. По истечении срока аренды IP-адрес вновь считается свободным, и клиент обязан запросить новый (он, впрочем, может оказаться тем же самым). Кроме того, клиент сам может отказаться от полученного адреса.

Некоторые реализации службы DHCP способны автоматически обновлять записи DNS, соответствующие клиентским компьютерам, при выделении им новых адресов. Это производится при помощи протокола обновления DNS.

Помимо IP-адреса, DHCP также может сообщать клиенту дополнительные параметры, необходимые для нормальной работы в сети. Эти параметры называются опциями DHCP.

Некоторыми из наиболее часто используемых опций являются:

- IP-адрес маршрутизатора по умолчанию;
- маска подсети;
- адреса серверов DNS;
- имя домена DNS.

Некоторые поставщики программного обеспечения могут определять собственные, дополнительные опции DHCP.

Все сообщения протокола DHCP разбиваются на поля, каждое из которых содержит определённую информацию (рис.33). Все поля, кроме последнего (поля опций DHCP), имеют фиксированную длину.

Поле	Описание
op	Тип сообщения (1 = BOOTREQUEST, 2 = BOOTREPLY)
htype	Тип адреса оборудования
hlen	Длина адреса оборудования
hops	Используется ретранслирующим агентом
xid	Идентификатор транзакции между сервером и клиентом
secs	Время с момента выдачи DHCPREQUEST или начала обновления конфигурации
flags	Флаги (первый бит маркирует широковещательные сообщения)
ciaddr	IP-адрес клиента
yiaddr	<Ваш> (клиентский) IP-адрес
siaddr	IP-адрес следующего сервера, участвующего в загрузке
giaddr	IP-адрес ретранслирующего агента
chaddr	<Аппаратный> адрес клиента
sname	Хост-имя сервера (опция)
file	Имя загрузочного файла
options	Поле дополнительных параметров

Рисунок 33 – Поля DHCP

Протокол DHCP является клиент-серверным, то есть в его работе участвуют клиент DHCP и сервер DHCP. Передача данных производится при помощи протокола UDP, при этом сервер принимает сообщения от клиентов на порт 67 и отправляет сообщения клиентам на порт 68.

Работа протокола DHCP базируется на классической схеме клиент-сервер. В роли клиентов выступают компьютеры сети, стремящиеся получить IP-адреса в так называемую аренду (lease), а DHCP-серверы выполняют функции диспетчеров, которые выдают адреса, контролируют их использование и сообщают клиентам требуемые параметры конфигурации. Сервер поддерживает пул свободных адресов и, кроме того, ведет собственную регистрационную базу данных. Взаимодействие

DHCP-серверов со станциями-клиентами осуществляется путем обмена сообщениями.



Рисунок 34 – Формат сообщения DHCP (в скобках - размер поля в байтах)

DHCP разрабатывался как непосредственное расширение BOOTP и именно в таком качестве воспринимается BOOTP-клиентами. Этому обстоятельству в первую очередь способствует формат сообщений DHCP, во многом совпадающий с форматом, который применяется протоколом-предшественником и определен в документе RFC 951 (рис. 34).

Сравнивая протоколы BOOTP и DHCP, нельзя не отметить появления в DHCP новых услуг. Во-первых, в этом протоколе предусмотрен механизм автоматической выдачи IP-адресов во временное пользование с возможностью их последующего присвоения новым клиентам. Во-вторых, клиент может получить от сервера все параметры конфигурации, которые ему необходимы для успешного функционирования в IP-сети.

Указанные отличия потребовали частичного расширения формата сообщений. Так, в нем появилось отдельное поле идентификатора клиента, сделана более прозрачной интерпретация адреса сервера (поле siaddr), переменный размер получило поле options, используемое, в частности, для передачи параметров конфигурации (его длина обычно находится в диапазоне 312-576 байт, хотя возможно и дополнительное расширение этого поля за счет полей sname и file).

В роли транспортного протокола для обмена DHCP-сообщениями выступает UDP. При отправке сообщения с клиента на сервер используется 67-й порт DHCP-сервера, при передаче в обратном направлении - 68-й. Эти номера портов, как и схожая структура сообщений, обеспечивают обратную совместимость DHCP с BOOTP. Конкретные процедуры взаимодействия клиентов и серверов BOOTP и DHCP регламентирует документ RFC 1542.

Выдача адреса в аренду производится по запросу клиента. DHCP-сервер (или группа серверов) гарантирует, что выделенный адрес до истечения срока его аренды не будет выдан другому клиенту; при повторных обращениях сервер старается предложить клиенту адрес, которым тот пользовался ранее. Со своей стороны, клиент может запросить пролонгацию срока аренды адреса либо, наоборот, досрочно отказаться от него. Протоколом предусмотрена также выдача IP-адреса в

неограниченное пользование. При острой нехватке адресов сервер может сократить срок аренды адреса по сравнению с запрошенным.

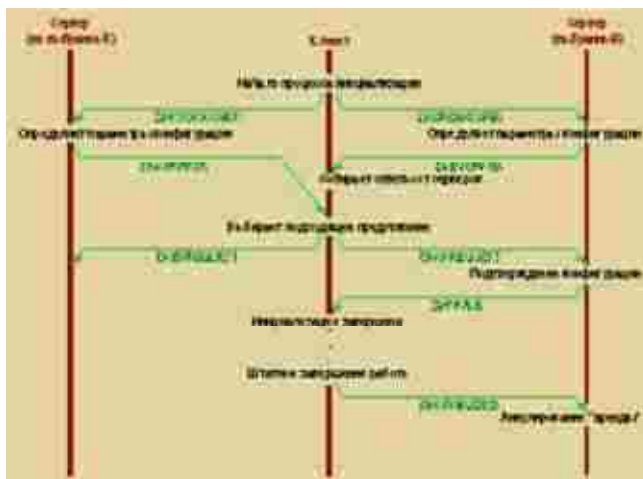


Рисунок 35 – Последовательность событий при выделении IP-адреса

Выдача нового адреса. Последовательность событий в этом случае такова (рис. 35).

1. Клиент посылает в собственную физическую подсеть широковещательное сообщение DHCPDISCOVER, в котором могут указываться устраивающие клиента IP-адрес и срок его аренды. Если в данной подсети DHCP-сервер отсутствует, сообщение будет передано в другие подсети ретранслирующими агентами протокола BOOTP (они же вернут клиенту ответные сообщения сервера).

2. Любой из DHCP-серверов может ответить на поступившее сообщение DHCPDISCOVER сообщением DHCPOFFER, включив в него доступный IP-адрес (yiaddr) и, если требуется, параметры конфигурации клиента. На этой стадии сервер не обязан резервировать указанный адрес. В принципе, он имеет право предложить его другому клиенту, также отправившему запрос DHCPDISCOVER. Тем не менее спецификации RFC 2131 рекомендуют серверу без необходимости не применять подобную тактику, а кроме того, убедиться (например, выдав эхо-запрос ICMP) в том, что предложенный адрес в текущий момент не используется каким-либо из компьютеров сети.

3. Клиент не обязан реагировать на первое же поступившее предложение. Допускается, чтобы он дождался откликов от нескольких серверов и, остановившись на одном из предложений, отправил в сеть широковещательное сообщение DHCPREQUEST. В нем содержатся идентификатор выбранного сервера и, возможно, желательные значения запрашиваемых параметров конфигурации.

Не исключено, что клиента не устроит ни одно из серверных предложений. Тогда вместо DHCPREQUEST он снова выдаст в сеть запрос DHCPDISCOVER, а серверы так и не узнают, что их предложения отклонены. Именно по этой причине сервер не обязан резервировать помещенный в DHCPOFFER адрес.

Если в процессе ожидания серверных откликов на DHCPDISCOVER достигнут тайм-аут, клиент выдает данное сообщение повторно.

4. Присутствующий в сообщении DHCPREQUEST идентификатор позволяет соответствующему DHCP-серверу убедиться в том, что клиент принял именно его предложение. В ответ сервер отправляет подтверждение DHCPACK, содержащее значения требуемых параметров конфигурации, и производит соответствующую запись в базу данных.

Если к моменту поступления сообщения DHCPREQUEST предложенный адрес уже <ушел> к другому клиенту (например, первая станция слишком долго <размышляла> над поступившими предложениями), сервер отвечает сообщением DHCPNACK.

5. Получив сообщение DHCPACK, клиент обязан убедиться в уникальности IP-адреса (средствами протокола ARP) и зафиксировать суммарный срок его аренды. Последний рассчитывается как время, прошедшее между отправкой сообщения DHCPREQUEST и приемом ответного сообщения DHCPACK, плюс срок аренды, указанный в DHCPACK.

Обнаружив, что адрес уже используется другой станцией, клиент обязан отправить серверу сообщение DHCPDECLINE и не ранее чем через 10 с начать всю процедуру снова. Процесс конфигурирования возобновляется и при получении серверного сообщения DHCPNACK.

При достижении тайм-аута в процессе ожидания серверных откликов на сообщение DHCPREQUEST клиент выдает его повторно.

6. Для досрочного прекращения аренды адреса клиент отправляет серверу сообщение DHCPRELEASE.

Приведенная последовательность действий заметно упрощается, если станция-клиент желает повторно работать с IP-адресом, который когда-то уже был ей выделен. В этом случае первым отправляемым сообщением является DHCPREQUEST, в котором клиент указывает прежде использовавшийся адрес. В ответ он может получить сообщение DHCPACK или DHCPNACK (если адрес занят либо клиентский запрос является некорректным, например из-за перемещения клиента в другую подсеть). Обязанность проверить уникальность IP-адреса опять-таки возлагается на клиента.

Выбор адреса DHCP-сервером. Если на момент получения запроса DHCPDISCOVER сервер не располагает свободными IP-адресами, он может направить уведомление о возникшей проблеме администратору. В противном случае при выборе адреса обычно применяется следующий алгоритм. Клиенту выделяется адрес, записанный за ним в данный момент. Если это невозможно, сервер предложит адрес, которым пользовался клиент до окончания срока последней аренды (при условии, что данный адрес свободен), либо адрес, запрошенный самим клиентом при помощи соответствующей опции (опять же, если адрес не занят). Наконец, в том случае, когда все предыдущие варианты не

проходят, новый адрес выбирается из пула доступных адресов с учетом подсети, из которой поступил клиентский запрос.

Заметим, что исходя из определенной сетевым администратором политики сервер может выдать клиенту адрес, отличающийся от запрошенного (даже при доступности последнего), вообще отказать в предоставлении адреса или предложить адрес, относящийся к другой подсети. Более того, DHCP-сервер вообще не обязан реагировать на каждый поступивший запрос DHCPDISCOVER. Это предоставляет администратору возможность контролировать доступ к сети, например разрешив серверу отвечать только тем клиентам, которые предварительно зарегистрировались с помощью специальной процедуры.

Истечение срока аренды. По мере того как срок аренды подходит к концу, клиент может завершить работу с данным адресом, отправив на DHCP-сервер сообщение DHCPRELEASE, либо заблаговременно запросить продление срока аренды. В первом случае возвращение в сеть потребует выполнения всей процедуры инициализации заново. Во втором - станция продолжит функционировать в сети без видимого замедления работы пользовательских приложений.

При пролонгировании аренды клиент проходит два состояния - обновления адреса (RENEWING) и обновления конфигурации (REBINDING). Первое наступает примерно на половине срока аренды адреса (так называемый момент T1), второе - по истечении приблизительно 7/8 полного времени аренды (момент T2); для рассинхронизации процессов реконфигурирования разных клиентов значения этих временных меток рандомизируются с помощью случайной добавки.

В момент T1 клиент отправляет DHCP-серверу, выдавшему адрес, сообщение DHCPREQUEST с просьбой продлить срок аренды. Получив положительный ответ (DHCPACK), клиент пересчитывает срок аренды и продолжает работу в обычном режиме. Клиент ожидает прихода ответа от сервера в течение $(T2 - t)/2$ с (при условии, что это значение не меньше 60 с), где t - время отсылки последнего сообщения DHCPREQUEST, после чего отправляет данное сообщение повторно.

Если ответ от сервера не поступил к моменту T2, клиент переходит в состояние REBINDING и передает уже широковещательное сообщение DHCPREQUEST со своим текущим сетевым адресом. В этом случае моменты повторных выдач запросов DHCPREQUEST рассчитываются аналогично предыдущему случаю, только вместо T2 фигурирует время окончания срока аренды.

Не исключено, однако, что ответ DHCPACK не придет до окончания срока аренды. Тогда клиент обязан немедленно прекратить выполнение любых сетевых операций и заново начать процесс инициализации. Если запоздавший ответ DHCPACK все-таки поступит, клиенту рекомендуется сразу же возобновить работу под прежним адресом.

Параметры конфигурации

Хранение параметров сетевой конфигурации станций-клиентов является второй услугой, предоставляемой DHCP-сервером. В создаваемой базе данных на

каждого клиента заводится отдельная запись с уникальным ключом-идентификатором и строкой конфигурационных параметров.

Роль идентификатора может играть пара <номер подсети IP, аппаратный адрес>, которая позволит использовать аппаратный адрес сразу в нескольких подсетях, либо пара <номер подсети IP, имя хост-компьютера>, позволяющая серверу взаимодействовать с клиентом, перемещенным в другую подсеть.

Что касается собственно параметров конфигурации, то их набор, поддерживаемый протоколом DHCP, определен в спецификациях RFC 1122, 1123, 1196 и 1256. В него входят выданный адрес, срок его аренды, назначавшиеся ранее адреса, а также максимальный размер реассемблируемого пакета, перечень фильтров для нелокальной маршрутизации от источника, адрес, используемый в широковещательных пакетах, параметры статических маршрутов и т.д. Впрочем, из всей совокупности допустимых параметров (а их более 30) в процессе инициализации могут передаваться только те, которые действительно необходимы для работы клиента либо определяются спецификой конкретной подсети.

Редукция объема передаваемых сведений о конфигурации достигается двумя способами. Во-первых, для большей части параметров в упомянутых выше документах RFC определены значения, принимаемые по умолчанию. Клиент будет использовать их, если в сообщении, поступившем от сервера, какие-то параметры опущены. Во-вторых, отправляя сообщение DHCPDISCOVER или DHCPREQUEST, клиентская станция может явно указать в нем параметры, значения которых она хотела бы получить.

Очевидно, что в обоих случаях передача параметров конфигурации осуществляется в ходе основной процедуры выделения IP-адреса. Возможен, однако, случай, когда клиент уже имеет IP-адрес (например, он был задан вручную). Тогда он может выдать сообщение DHCPINFORM*, содержащее уже имеющийся адрес и запрос об отдельных параметрах конфигурации. Получив это сообщение, DHCP-сервер проверяет правильность адреса клиента (но не наличие аренды) и направляет ему сообщение DHCPACK с требуемыми параметрами конфигурации.

Отметим одно логическое противоречие, с которым связано применение протокола DHCP. Алгоритм выделения IP-адреса компьютеру сети предполагает, что установленное на нем программное обеспечение TCP/IP в состоянии воспринимать адресованные ему посредством <аппаратного> адреса IP-пакеты и транслировать их на IP-уровень еще до того, как станция получит свой IP-адрес, а сами средства TCP/IP будут полностью сконфигурированы. Такая возможность, очевидно, существует не всегда. Для работы с клиентами, не способными корректно обрабатывать одноадресные IP-дейтаграммы, используется поле flags. Такие клиенты должны установить первый бит данного поля в единичное значение, тем самым указав серверу на необходимость отправки в соответствующую подсеть только широковещательных сообщений.

Недостатки DHCP

Освобождая сетевых администраторов от множества рутинных операций, DHCP оставляет нерешенными ряд проблем, которые рано или поздно могут возникнуть в реальной сетевой среде.

К недостаткам этого протокола прежде всего следует отнести крайне низкий уровень информационной безопасности, что обусловлено непосредственным использованием протоколов UDP и IP. В настоящее время не существует практически никакой защиты от появления в сети несанкционированных DHCP-серверов, способных рассылать клиентам ошибочную или потенциально опасную информацию - некорректные или уже задействованные IP-адреса, неверные сведения о маршрутизации и т.д. И наоборот, клиенты, запущенные с неблагоприятными целями, могут извлекать конфигурационные сведения, предназначенные для <законных> компьютеров сети, и тем самым оттягивать на себя значительную часть имеющихся ресурсов. Понятно, что возможности административного ограничения доступа, о которых говорилось выше, не способны закрыть эту брешь в системе информационной безопасности.

По мнению некоторых экспертов, в настоящее время DHCP недостаточно отказоустойчив. Протоколу явно недостает механизма активного уведомления клиентов об экстремальных ситуациях (например, о систематической нехватке адресов) и серверного подтверждения об освобождении адреса, иногда в сети наблюдаются всплески числа запросов на повторное использование адресов и т.д. Впрочем, работа над протоколом еще не завершена, и не исключено, что некоторые недостатки будут устранены в последующих редакциях.

Тема 13. Основные принципы маршрутизации

Маршрутизаторы в сетевых технологиях

Объединение нескольких локальных сетей в глобальную (распределенную, составную) WAN-сеть происходит с помощью устройств и протоколов сетевого Уровня 3 семиуровневой эталонной модели или уровня межсетевого взаимодействия четырехуровневой модели TCP/IP. Если LAN объединяют рабочие станции, периферию, терминалы и другое сетевое оборудование в одной аудитории или в одном здании, то WAN обеспечивают соединение LAN на широком географическом пространстве. В составную распределенную сеть (internetwork, internet) входят как локальные сети и подсети (subnet), так и отдельные пользователи. Устройствами, объединяющими LAN в составную сеть, являются:

- маршрутизаторы (routers);
- модемы;
- коммуникационные серверы.

Наиболее распространенными устройствами межсетевого взаимодействия сетей, подсетей и устройств являются маршрутизаторы. Они представляют собой

специализированные компьютеры для выполнения специфических функций сетевых устройств. В лекции 4 было показано, что маршрутизаторы используются, чтобы сегментировать локальную сеть на широковещательные домены, т. е. являются устройствами LAN, но они применяются и как устройства формирования глобальных сетей. Поэтому маршрутизаторы имеют как LAN-, так и WAN-интерфейсы. Маршрутизаторы используют WAN-интерфейсы, чтобы связываться друг с другом, и LAN-интерфейсы – для связи с узлами (компьютерами), например через коммутаторы. Поэтому маршрутизаторы являются устройствами как локальных, так и глобальных сетей. Маршрутизаторы являются также основными устройствами больших корпоративных сетей.

На рис. 36 приведен пример того, как маршрутизаторы А, В и С объединяют несколько локальных сетей (локальные сети № 1, № 2, № 3) в распределенную (составную) сеть. Поэтому маршрутизаторы имеют интерфейсы как локальных, так и глобальных соединений. К локальным сетям, созданным на коммутаторах, маршрутизатор присоединен через интерфейсы, которые на рис. 6.1 обозначены через F0/1, что означает: интерфейс Fast Ethernet, слот 0, номер 1. Глобальные соединения на рис. 6.1 представлены последовательными или серийными (serial) интерфейсами S0/1, S0/2. Через такой же последовательный интерфейс реализовано соединение составной сети с сетью Интернет (Internet). Подобная структурная схема, включающая несколько последовательно соединенных маршрутизаторов, характерна для многих корпоративных сетей.

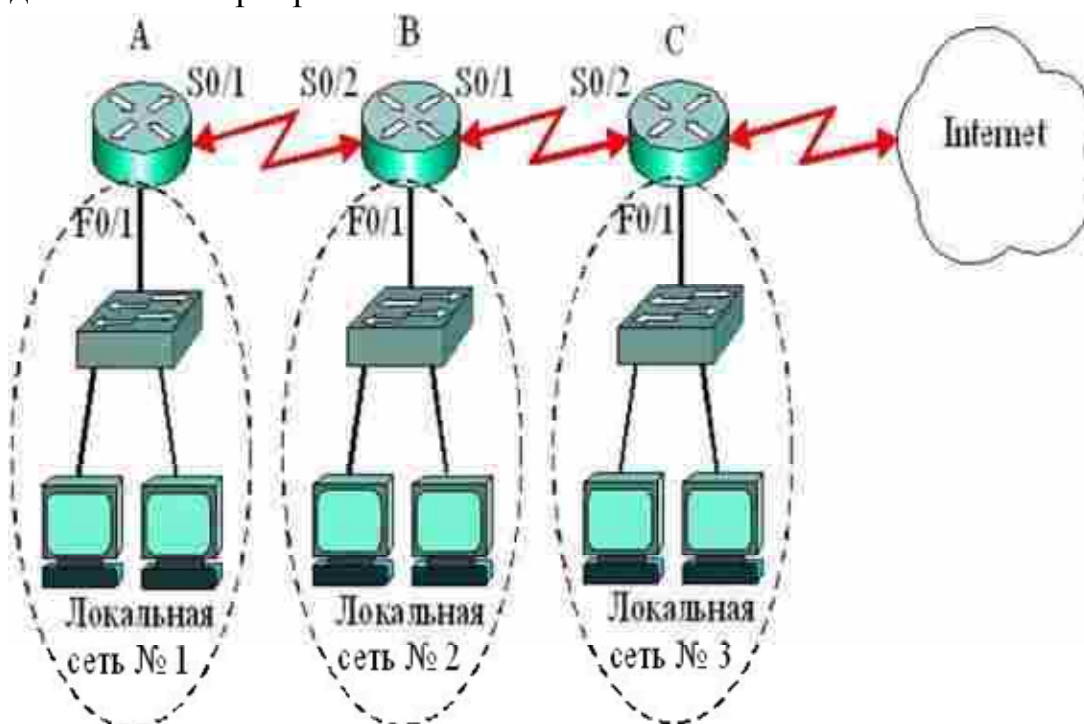


Рисунок 36 – Составная сеть на маршрутизаторах

В большинстве случаев соединение маршрутизатора локальной сети с сетью Интернет производится через сеть провайдера. Терминальное (оконечное) оборудование (Data Terminal Equipment – DTE), к которому относится и

маршрутизатор, подсоединяется к глобальной сети (или к сети провайдера) через канальное коммуникационное оборудование (Data Communications Equipment, или Data Circuit-Terminating Equipment, – DCE). Маршрутизатор обычно является оборудованием пользователя, а оборудование DCE предоставляет провайдер. Услуги, предоставляемые провайдером для терминальных устройств DTE, доступны через модем или цифровое устройство согласования с каналом связи (Channel Service Unit / Data Service Unit – CSU/DSU), которые и являются оборудованием DCE (рис. 37). Оборудование DCE является ведущим в паре DCE-DTE, оно обеспечивает синхронизацию и задает скорость передачи данных.

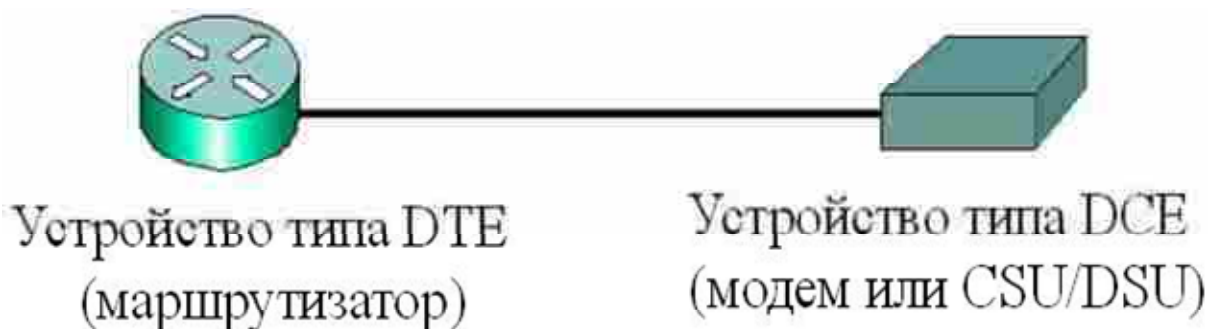


Рисунок 37 – Устройства распределенных сетей

Поскольку маршрутизаторы в распределенных сетях (рис. 36) часто соединяются последовательно, из двух последовательно соединенных серийных интерфейсов маршрутизаторов один должен выполнять роль устройства DCE, а второй – устройства DTE (рис. 38).

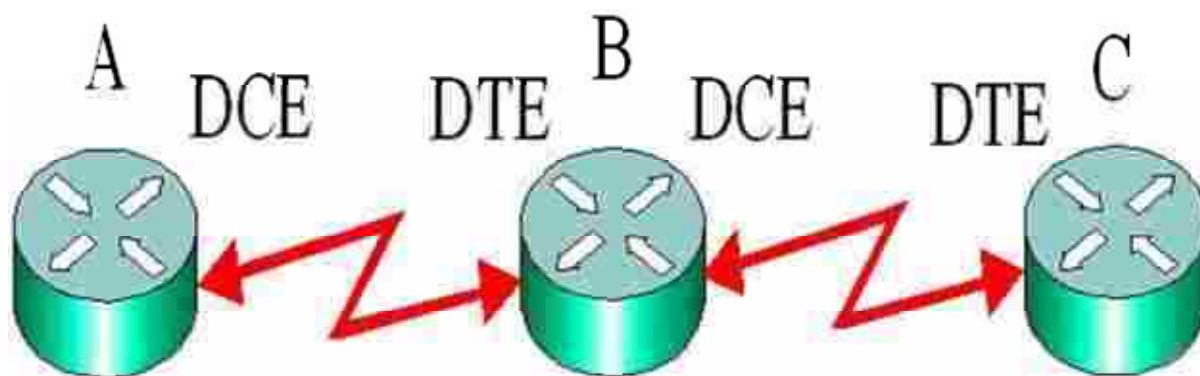


Рисунок 38 – Последовательное соединение маршрутизаторов

Главными функциями маршрутизаторов являются:
 выбор наилучшего пути для пакетов к адресату назначения;
 продвижение (коммутация) принятого пакета с входного интерфейса на соответствующий выходной интерфейс.

Таким образом, маршрутизаторы обеспечивают связь между сетями и определяют наилучший путь пакета данных к сети адресата, причем технологии объединяемых локальных сетей могут быть различными.

Протоколы канального (data link) уровня WAN описывают, как по сети передаются кадры. Они включают протоколы, обеспечивающие функционирование через выделенные соединения "точка-точка" и через коммутируемые соединения. Основными WAN протоколами и стандартами канального уровня являются: High-level Data Link Control (HDLC), Point-to-Point Protocol (PPP), Synchronous Data Link Control (SDLC), Serial Line Internet Protocol (SLIP), X.25, Frame Relay, ATM. Основными протоколами и стандартами физического уровня являются: EIA/TIA-232, EIA/TIA-449, V.24, V.35, X.21, G.703, EIA-530, ISDN, E1, E3, XDSL, SDH (STM-1, STM-4 и др.).

Функционируя на Уровне 3 модели OSI, маршрутизаторы принимают решения, базируясь на сетевых логических адресах (IP-адресах). Для определения наилучшего пути передачи данных через связываемые сети маршрутизаторы строят таблицы маршрутизации и обмениваются сетевой маршрутной информацией с другими маршрутизаторами. Администратор может конфигурировать статические маршруты и поддерживать таблицы маршрутизации вручную. Однако большинство таблиц маршрутизации создается и поддерживается динамически, за счет использования протоколов маршрутизации (routing protocol), которые позволяют маршрутизаторам автоматически обмениваться информацией о сетевой топологии друг с другом.

Функционирование маршрутизаторов происходит под управлением сетевой операционной системы (Internetwork Operation System – IOS), текущая (running) версия которой находится в оперативной памяти RAM (рис. 6.4). Помимо текущей версии IOS оперативная память хранит активный конфигурационный файл (Active Configuration File) и таблицы протоколов динамической маршрутизации, выполняет буферизацию пакетов и поддерживает их очередь, обеспечивает временную память для конфигурационного файла маршрутизатора, пока включено питание.

Загрузка операционной системы IOS в оперативную память обычно производится из энергонезависимой флэш-памяти (Flash), которая является перепрограммируемым запоминающим устройством (ППЗУ). После модернизации IOS она перезаписывается во флэш-память, где может храниться несколько версий. Версию операционной системы можно также сохранять на TFTP-сервере (рис. 39).

Постоянное запоминающее устройство (ПЗУ – ROM) содержит программу начальной загрузки (bootstrap) и сокращенную версию операционной системы, установленную при изготовлении маршрутизатора. Обычно эта версия IOS используется только при выходе из строя флэш-памяти. Память ROM также поддерживает команды для теста диагностики аппаратных средств (Power-On Self Test – POST).

Энергонезависимая (non-volatile) оперативная память NVRAM маршрутизатора является перепрограммируемым запоминающим устройством (ППЗУ). NVRAM хранит стартовый (startup) конфигурационный файл, который после изменения конфигурации перезаписывается в ППЗУ, где создается резервная копия (backup). Конфигурационные файлы содержат команды и параметры для управления потоком трафика, проходящим через маршрутизатор. Конфигурационный файл используется для выбора сетевых протоколов и протоколов маршрутизации, которые определяют наилучший путь для пакетов к адресуемой сети. Первоначально конфигурационный файл обычно создается с консольной линии (console) и помимо памяти NVRAM может сохраняться на TFTP-сервере (рис. 6.4). Временное хранение входящих и исходящих пакетов обеспечивается в памяти интерфейсов, которые могут быть выполнены на материнской плате или в виде отдельных модулей.

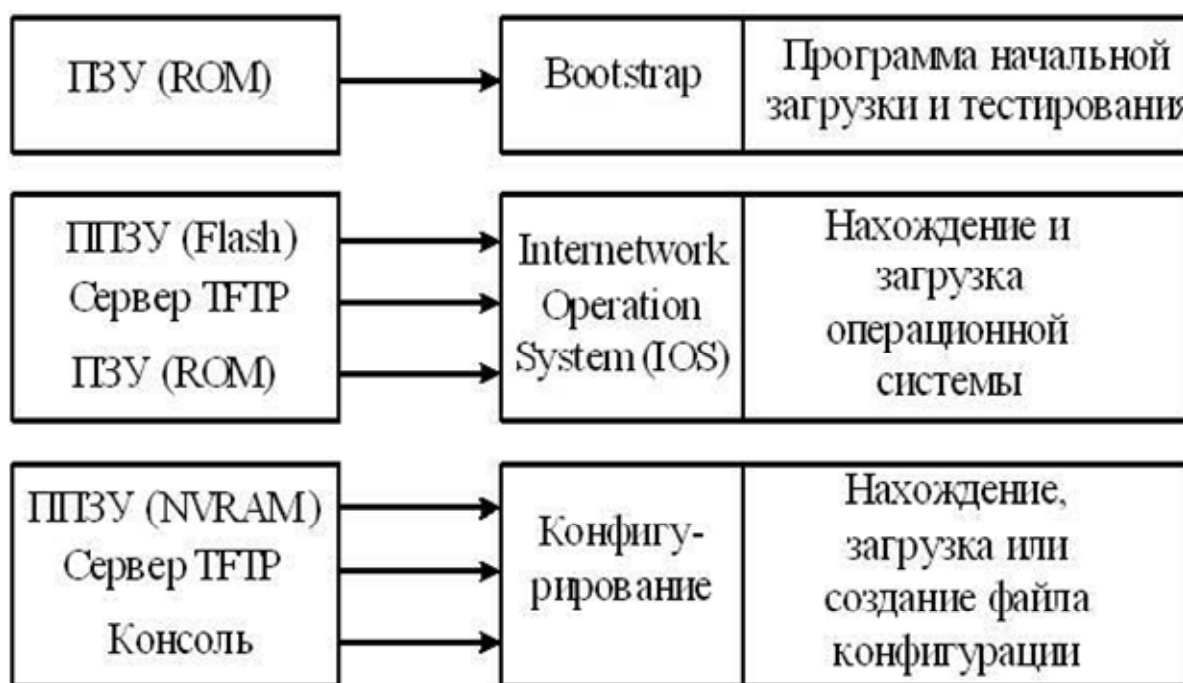


Рисунок 39 – Элементы памяти и программы маршрутизатора

При включении маршрутизатора начинает функционировать программа начальной загрузки bootstrap, которая тестирует оборудование и загружает операционную систему IOS в оперативную память RAM. В оперативную память загружается также конфигурационный файл, хранящийся в NVRAM. В процессе конфигурирования маршрутизатора задаются адреса интерфейсов, пароли, создаются таблицы маршрутизации, устанавливаются протоколы, проводится проверка параметров. Процесс коммутации и продвижения данных проходит под управлением операционной системы.

Принципы маршрутизации

Информационный поток данных, созданный на прикладном уровне, на транспортном уровне "нарезается" на сегменты, которые на сетевом уровне снабжаются заголовками и образуют пакеты. Заголовок пакета содержит сетевые IP-адреса узла назначения и узла источника. На основе этой информации средства сетевого уровня – маршрутизаторы осуществляют передачу пакетов между конечными узлами составной сети по определенному маршруту.

Маршрутизатор оценивает доступные пути к адресату назначения и выбирает наиболее рациональный маршрут на основе некоторого критерия – метрики. При оценке возможных путей маршрутизаторы используют информацию о топологии сети. Эта информация может быть сконфигурирована сетевым администратором или собрана в ходе динамического процесса обмена информацией между маршрутизаторами, который выполняется в сети протоколами маршрутизации.

Пакет, принятый на одном (входном) интерфейсе, маршрутизатор должен отправить (продвинуть) на другой (выходной) интерфейс (порт), который соответствует наилучшему пути к адресату. Чтобы передать пакеты от исходной сети (от источника) до сети адресата (назначения), на сетевом Уровне 3 маршрутизаторы используют таблицы маршрутизации для определения наиболее рационального пути.

Процесс прокладывания маршрута происходит последовательно от маршрутизатора к маршрутизатору. При прокладывании пути для пакета каждый маршрутизатор анализирует сетевую часть адреса узла назначения, заданного в заголовке поступившего пакета, т.е. вычленяет адрес сети назначения. Затем маршрутизатор обращается к таблице маршрутизации, в которой хранятся адреса всех доступных сетей, и определяет свой выходной интерфейс, на который необходимо передать (продвинуть) пакет. Таким образом, маршрутизатор ретранслирует пакет, продвигая его с входного интерфейса на выходной, для чего использует сетевую часть адреса назначения, обращаясь к таблице маршрутизации.

Выходной интерфейс связан с наиболее рациональным маршрутом к адресату. Конечный маршрутизатор на пути пакета непосредственно (прямо) связан с сетью назначения. Он использует часть сетевого адреса, содержащую адрес узла назначения, чтобы доставить пакет получателю данных.

Процесс ретрансляции пакетов маршрутизаторами рассмотрен на примере сети, приведенной на рис. 40. Маршрутизаторы в целом сетевого адреса не имеют, но поскольку они связывают между собой несколько сетей, каждый интерфейс (порт) маршрутизатора имеет уникальный адрес, сетевая часть которого совпадает с номером сети, соединенной с данным интерфейсом. Последовательные (serial) порты, соединяющие между собой маршрутизаторы, на рисунке обозначены молниевидной линией.

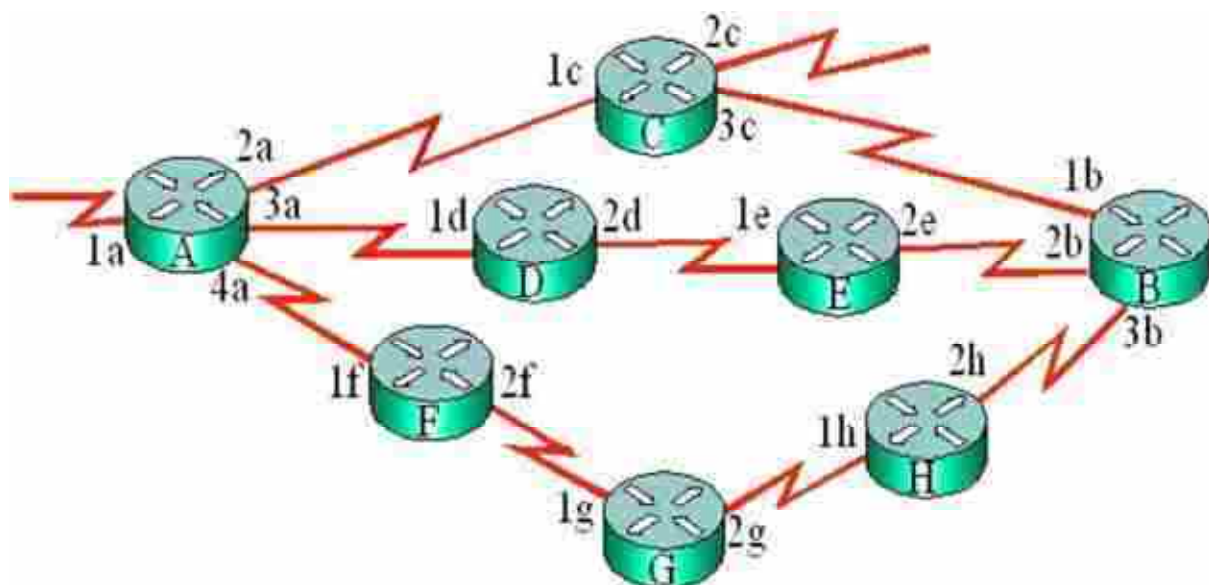


Рисунок 40 – Определения пути пакета

Путь от маршрутизатора А к маршрутизатору В может быть выбран:

- через маршрутизатор С;
- через маршрутизаторы D и E;
- через маршрутизаторы F, G и H.

Оценка наилучшего пути производится на основе метрики. Например, если метрика учитывает только количество маршрутизаторов на пути к адресату, то будет выбран первый маршрут. Если же метрика учитывает полосу пропускания линий связи, соединяющих маршрутизаторы, то может быть выбран второй или третий маршрут при условии, что на этом пути наиболее широкополосные линии связи.

При выборе первого пути функция коммутации реализуется за счет продвижения поступившего на интерфейс 1а маршрутизатора А пакета на интерфейс 2а. Таким образом, пакет попадает на интерфейс 1с маршрутизатора С, который продвинет полученный пакет на свой выходной интерфейс 3с, т. е. передаст полученный пакет маршрутизатору В.

В процессе передачи пакета по сети используются как сетевые логические адреса (IP-адреса), так и физические адреса устройств (MAC-адреса в сетях Ethernet). Например, при передаче информации с компьютера Host X локальной сети Сеть 1, (рис. 41) на компьютер Host Y, находящийся в удаленной Сети 2, определен маршрут через маршрутизаторы А, В, С.

Когда узел Host X Сети 1 передает пакет адресату Host Y из другой Сети 2, ему известен сетевой IP-адрес получателя, который записывается в заголовке пакета, т. е. известен адрес 3-го уровня. При инкапсуляции пакета в кадр источник информации Host X должен задать в заголовке кадра канальные адреса назначения и источника, т. е. адрес 2-го уровня (табл. 6).

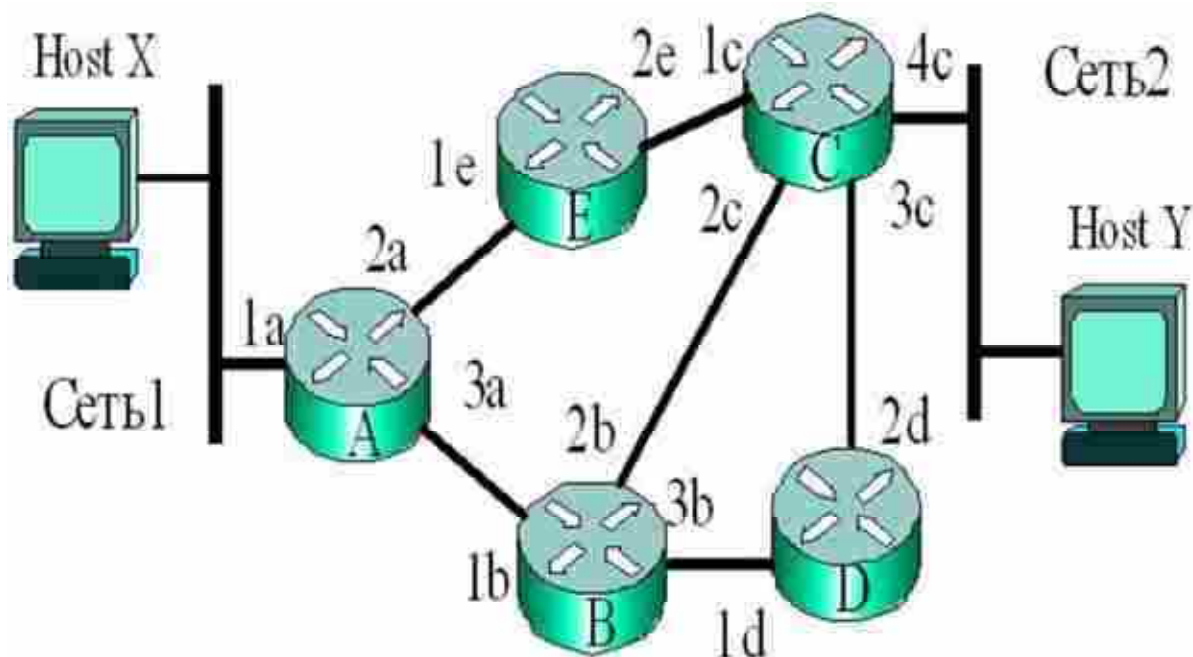


Рисунок 41 – Использование маршрутизаторов для передачи данных по сети

Таблица 6 – Основные поля кадра

Заголовок кадра		Заголовок пакета		Поле данных	Концевик (трейлер)
MAC-адрес назначения	MAC-адрес источника	IP-адрес назначения	IP-адрес источника	Данные	Контрольная сумма

У передающего узла нет информации об адресе канального уровня (MAC-адресе) узла назначения Host Y, поэтому Host X в заголовке кадра в качестве адреса назначения задаст MAC-адрес входного интерфейса 1a маршрутизатора A. Именно через этот интерфейс, называемый шлюзом по умолчанию (Default gateway), все пакеты из локальной Сети 1 будут передаваться в удаленные сети. Однако и этот адрес источнику информации Host X не известен. Процесс нахождения MAC-адреса по известному сетевому адресу реализуется с помощью протокола разрешения адресов Address Resolution Protocol – ARP, который входит в стек протоколов TCP/IP.

Протокол ARP

В локальных сетях телекоммуникаций на основе дейтаграмм устройствам необходимы как MAC-адрес, так и IP-адрес, которые для каждого узла образуют соответствующую пару. На каждом конечном узле можно посмотреть его

физический адрес и IP-адрес по команде `ipconfig /all` (рис. 42). Из распечатки следует, что физическим MAC-адресом конечного узла является 00-19-D1-93-7E-BE, а логическим IP-адресом – 10.0.118.52.

Протокол ARP может по IP-адресу автоматически определить MAC-адрес устройства. Каждое устройство в сети поддерживает таблицу ARP table, которая содержит соответствующие MAC- и IP-адреса других устройств той же локальной сети. Таблица ARP любого узла может быть просмотрена по команде `arp -a` (рис. 43). Записи таблицы хранятся в памяти RAM, где динамически поддерживаются.

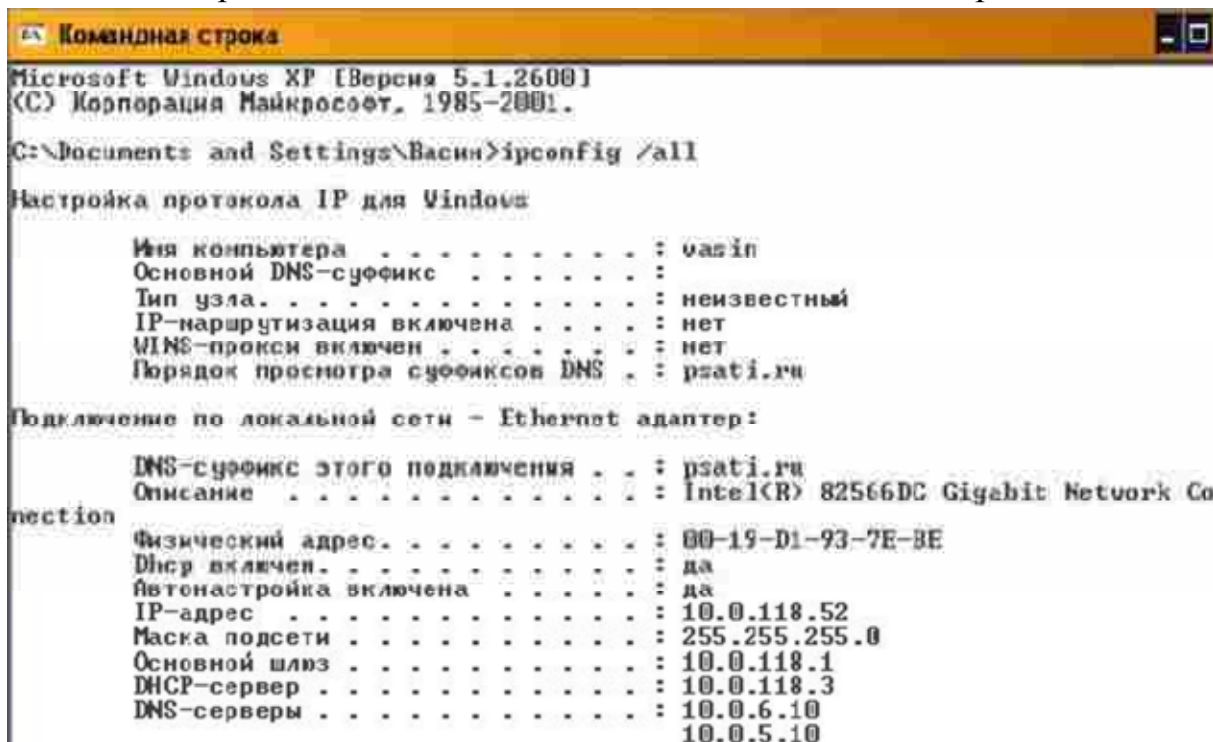


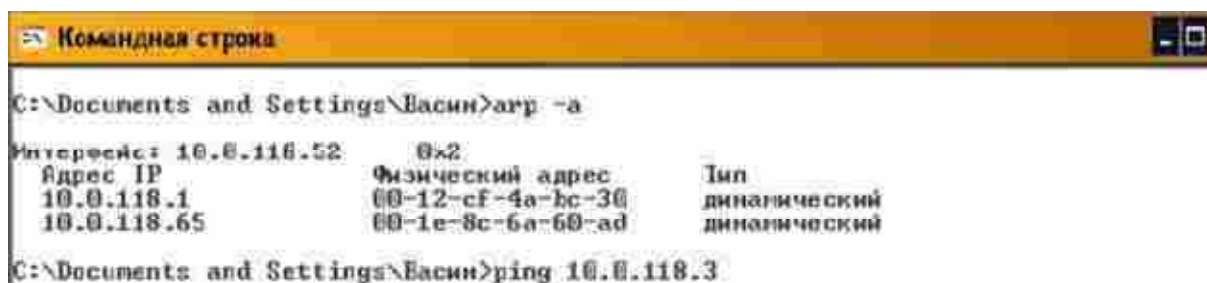
Рисунок 42 – Результат выполнения команды `ipconfig /all`

Если узлы долго не передают данные, то соответствующие записи из таблицы удаляются, что представлено на рис. 43, где таблица содержит только одну пару IP- и MAC-адресов.



Рисунок 43 – Таблица ARP

Таблица ARP пополняется динамически путем контроля трафика локального сегмента сети. Все станции локальной сети Ethernet анализируют трафик, чтобы определить, предназначены ли данные для них. При этом IP- и MAC-адреса источников дейтаграмм записываются в таблице ARP. Например, после общения с узлом 10.0.118.65 в таблице ARP появляется вторая запись (рис. 44).



```
Командная строка
C:\Documents and Settings\Васин>arp -a

Интерфейс: 10.0.118.52      0x2
Адрес IP      Физический адрес      Тип
10.0.118.1     00-12-cf-4a-bc-30     динамический
10.0.118.65    00-1e-8c-6a-60-ad     динамический

C:\Documents and Settings\Васин>ping 10.0.118.3
```

Рисунок 44 – Изменения в таблице ARP

Когда устройство передает пакет по IP-адресу назначения, оно проверяет, имеется ли в ARP-таблице соответствующий MAC-адрес назначения. Если соответствующая запись имеется, то она используется при инкапсуляции пакета в кадр данных. Данные передаются по сетевой среде, устройство назначения принимает их.

Если узел не находит соответствующей записи в таблице ARP, то он для получения MAC-адреса назначения посылает в локальную сеть широковещательный ARP-запрос, в котором задается сетевой логический IP-адрес устройства назначения. Все другие устройства сети анализируют его. Если у одного из локальных устройств IP-адрес совпадает с запрашиваемым, то устройство посылает ARP-ответ, который содержит пару IP- и MAC-адресов. Эта пара записывается в ARP-таблице. Если в локальной сети нет запрашиваемого IP-адреса, то устройство-источник сообщает об ошибке.

Когда данные передаются за пределы локальной сети, то для передачи сообщения необходимы IP- и MAC-адреса как устройства назначения, так и промежуточных маршрутизирующих устройств. Поскольку маршрутизаторы не транслируют широковещательные запросы в другие сегменты сети, в этом случае маршрутизатор в ответ на запрос посылает ARP-ответ с MAC-адресом своего входного интерфейса, на который поступил запрос. Таким образом, сформированный конечным устройством кадр поступит на интерфейс маршрутизатора, который после анализа адреса сети назначения и обращения к таблице маршрутизации продвинет пакет на выходной интерфейс.

Передать данные по адресу устройства, которое находится в другом сегменте сети, можно также за счет установки шлюза по умолчанию. Шлюз по умолчанию имеет IP-адрес входного интерфейса маршрутизатора на пути к устройству назначения. Этот адрес хранится в конфигурационном файле конечного узла (хоста). Источник сообщения сравнивает IP-адрес назначения со своим IP-адресом и определяет, находятся ли эти адреса в одном сегменте сети или в разных сегментах. Если они находятся в разных сегментах, то данные будут переданы только при условии, что установлен шлюз по умолчанию.

Таким образом, при передаче данных по сети (рис. 41) Host X для нахождения MAC-адреса назначения посылает в сеть широковещательный ARP запрос, в котором задается IP-адрес устройства назначения, на который Router A в ответ

посылает MAC-адрес своего входного интерфейса, и передаваемый пакет поступает в маршрутизатор.

Маршрутизатор А извлекает пакет из кадра, обрабатывает заголовок поступившего пакета, использует таблицу маршрутизации, чтобы определить сеть адресата, и затем продвигает пакет к выходному интерфейсу. Пакет вновь инкапсулируется в новый кадр данных и направляется следующему маршрутизатору В, при этом в заголовке кадра может указываться новый MAC-адрес входного интерфейса этого маршрутизатора. Этот процесс происходит каждый раз, когда пакет проходит через очередной маршрутизатор. В конечном маршрутизаторе (в данном примере – маршрутизатор С, рис. 41), который связан с сетью узла назначения Сеть 2, пакет инкапсулируется в кадр локальной сети адресата с MAC-адресом устройства назначения и доставляется адресату Host Y.

Для продвижения пакета к узлу назначения маршрутизатор использует таблицу маршрутизации, основными параметрами которой являются номер (адрес) сети назначения и сетевой адрес входного интерфейса следующего маршрутизатора на пути к адресату назначения. Этот адрес интерфейса получил название следующего перехода (next hop).

Таким образом, в таблице задаются:

- адрес сети назначения;
- адрес следующего перехода;
- другие дополнительные параметры, которые различаются для разных маршрутизирующих протоколов и маршрутизаторов разных фирм, производящих оборудование.

Из дополнительных параметров в таблицы маршрутизации включается информация:

- о статической или динамической маршрутизации,
- о типе используемых протоколов маршрутизации,
- о метрике, используемой при выборе возможного пути.

Принцип построения таблиц маршрутизации рассмотрен на примере сети, построенной на маршрутизаторах и коммутаторах (рис. 45). Последовательные (serial) интерфейсы маршрутизаторов на рис. 45 соединены между собой молниевидной линией, а порты Fast Ethernet – прямой линией. В приведенной схеме, например, D-f1 означает – первый Fast Ethernet порт маршрутизатора D, B-s2 – второй последовательный порт маршрутизатора В.

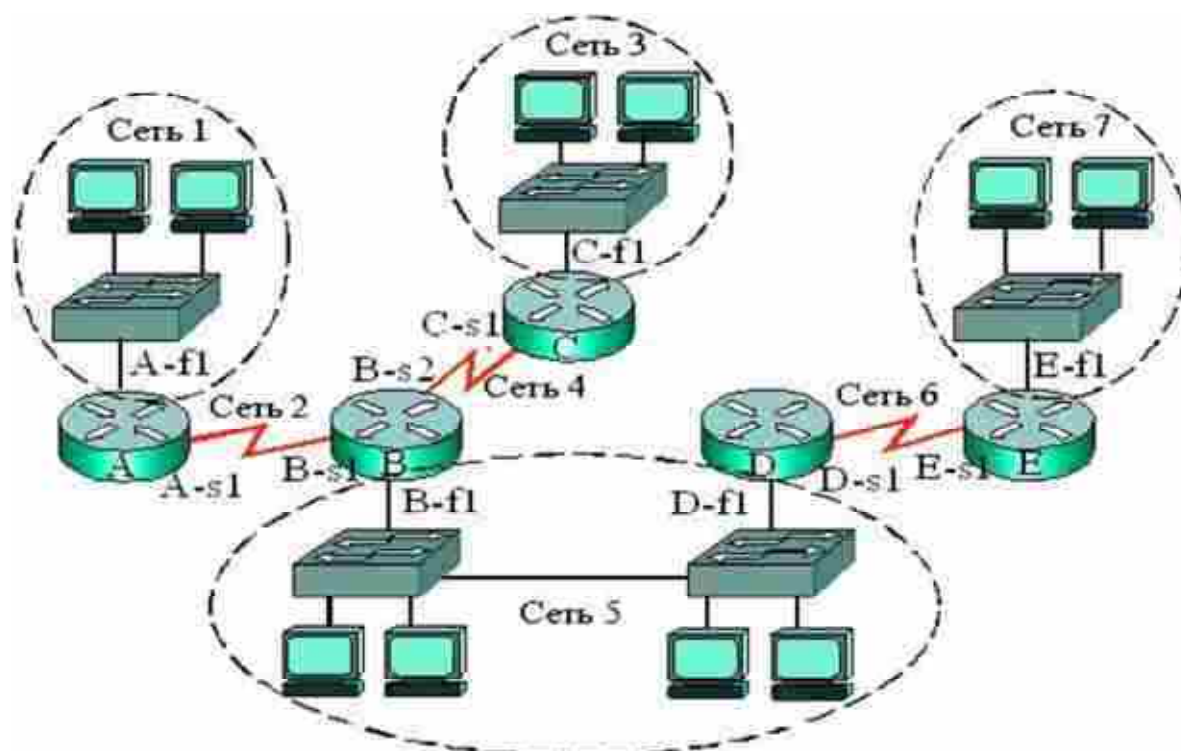


Рисунок 45 – Принцип маршрутизации в сети

Таблица маршрутизации, например маршрутизатора В (таблица 7), будет содержать информацию о маршрутах ко всем сетям (рис. 45). Маршрут к Сети 1 лежит через последовательный интерфейс A-s1 маршрутизатора А, к Сети 3 – через последовательный интерфейс C-s1 маршрутизатора С, а к сетям Сеть 6, Сеть 7 – через интерфейс D-f1 маршрутизатора D. Адреса входных интерфейсов маршрутизаторов на пути следования пакета к адресату назначения называются адресами следующего перехода (next hop).

Таблица 7 Основные параметры таблицы маршрутизации

Адрес сети назначения	Адрес следующего перехода
Сеть 1	A-s1
Сеть 3	C-s1
Сеть 6	D-f1
Сеть 7	D-f1

Вместо адреса следующего перехода часто указывают обозначение выходного интерфейса маршрутизатора, отправляющего пакет. Поскольку выходной интерфейс маршрутизатора, отправляющего пакет, и входной интерфейс следующего маршрутизатора на пути к адресату назначения соединены между собой, противоречий при этом никаких нет. Кроме удаленных сетей назначения в таблице маршрутизации указываются непосредственно (прямо) присоединенные сети с указанием выходного интерфейса. Например, таблица маршрутизации В (таблица 8) будет содержать три прямо присоединенных сети.

Таблица 8 Прямо присоединенные сети таблицы маршрутизации

Адрес присоединенной сети	Выходной интерфейс
Сеть 2	s1
Сеть 4	S2
Сеть 5	f1

Таким образом, пакет, предназначенный одному из узлов сети, например Сети 7, попав в маршрутизатор В, будет направлен на входной интерфейс D-f1 маршрутизатора D (следующий переход). В свою очередь, в таблице маршрутизации D будет задан адрес входного интерфейса E-s1 следующего маршрутизатора E, для которого Сеть 7 является непосредственно присоединенной. Поэтому маршрутизатор E направит пакет узлу назначения.

В Интернете нашли применение два основных протокола транспортного уровня, один из которых ориентирован на соединение, другой — нет. Протоколом без установления соединения является UDP. Протокол TCP, напротив, ориентирован на соединение. Так как UDP — это, на самом деле, просто IP с добавлением небольшого заголовка, мы изучим сперва его. Рассмотрим также два практических применения UDP.

Основы UDP

UDP (User Datagram Protocol — пользовательский дейтаграмм-ный протокол). UDP позволяет приложениям отправлять инкапсулированные IP-дейтаграммы без установления соединений. UDP описан в RFC 768.

С помощью протокола UDP передаются сегменты, состоящие из 8-байтного заголовка, за которым следует поле полезной нагрузки. Два номера портов служат для идентификации конечных точек внутри отправляющей и принимающей машин. Когда прибывает пакет UDP, содержимое его поля полезной нагрузки передается процессу, связанному с портом назначения. В сущности, весь смысл использования UDP вместо обычного IP заключается как раз в указании портов источника и приемника. Без этих двух полей на транспортном уровне невозможно было бы определить действие, которое следует произвести с пакетом. В соответствии с полями портов производится корректная доставка сегментов.

Информация о порте источника требуется прежде всего при создании ответа, пересылаемого отправителю. Копируя значения поля Порт источника из входящего сегмента в поле Порт назначения исходящего сегмента, процесс, посылающий ответ, может указать, какому именно процессу на противоположной стороне он предназначен.

Поле Длина UDP содержит информацию о длине сегмента, включая заголовок и полезную нагрузку. Контрольная сумма UDP не является обязательной. Если она не подсчитывается, ее значение равно 0 (настоящая нулевая контрольная сумма кодируется всеми единицами).

Отключать функцию подсчета контрольной суммы глупо, за исключением одного случая — когда нужна высокая производительность (например, при передаче оцифрованной речи).

Наверное, стоит прямо сказать о том, чего UDP не делает. Итак, UDP не занимается контролем потока, контролем ошибок, повторной передачей после приема испорченного сегмента. Все это перекладывается на пользовательские процессы. Что же он делает? UDP предоставляет интерфейс для IP путем демультиплексирования нескольких процессов, использующих порты. Это все, что он делает. Для процессов, которым хочется управлять потоком, контролировать ошибки и временные интервалы, протокол UDP — это как раз то, что доктор прописал.

Одной из областей, где UDP применяется особенно широко, является область клиент-серверных приложений. Зачастую клиент посылает короткий запрос серверу и надеется получить короткий ответ. Если запрос или ответ теряется, клиент по прошествии определенного временного интервала может попытаться еще раз. Это позволяет не только упростить код, но и уменьшить требуемое количество сообщений по сравнению с протоколами, которым требуется начальная настройка.

DNS (Domain Name System — служба имен доменов) — это приложение, которое использует UDP именно так, как описано выше. В двух словах, если программе нужно найти IP-адрес по имени хоста, например, www.vanderboot.ru, она может послать UDP-пакет с этим именем на сервер DNS. Сервер в ответ на запрос посылает UDP-пакет с IP-адресом хоста. Никакой предварительной настройки не требуется, как не требуется и разрыва соединения после завершения задачи. По сети просто передаются два сообщения.

Транспортные протоколы - TCP

UDP является простым протоколом и имеет определенную область применения. В первую очередь, это клиент-серверные взаимодействия и мультимедиа. Тем не менее, большинству интернет-приложений требуется надежная, последовательная передача. UDP не удовлетворяет этим требованиям, поэтому требуется иной протокол. Такой протокол называется TCP, и он является рабочей лошадкой Интернета.

Основы TCP

Протокол TCP (Transmission Control Protocol — протокол управления передачей) был специально разработан для обеспечения надежного сквозного байтового потока по ненадежной интерсети. Объединенная сеть отличается от отдельной сети тем, что ее различные участки могут обладать сильно различающейся топологией, пропускной способностью, значениями времени задержки, размерами пакетов и другими параметрами. При разработке TCP основное внимание уделялось способности протокола адаптироваться к свойствам объединенной сети и отказоустойчивости при возникновении различных проблем.

Протокол TCP описан в RFC 793. Со временем были обнаружены различные ошибки и неточности, и по некоторым пунктам требования были изменены. Подробное описание этих уточнений и исправлений дается в RFC 1122. Расширения протокола приведены в RFC 1323.

Каждая машина, поддерживающая протокол TCP, обладает транспортной сущностью TCP, являющейся либо библиотечной процедурой, либо пользовательским процессом, либо частью ядра системы. В любом случае, транспортная сущность управляет TCP-потоками и интерфейсом с IP-уровнем. TCP-сущность принимает от локальных процессов пользовательские потоки данных, разбивает их на куски, не превосходящие 64 Кбайт (на практике это число обычно равно 460 байтам данных, что позволяет поместить их в один кадр Ethernet с заголовками IP и TCP), и посылает их в виде отдельных IP-дейтаграмм. Когда IP-дейтаграммы с TCP-данными прибывают на машину, они передаются TCP-сущности, которая восстанавливает исходный байтовый поток. Для простоты мы иногда будем употреблять «TCP» для обозначения транспортной сущности TCP (части программного обеспечения) или протокола TCP (набора правил). Из контекста будет понятно, что имеется в виду. Например, в выражении «Пользователь передает данные TCP» подразумевается, естественно, транспортная сущность TCP.

Уровень IP не гарантирует правильной доставки дейтаграмм, поэтому именно TCP приходится следить за истекшими интервалами ожидания и в случае необходимости заниматься повторной передачей пакетов. Бывает, что дейтаграммы прибывают в неправильном порядке. Восстанавливать сообщения из таких дейтаграмм обязан также TCP. Таким образом, протокол TCP призван обеспечить надежность, о которой мечтают многие пользователи и которая не предоставляется протоколом IP.

Тема 14. Передача данных по сети через сокеты

В библиотеке классов Java есть очень удобное средство, с помощью которых можно организовать взаимодействие между приложениями Java и апплетами, работающими как на одном и том же, так и на разных узлах сети TCP/IP. Это средство, родившееся в мире операционной системы UNIX, - так называемые сокеты (sockets).

Что такое сокеты?

Вы можете представить себе сокеты в виде двух розеток, в которые включен кабель, предназначенный для передачи данных через сеть. Переходя к компьютерной терминологии, скажем, что сокеты - это программный интерфейс, предназначенный для передачи данных между приложениями.

Прежде чем приложение сможет выполнять передачу или прием данных, оно должно создать сокет, указав при этом адрес узла IP, номер порта, через который будут передаваться данные, и тип сокета.

С адресом узла IP мы уже сталкивались. Номер порта служит для идентификации приложения. Заметим, что существуют так называемые "хорошо известные" (well known) номера портов, зарезервированные для различных приложений. Например, порт с номером 80 зарезервирован для использования серверами Web при обмене данными через протокол HTTP.

Что же касается типов сокетов, то их два - потоковые и датаграммные.

С помощью потоковых сокетов вы можете создавать каналы передачи данных между двумя приложениями Java в виде потоков, которые мы уже рассматривали во второй главе. Потоки могут быть входными или выходными, обычными или форматированными, с использованием или без использования буферизации. Скоро вы убедитесь, что организовать обмен данными между приложениями Java с использованием потоковых сокетов не труднее, чем работать через потоки с обычными файлами.

Заметим, что потоковые сокететы позволяют передавать данные только между двумя приложениями, так как они предполагают создание канала между этими приложениями. Однако иногда нужно обеспечить взаимодействие нескольких клиентских приложений с одним серверным или нескольких клиентских приложений с несколькими серверными приложениями. В этом случае вы можете либо создавать в серверном приложении отдельные задачи и отдельные каналы для каждого клиентского приложения, либо воспользоваться датаграммными сокетами. Последние позволяют передавать данные сразу всем узлам сети, хотя такая возможность редко используется и часто блокируется администраторами сети.

Для передачи данных через датаграммные сокететы вам не нужно создавать канал - данные посылаются непосредственно тому приложению, для которого они предназначены с использованием адреса этого приложения в виде сокета и номера порта. При этом одно клиентское приложение может обмениваться данными с несколькими серверными приложениями или наоборот, одно серверное приложение - с несколькими клиентскими.

К сожалению, датаграммные сокететы не гарантируют доставку передаваемых пакетов данных. Даже если пакеты данных, передаваемые через такие сокететы, дошли до адресата, не гарантируется, что они будут получены в той же самой последовательности, в которой были переданы. Потоковые сокететы, напротив, гарантируют доставку пакетов данных, причем в правильной последовательности.

Причина отсутствия гарантии доставки данных при использовании датаграммных сокетов заключается в использовании такими сокетами протокола UDP, который, в свою очередь, основан на протоколе с негарантированной доставкой IP. Потоковые сокететы работают через протокол гарантированной доставки TCP.

Работа с потоковыми сокетами

Как мы уже говорили, интерфейс сокетов позволяет передавать данные между двумя приложениями, работающими на одном или разных узлах сети. В процессе создания канала передачи данных одно из этих приложений выполняет роль сервера, а другое - роль клиента. После того как канал будет создан, приложения становятся равноправными - они могут передавать друг другу данные симметричным образом.

Рассмотрим этот процесс в деталях.

Инициализация сервера

Вначале мы рассмотрим действия приложения, которое на момент инициализации является сервером.

Первое, что должно сделать серверное приложение, это создать объект класса `ServerSocket`, указав конструктору этого класса номер используемого порта:

```
ServerSocket ss;  
ss = new ServerSocket(9999);
```

Заметим, что объект класса `ServerSocket` вовсе не является сокетом. Он предназначен всего лишь для установки канала связи с клиентским приложением, после чего создается сокет класса `Socket`, пригодный для передачи данных.

Установка канала связи с клиентским приложением выполняется при помощи метода `accept`, определенного в классе `ServerSocket`:

```
Socket s;  
s = ss.accept();
```

Метод `accept` приостанавливает работу вызвавшего потока до тех пор, пока клиентское приложение не установит канал связи с сервером. Если ваше приложение однопоточное, его работа будет блокирована до момента установки канала связи. Избежать полной блокировки приложения можно, если выполнять создание канала передачи данных в отдельном потоке.

Как только канал будет создан, вы можете использовать сокет сервера для образования входного и выходного потока класса `InputStream` и `OutputStream`, соответственно:

```
InputStream is;  
OutputStream os;  
is = s.getInputStream();  
os = s.getOutputStream();
```

Эти потоки можно использовать таким же образом, что и потоки, связанные с файлами.

Обратите также внимание на то, что при создании серверного сокета мы не указали адрес IP и тип сокета, ограничившись только номером порта.

Что касается адреса IP, то он, очевидно, равен адресу IP узла, на котором запущено приложение сервера. В классе `ServerSocket` определен метод `getInetAddress`, позволяющий определить этот адрес:

```
public InetAddress getInetAddress();
```

Тип сокета указывать не нужно, так как для работы с датаграммными сокетами предназначен класс `DatagramSocket`, который мы рассмотрим позже.

Инициализация клиента

Процесс инициализации клиентского приложения выглядит весьма просто. Клиент должен просто создать сокет как объект класса `Socket`, указав адрес IP серверного приложения и номер порта, используемого сервером:

```
Socket s;  
s = new Socket("localhost",9999);
```

Здесь в качестве адреса IP мы указали специальный адрес `localhost`, предназначенный для тестирования сетевых приложений, а в качестве номера порта - значение 9999, использованное сервером.

Теперь можно создавать входной и выходной потоки. На стороне клиента эта операция выполняется точно также, как и на стороне сервера:

```
InputStream is;  
OutputStream os;  
is = s.getInputStream();  
os = s.getOutputStream();
```

Передача данных между клиентом и сервером

После того как серверное и клиентское приложения создали потоки для приема и передачи данных, оба этих приложения могут читать и писать в канал данных, вызывая методы `read` и `write`, определенные в классах `InputStream` и `OutputStream`.

Ниже представлен фрагмент кода, в котором приложение вначале читает данные из входного потока в буфер `buf`, а затем записывает прочитанные данные в выходной поток:

```
byte buf[] = new byte[512];  
int lenght;  
lenght = is.read(buf);  
os.write(buf, 0, lenght);  
os.flush();
```


На базе потоков класса `InputStream` и `OutputStream` вы можете создать буферизованные потоки и потоки для передачи форматированных данных, о которых мы рассказывали раньше.

Завершение работы сервера и клиента

После завершения передачи данных вы должны закрыть потоки, вызвав метод `close`:

```
Яis.close();  
os.close();
```

Когда канал передачи данных больше не нужен, сервер и клиент должны закрыть сокет, вызвав метод `close`, определенный в классе `Socket`:

```
s.close();
```

Серверное приложение, кроме того, должно закрыть соединение, вызвав метод `close` для объекта класса `ServerSocket`:

```
ss.close();
```

Конструкторы класса Socket

Чаще всего для создания сокетов в клиентских приложениях вы будете использовать один из двух конструкторов, прототипы которых приведены ниже:

```
public Socket(String host,int port);  
public Socket(InetAddress address,int port);
```

Первый из этих конструкторов позволяет указывать адрес серверного узла в виде текстовой строки, второй - в виде ссылки на объект класса `InetAddress`. Вторым параметром задается номер порта, с использованием которого будут передаваться данные.

В классе `Socket` определена еще одна пара конструкторов, которая, однако не рекомендуется для использования:

```
public Socket(String host,  
int port, boolean stream);  
public Socket(InetAddress address,  
int port, boolean stream);
```

В этих конструкторах последний параметр определяет тип сокета. Если этот параметр равен `true`, создается потоковый сокет, а если `false` - датаграммный. Заметим, что для работы с датаграммными сокетами следует использовать класс `DatagramSocket`.

Методы класса Socket

Перечислим наиболее интересные, на наш взгляд, методы класса Socket.

Прежде всего, это методы `getInputStream` и `getOutputStream`, предназначенные для создания входного и выходного потока, соответственно:

```
public InputStream getInputStream();  
public OutputStream getOutputStream();
```

Эти потоки связаны с сокетом и должны быть использованы для передачи данных по каналу связи.

Методы `getInetAddress` и `getPort` позволяют определить адрес IP и номер порта, связанные с данным сокетом (для удаленного узла):

```
public InetAddress getInetAddress();  
public int getPort();
```

Метод `getLocalPort` возвращает для данного сокета номер локального порта:

```
public int getLocalPort();
```

После того как работа с сокетом завершена, его необходимо закрыть методом `close`:

```
public void close();
```

И, наконец, метод `toString` возвращает текстовую строку, представляющую сокет:

```
public String toString();
```

Использование датаграммных сокетов

Как мы уже говорили, датаграммные сокет не гарантируют доставку пакетов данных. Тем не менее, они работают быстрее потоковых и обеспечивают возможность широковещательной рассылки пакетов данных одновременно всем узлам сети. Последняя возможность используется не очень широко в сети Internet, однако в корпоративной сети Intranet вы вполне можете ей воспользоваться.

Для работы с датаграммными сокетами приложение должно создать сокет на базе класса `DatagramSocket`, а также подготовить объект класса `DatagramPacket`, в который будет записан принятый от партнера по сети блок данных.

Канал, а также входные и выходные потоки создавать не нужно. Данные передаются и принимаются методами `send` и `receive`, определенными в классе `DatagramSocket`.

Класс DatagramSocket

Рассмотрим конструкторы и методы класса `DatagramSocket`, предназначенного для соеяздания и использования датаграммных сокетов.

В классе `DatagramSocket` определены два конструктора, прототипы которых представлены ниже:

```
public DatagramSocket(int port);  
public DatagramSocket();
```

Первый из этих конструкторов позволяет определить порт для сокета, второй предполагает использование любого свободного порта.

Обычно серверные приложения работают с использованием какого-то заранее определенного порта, номер которого известен клиентским приложениям. Поэтому для серверных приложений больше подходит первый из приведенных выше конструкторов.

Клиентские приложения, напротив, часто применяют любые свободные на локальном узле порты, поэтому для них годится конструктор без параметров.

Кстати, с помощью метода `getLocalPort` приложение всегда может узнать номер порта, закрепленного за данным сокетом:

```
public int getLocalPort();
```

Прием и передача данных на датаграммном соquete выполняется с помощью методов `receive` и `send`, соответственно:

```
public void receive(DatagramPacket p);  
public void send(DatagramPacket p);
```

В качестве параметра этим методам передается ссылка на пакет данных (соответственно, принимаемый и передаваемый), определенный как объект класса `DatagramPacket`. Этот класс будет рассмотрен позже.

Еще один метод в классе `DatagramSocket`, которым вы будете пользоваться, это метод `close`, предназначенный для закрытия сокета:

```
public void close();
```

Напомним, что сборка мусора в Java выполняется только для объектов, находящихся в оперативной памяти. Такие объекты, как потоки и сокеты, вы должны закрывать после использования самостоятельно.

Класс DatagramPacket

Перед тем как принимать или передавать данные с использованием методов `receive` и `send` вы должны подготовить объекты класса `DatagramPacket`. Метод `receive` запишет в такой объект принятые данные, а метод `send` - перешлет данные из объекта класса `DatagramPacket` узлу, адрес которого указан в пакете.

Подготовка объекта класса DatagramPacket для приема пакетов выполняется с помощью следующего конструктора:

```
public DatagramPacket(byte ibuf[],  
    int ilength);
```

Этому конструктору передается ссылка на массив *ibuf*, в который нужно будет записать данные, и размер этого массива *ilength*.

Если вам нужно подготовить пакет для передачи, воспользуйтесь конструктором, который дополнительно позволяет задать адрес IP *iaddr* и номер порта *ipor* узла назначения:

```
public DatagramPacket(byte ibuf[],  
    int ilength,  
    InetAddress iaddr, int ipor);
```

Таким образом, информация о том, в какой узел и на какой порт необходимо доставить пакет данных, хранится не в сокете, а в пакете, то есть в объекте класса DatagramPacket.

Помимо только что описанных конструкторов, в классе DatagramPacket определены четыре метода, позволяющие получить данные и информацию об адресе узла, из которого пришел пакет, или для которого предназначен пакет.

Метод *getData* возвращает ссылку на массив данных пакета:

```
public byte[] getData();
```

Размер пакета, данные из которого хранятся в этом массиве, легко определить с помощью метода *getLength*:

```
public int getLength();
```

Методы *getAddress* и *getPort* позволяют определить адрес и номер порта узла, откуда пришел пакет, или узла, для которого предназначен пакет:

```
public InetAddress getAddress();  
public int getPort();
```

Если вы создаете клиент-серверную систему, в которой сервер имеет заранее известный адрес и номер порта, а клиенты - произвольные адреса и различные номера портов, то после получения пакета от клиента сервер может определить с помощью методов *getAddress* и *getPort* адрес клиента для установления с ним связи.

Если же адрес сервера неизвестен, клиент может посылать широковещательные пакеты, указав в объекте класса DatagramPacket адрес сети. Такая методика обычно используется в локальных сетях.

Адрес IP состоит из двух частей - адреса сети и адреса узла. Для разделения компонент 32-разрядного адреса IP используется 32-разрядная маска, в которой битам адреса сети соответствуют единицы, а битам адреса узла - нули.

Например, адрес узла может быть указан как 193.24.111.2. Исходя из значения старшего байта адреса, это сеть класса C, для которой по умолчанию используется маска 255.255.255.0. Следовательно, адрес сети будет такой: 193.24.111.0.